SecLab
Computer Security Laboratory

Teecert Labs

23rd May, 2022

vSGX

Virtualizing SGX Enclaves on AMD SEV

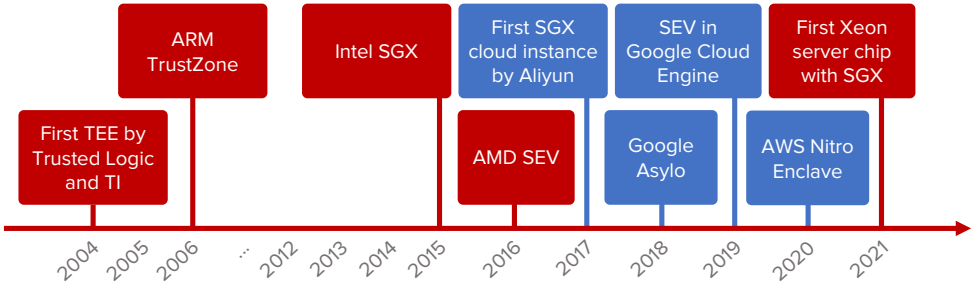
Shixuan Zhao
PhD Student @ SecLab
CSE, The Ohio State University
zhao.3289@osu.edu

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

SUSTech Southern University of Science and Technology

The Ohio State University
College of Engineering

Trusted Execution Environment



Intel SGX - An x86-64 TEE standard

Anonymity network

Game protection

Machine learning

Privacy-preserving data analytics

IoT network

Blockchains

Privacy-preserving contact-tracing

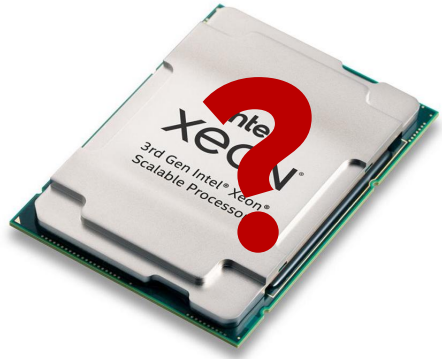
A problem of Intel SGX…

Vendor lock-in

Apps have to be written specifically for SGX and can't run else where

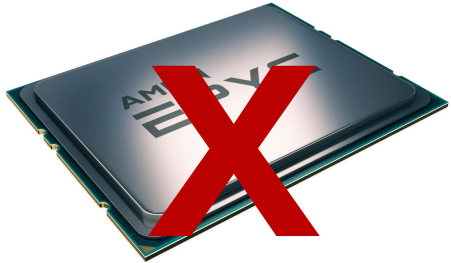
A problem of Intel SGX…

Vendor lock-in



A problem of Intel SGX…

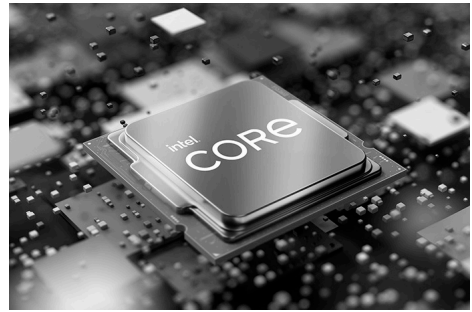
Vendor lock-in



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

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]

[1] A. Ferraiuolo, A. Baumann, C. Hawblitzel, and B. Parno. Komodo: Using verification to disentangle secure-enclave hardware from software. In Proc. of the 26th Symposium on Operating Systems Principles (SOSP '17), Oct. 2017

Software-defined TEE

	SGX	SEV	TrustZone	Komodo
Isolation	MMU	PSP	MMU	TrustZone
Interface	SGX	SEV	TrustZone	Komodo
Application	Enclave	OS/App	Secure OS/App	Enclave

