Towards Distributed and Virtualized Trusted Execution Environments

Zhiqiang Lin
zlin@cse.ohio-state.edu

07/14/2022
Trusted Execution Environments (TEEs)

A hardware ensured, **isolated** execution environment (i.e., enclave) provides security features such as isolated execution, **integrity** and **confidenality** of both **code** and **data** inside the enclave.
A hardware ensured, **isolated** execution environment (i.e., enclave) provides security features such as isolated execution, **integrity** and **confidenality** of both **code** and **data** inside the enclave.
Trusted Execution Environments (TEEs)

First TEE by Trusted Logic and TI
Trusted Execution Environments (TEEs)

First TEE by Trusted Logic and TI
ARM TrustZone
Trusted Execution Environments (TEEs)

- ARM TrustZone
- Intel SGX

First TEE by Trusted Logic and TI
Trusted Execution Environments (TEEs)

First TEE by Trusted Logic and TI

ARM TrustZone

Intel SGX

AMD SEV
Trusted Execution Environments (TEEs)

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
Trusted Execution Environments (TEEs)

- ARM TrustZone
- Intel SGX
- AMD SEV
- First SGX cloud instance by Aliyun
- SEV in Google Cloud Engine
- Intel MKTME

First TEE by Trusted Logic and TI
First SGX cloud instance by Aliyun
SEV in Google Cloud Engine
Intel MKTME

Why TEEs

Operating Systems

Hardware
Why TEEs

- Hardware
- Operating Systems
- Hardware
Why TEEs

![Diagram showing Why TEEs]

- Operating Systems
- Hardware

- Motivations
- Background
- Virtualized TEE
- TEE-based V2V Protocol
- Related Work
- Takeaway
- References
Why TEEs

Operating Systems

Hardware
Why TEEs
Why TEEs
Why TEEs

Hardware

Operating Systems

Hardware

Why TEEs
Why TEEs

Diagram:

- Operating Systems
- Virtualization
- Hardware

Why TEEs
Why TEEs

Operating Systems

Virtualization

Hardware
Why TEEs

- Operating Systems
- Operating Systems
- Virtualization
- Hardware
Why TEEs

Operating Systems

Virtualization

Hardware
Why TEEs

- Hardware
- Virtualization
- Operating Systems

Diagram showing TEE (Trust Execution Environment) as the central element, with Operating Systems and Virtualization as layers on top, and Hardware at the bottom.
Why TEEs
Sandbox vs. Reverse Sandbox
Sandbox vs. Reverse Sandbox

![Diagram showing the difference between Sandbox and Enclave](image-url)
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 Efforts

2. "SGX-Log: Securing System Logs With SGX". ASIACCS 2017
3. "Securing Data Analytics on SGX With Randomization". ESORICS 2017
8. "Towards Memory Safe Enclave Programming with Rust-SGX". CCS 2019
11. "vSGX: Virtualizing SGX on AMD SEV". Oakland 2022
12. "Towards A TEE-based V2V Protocol For Connected And Autonomous Vehicles". AutoSec 2022
Our Efforts

3. “Securing Data Analytics on SGX With Randomization”. ESORICS 2017
11. “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 → pps have to be written specifically for SGX and can’t run else where (e.g., you can’t run it on AMD chips)

2. Enumerous 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
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
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) TEEs
The Need of Distributed (Edge) TEEs

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

Security + Privacy
The Need of Distributed (Edge) TEEs

Benefits of Distributed TEEs

1. Distributed (no single point of failure, no leakage from centrals)
2. Near-data computing (efficient)
3. Privacy-preserving
4. Integrity-guarantee (provided by TEEs)
The Need of Distributed (Edge) TEEs

Traffic Safety
Traffic Efficiency
Energy Savings

Our goals: How to instantly authenticate a vehicle (Security)? How to protect a vehicle from being tracked (Privacy)?
### 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>EEXTEND</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>EWB</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: ECREATE, EADD; EEXTEND, EINIT
SGX 101: ECREATE, EADD; EEXTEND, EINIT
SGX 101: ECREATE, EADD; EEXTEND, EINIT
SGX 101: ECREATE, EADD; EEXTEND, EINIT
SGX 101: ECREATE, EADD; EEXTEND, EINIT
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
...