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

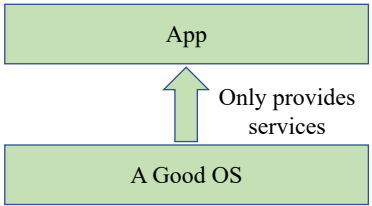
What our solution is

	SGX	SEV	Komodo
Isolation	MMU	PSP	TrustZone
Interface	SGX	SEV	Komodo
Application	Enclave	OS/App	Enclave

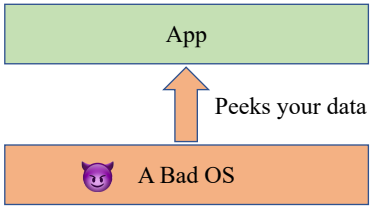
What our solution is

	SGX	SEV	Komodo	vSGX
Isolation	MMU	PSP	TrustZone	SEV
Interface	SGX	SEV	Komodo	SGX
Application	Enclave	OS/App	Enclave	Enclave

What is SGX

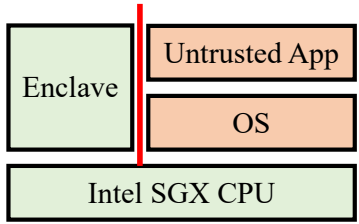


What is SGX



What is SGX

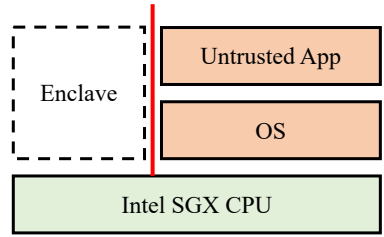
- Memory confidentiality
- Control flow integrity



SGX's workflow

Enclave initialization

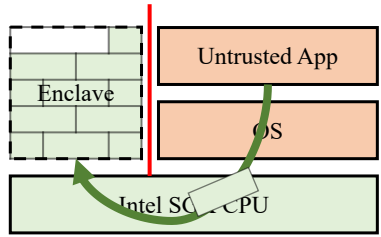
- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched



SGX's workflow

Enclave initialization

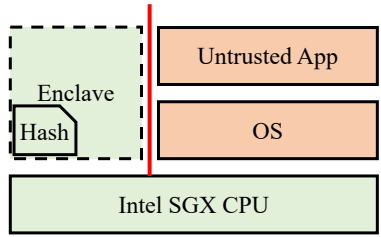
- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched



SGX's workflow

Enclave initialization

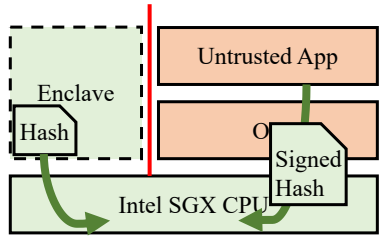
- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched



SGX's workflow

Enclave initialization

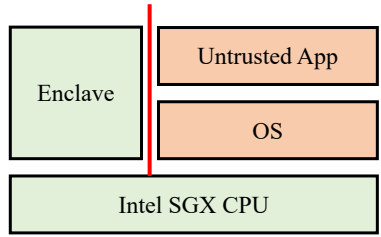
- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched



SGX's workflow

Enclave initialization

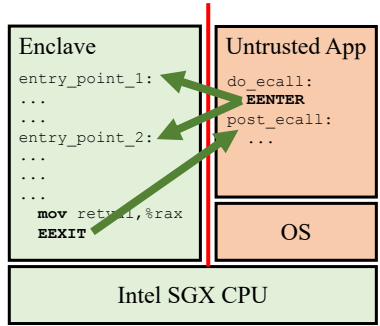
- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched



SGX's workflow

Control Flow

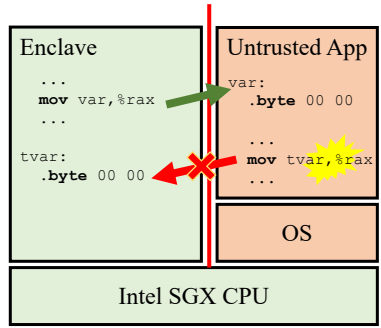
- Limited interface
- `EENTER` and `EEXIT`: Only to predefined entry points
- "ECalls": Intel SDK's wrapper



Enclave Memory in SGX

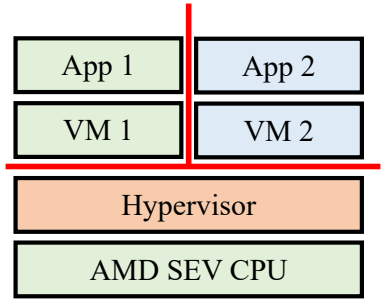
Memory Access

- Same virtual address space
- Single way trust

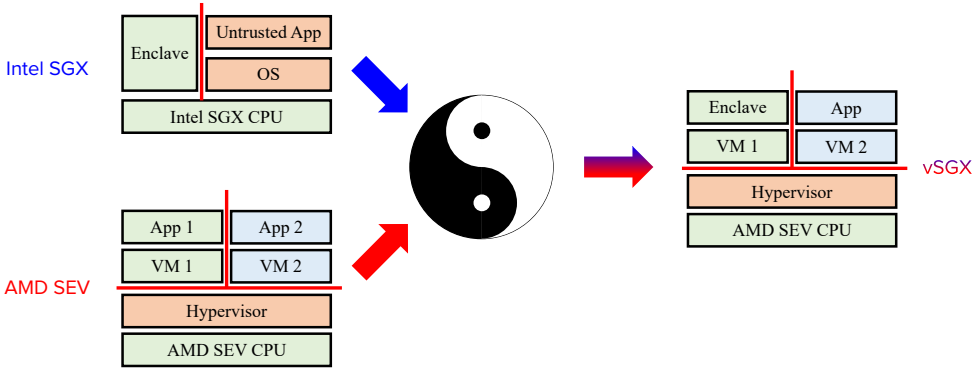


What is SEV (and friends)

- 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



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

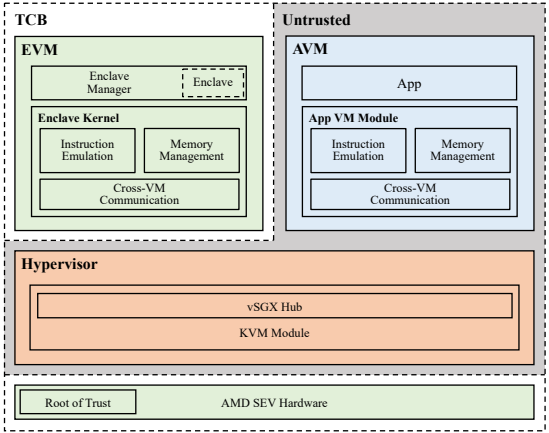
Designing the system comes with some nontrivial challenges