OS

Intel SGX CPU

SGX 101: EENTER, EEXIT; AEX, ERESUME
SGX 101: Enclave Memory in SGX

Enclave

...  
\texttt{mov \ var,} %rax 
...  
\texttt{tvar:}  
\texttt{.byte 00 00}  

Untrusted App

\texttt{var:}  
\texttt{.byte 00 00}  
...  
\texttt{mov tvar,} %rax  
...  

OS

Intel SGX CPU
AMD Secure Encrypted Virtualization (SEV)
vSGX: Virtualizing SGX Enclaves on AMD SEV
vSGX: Virtualizing SGX Enclaves on AMD SEV

Enclave

Intel SGX CPU

OS

Untrusted App

App 1

VM 1

Hypervisor

AMD SEV CPU

VM 2

App 2

Hypervisor

AMD SEV CPU

App

VM 1

VM 2

Hypervisor

AMD SEV CPU

vSGX

Design Goals

1. Binary compatibility
2. Comparable security guarantee with both SGX and SEV
3. 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?
vSGX Architecture
# 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>EADDD</td>
<td>Add an page to an uninitialized enclave</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>● ← ○</td>
<td>4185</td>
<td>19</td>
<td></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>25</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>EBLOCK</td>
<td>Block an EPC page</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>● ← ○</td>
<td>9</td>
<td>19</td>
<td></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>4105</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>EDBGDRD</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>9</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>EINIT</td>
<td>Initialize an enclave</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>● ← ○</td>
<td>2137</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>ELD/E/LDLU</td>
<td>Load a page to enclave</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>● ← ○</td>
<td>8370</td>
<td>4131</td>
<td></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>12</td>
<td>19</td>
<td></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>12</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>EPA</td>
<td>Add version array</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>● ← ○</td>
<td>9</td>
<td>4131</td>
<td></td>
</tr>
<tr>
<td>EREMOVE</td>
<td>Remove a page from EPC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>● ← ○</td>
<td>9</td>
<td>19</td>
<td></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>ENB</td>
<td>Write an EPC page to main memory</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>● ← ○</td>
<td>4137</td>
<td>8355</td>
<td></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>177</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>EEXIT</td>
<td>Exit an enclave</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>@ − ○</td>
<td>153</td>
<td></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>EMODPE</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>33</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

**Behavior**

- **AEX**: Exit an enclave due to interrupt or fault
  - ✓:** - ✓ - ✓ - ✓ - ○ → ○ - 166 -
Cross VM Communication

**VM2** – Cross-VM Communication

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

**VM1** – Cross-VM Communication

- Source
- Crypto Engine
- Sender
- Data Packer
- Dispatch Queue

**Hypervisor** – vSGX Hub

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

1. **Dispatch Queue**
2. **Send Worker**
Cross VM Communication

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

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

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
- Send Queue
- Send Worker

Hypervisor – vSGX Hub
- CPUID Handler
Cross VM Communication

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

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

**Hypervisor** – vSGX Hub

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

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

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

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

**VM2** – Cross-VM Communication

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

**VM1** – Cross-VM Communication

- **Source**
- **Sender**
- **Data Packer**
- **Crypto Engine**

**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
- CPUID Handler
- Send Queue
- Send Worker
Control Flow Transferring: EENTER, EEXIT

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

AVM
- App
- Trap Handler
- Fault Handler
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT

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

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

1. EENTER
2. EENTER Req.
3. EENTER Response
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT

EVM

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

AVM

- App
- Trap Handler
- Fault Handler
- EENTER
- EENTER Req.
- EENTER Response
- Wait for EEXIT

Diagram:

1. EENTER
2. EENTER Req.
3. EENTER Response
4. Wait for EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT

Diagram:

- EVM
  - Trap Handler
  - EENTER Handler
  - New Thread
  - EEXIT

- AVM
  - App
  - Trap Handler
  - Fault Handler
  - EENTER
  - EEXIT
  - Wait for EEXIT

Key Points:

1. EENTER
2. EENTER Request
3. EENTER Response
4. Wait for EEXIT
5. EEXIT
6. EEXIT Request
7. ERESUME Handler
8. AEX Handler
9. Manager
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT

EVM

1. Trap Handler
2. EENTER Request
3. EENTER Response
4. Wait for EEXIT
5. EEXIT
6. New thread terminates

AVM

1. EENTER
2. App
3. Fault Handler
4. Wait for EEXIT
5. EEXIT
6. EEXIT Handled
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
Connected and Automotive Vehicles (CAVs)