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

vSGX architecture

Enclave Isolation

- Two-VM architecture
- One enclave per VM

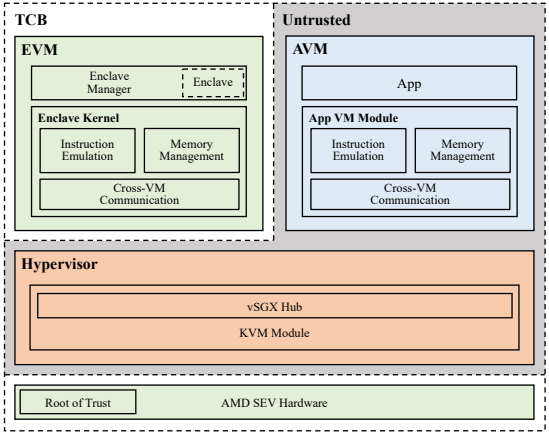


SecLab OSU CSE

Instruction emulation

Step 1: Interception

- Hook the #UD trap
- Check and emulate

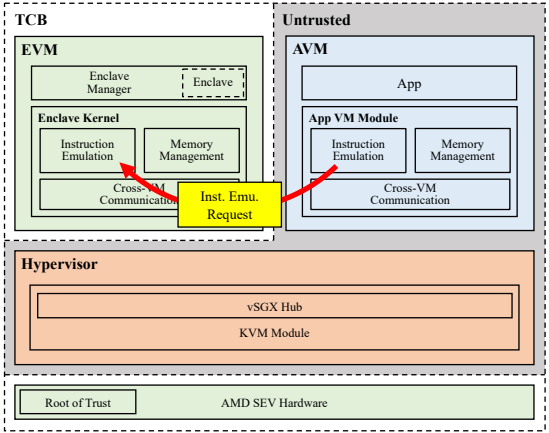


SecLab OSU CSE

Instruction emulation

Step 2: Emulation

- Accord to the Intel SGX's manual
- Send the request to EVM
- EVM send the result back

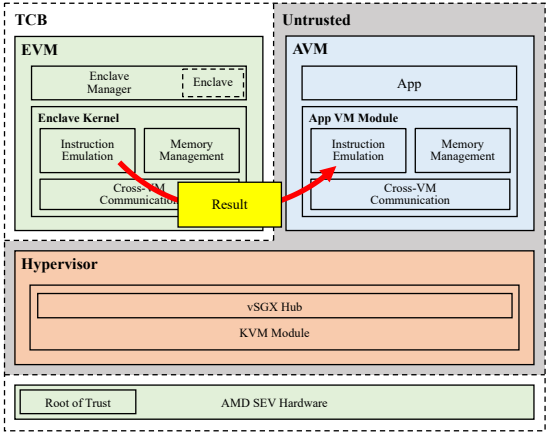


SecLab OSU CSE

Instruction emulation

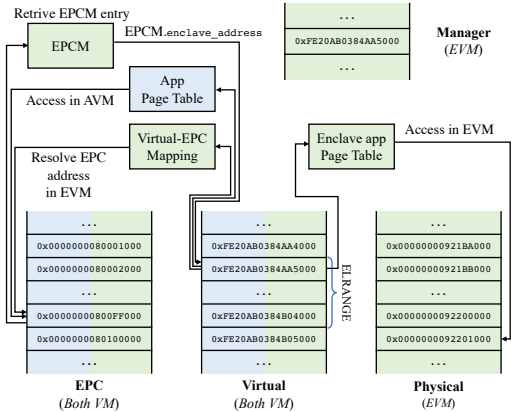
Step 2: Emulation

- 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"

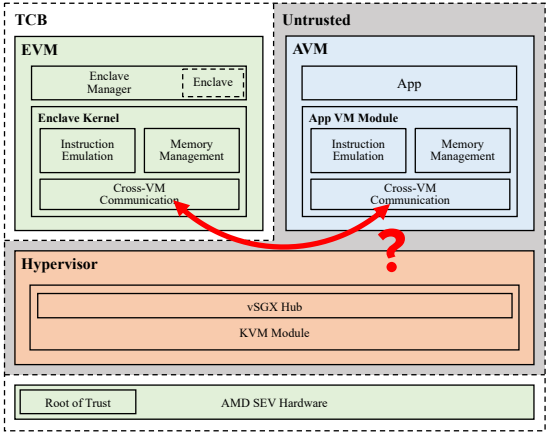


SecLab
OSU CSE

Cross-VM communication

Challenges

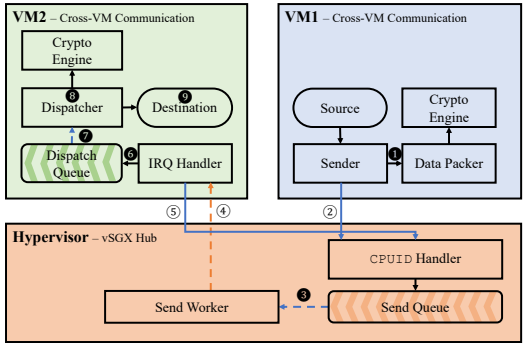
- SEV's security
- No data shall be leaked/altered/resent



SecLab OSU CSE

Cross-VM communication

- Encrypted & CMACed
- Replay protection

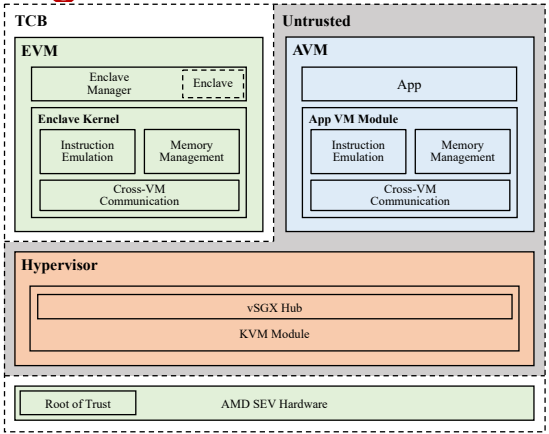


SecLab OSU CSE

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

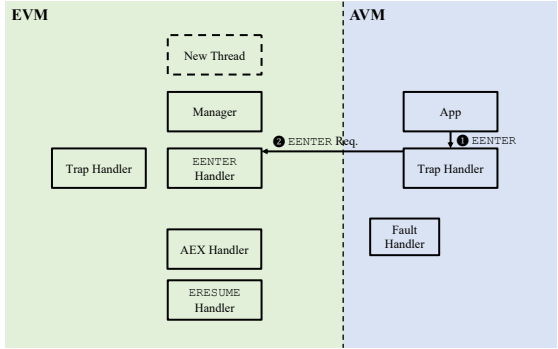


SecLab OSU CSE

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

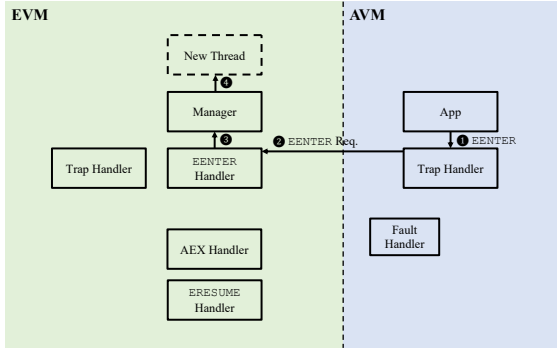


SecLab OSU CSE

Control flow transferring

EENTER

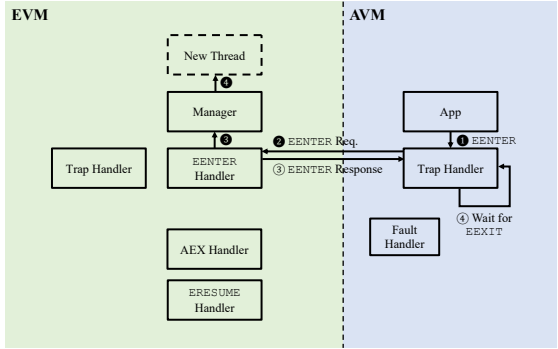
- The EENTER request is sent to the EVM
- A counterpart thread is launched within the EVM
- The AVM's app thread is put to sleep



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

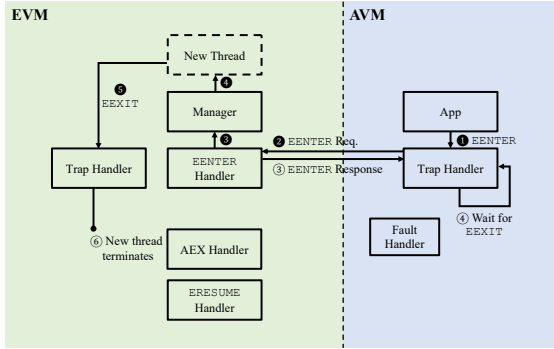


SecLab OSU CSE

Control flow transferring

EEXIT

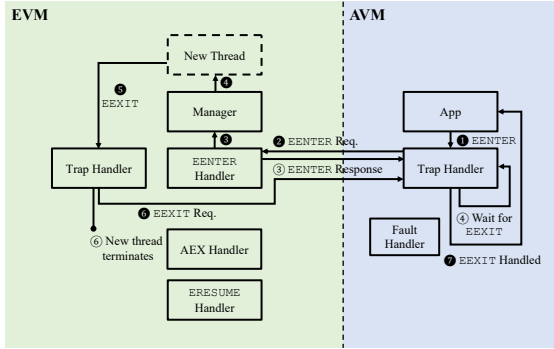
- The enclave thread is killed
- The AVM's thread is woken up



Control flow transferring

EEXIT

- The enclave thread is killed
- The AVM's thread is woken up

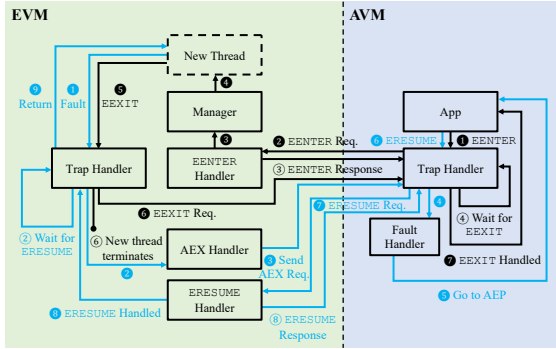


SecLab OSU CSE

Control flow transferring

AEX

Similar but reversed



SecLab OSU CSE

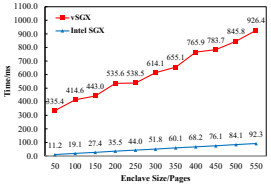
Prototype

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

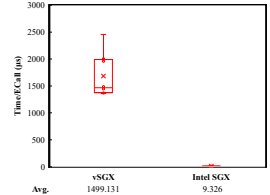
Capability tested

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

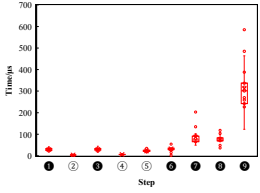
Performance - Microbenchmarks



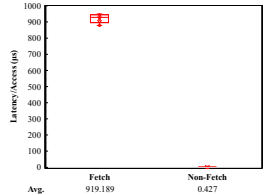
(a) **Enclave initialization overhead**



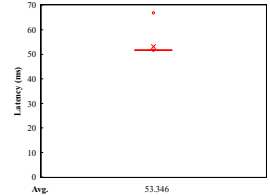
(b) **ECall overhead**



(c) **Cross-VM overhead**



(d) **Memory access latency**



(e) **Switchless syncing latency**

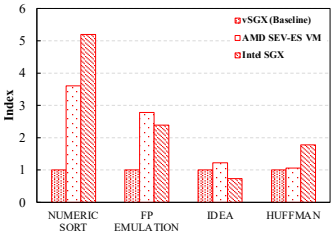
SecLab OSU CSE

Performance - Instructions

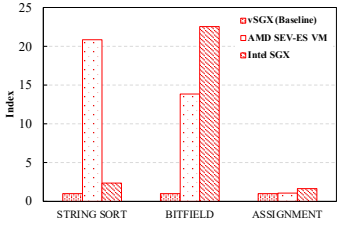
	Leaf	Average Overhead (µs)	Packets Sent
ENCLS	EADD	1421.23	3
	EAUG	990.20	2
	EBLOCK	840.85	2
	ECREATE	3719.06	3
	EDBGRD	N/A	N/A
	EDBGWR	N/A	N/A
	EEXTEND	986.76	2
	EINIT	811.03	2
	ELDB/ELDU	1958.13	4
	EMODPR	1071.26	2
	EMODT	976.15	2
	EPA	1273.26	3
	EREMOVE	1013.70	2
	ETRACK	N/A	N/A
	EWB	1818.66	4

	Leaf	Average Overhead (µs)
ENCLU	EACCEPT	0.79
	EACCEPTCOPY	2.19
	EENTER	N/A
	EEXIT	N/A
	EGETKEY	5.00
	EMODPE	0.91
	EREPORT	18.91
	ERESUME	N/A

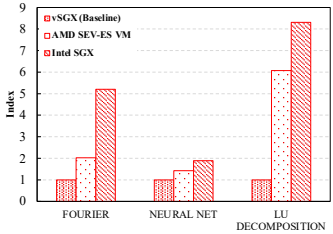
Performance - BYTEmark



(a) **CPU Intensive Test**



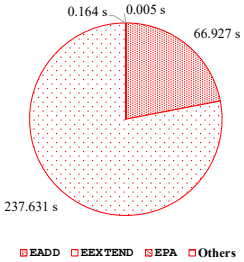
(b) **Memory Intensive Test**



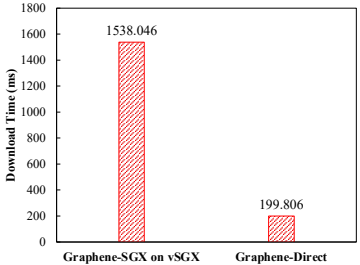
(c) **FP Intensive Test**

Significant performance drop only observed from I/O intensive workloads

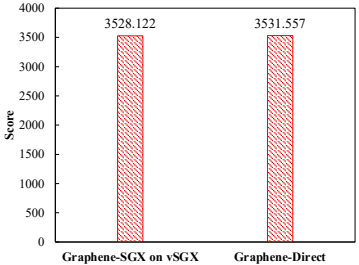
Performance - Graphene



(a) Time Consumption Launching Graphene SGX on vSGX



(b) cURL Execution Time



(c) GMPbench 0.2 Score

Performance - wolfCrypt

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

	vSGX	Intel SGX	Ratio
	MB/s	MB/s	
SHA-256	115.56	298.76	2.59
HMAC-MD5	377.70	821.12	2.17
HMAC-SHA	381.57	662.07	1.74
HMAC-SHA256	164.82	298.90	1.81
	KB/s	KB/s	
PBKDF2	9.49	34.63	3.65
	op/s	op/s	
RSA 2048 Public	10264.09	8443.25	0.82
RSA 2048 Private	188.40	146.93	0.78
DH 2048 Key Gen	378.24	374.80	0.99
DH 2048 Agree	614.50	375.19	0.61
ECC 256 Key Gen	453.50	6569.28	14.49
ECDHE 256 Agree	1461.67	2201.94	1.51
ECDSA 256 Sign	3611.59	5297.49	1.47
ECDSA 256 Verify	1336.96	1875.64	1.40
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

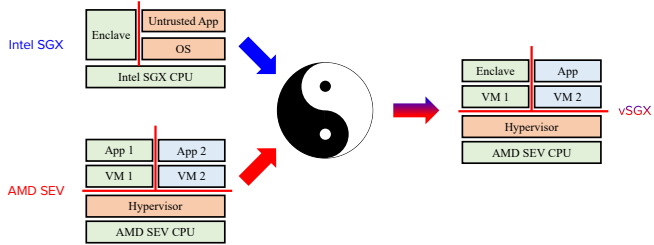
- vSGX: Virtualizing SGX enclaves on… Intel MKTME?

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