- Vehicle To Infrastructure (V2I)
- Vehicle To Vehicle (V2V)
- Benefits
  1. Traffic Safety
  2. Traffic Efficiency
  3. Energy Savings
V2V Protocol Requirements

1. **(Instant) Authentication**
2. Message Integrity (Tamper Resiliency)
3. Message Confidentiality (Privacy)
4. **Anti-tracking** (Privacy)
5. Revocation and **Accountability**
6. Scalability
PKI-based V2V Encrypted Broadcast

Vehicle V1

Vehicle V2
PKI-based V2V Encrypted Broadcast

Vehicle V1

Cert_{V1} \| G_{V1}

Vehicle V2
PKI-based V2V Encrypted Broadcast

Vehicle V1

Cert\textsubscript{V1} \parallel G\textsubscript{V1}

Verify Cert\textsubscript{V1}

Check Cert\textsubscript{V1} = \text{? CRL}\textsubscript{REVOKED}

Vehicle V2
PKI-based V2V Encrypted Broadcast

Vehicle V1

\[ \text{Cert}_{V1} \parallel G^{V1} \]

Vehicle V2

\[ \text{Cert}_{V2} \parallel G^{V2} \]

Verify \( \text{Cert}_{V1} \)

Check \( \text{Cert}_{V1} =? \ CRL_{REVOKED} \)
PKI-based V2V Encrypted Broadcast

- **Vehicle V1**
  - **Cert**\textsubscript{V1} || G\textsuperscript{V1}
  - Verify Cert\textsubscript{V1}
  - Check Cert\textsubscript{V1} =? CRL\textsubscript{REVOKED}
  - Derive K\textsubscript{DH} ← G\textsuperscript{V1*V2}

- **Vehicle V2**
  - **Cert**\textsubscript{V2} || G\textsuperscript{V2}
  - Verify Cert\textsubscript{V2}
  - Check Cert\textsubscript{V2} =? CRL\textsubscript{REVOKED}
  - Derive K\textsubscript{DH} ← G\textsuperscript{V2*V1}
PKI-based V2V Encrypted Broadcast

Vehicle V1

Cert_{V1} \parallel G^{V1}

Verify Cert_{V1}
Check Cert_{V1} =? CRL_{REVOKED}
Derive K_{DH} \leftarrow G^{V1 \cdot V2}

|BSM|K_{DH}, CMAC

Vehicle V2

Verify Cert_{V2}
Check Cert_{V2} =? CRL_{REVOKED}
Derive K_{DH} \leftarrow G^{V2 \cdot V1}
PKI-based V2V Encrypted Broadcast

Cert_{V1} || G_{V1}

Cert_{V2} || G_{V2}

Verify Cert_{V1}
Check Cert_{V1} =? CRL_{REVOKED}
Derive K_{DH} ← G^{V1*V2}

[BSM] K_{DH}, CMAC

Verify CMAC
Decrypt BSM
Process BSM

Derive K_{DH} ← G^{V2*V1}
Our Approach

Key Objective

1. **(Instant) Authentication**
2. Message Integrity (Tamper Resiliency)
3. Message Confidenality (Privacy)
4. **Anti-tracking** (Privacy)
5. Revocation and **Accountability**
6. Scalability
**Our Approach**

### Key Objective

1. **(Instant) Authentication**
2. Message Integrity (Tamper Resiliency)
3. Message Confidentiality (Privacy)
4. **Anti-tracking** (Privacy)
5. Revocation and **Accountability**
6. Scalability

### Our Approach

- Leveraging TEEs to protect secrets
- Using a shared Daily Symmetric (DS) Key among vehicles
- Storing DS key in the enclave
- Using Remote Attestation to ensure the right environment
- Generating temporal IDs for privacy
Our Approach

Key Objective

1. (Instant) Authentication
2. Message Integrity (Tamper Resiliency)
3. Message Confidentiality (Privacy)
4. Anti-tracking (Privacy)
5. Revocation and Accountability
6. Scalability

Our Approach

► Leveraging TEEs to protect secrets
► Using a shared Daily Symmetric (DS) Key among vehicles
► Storing DS key in the enclave
► Using Remote Attestation to ensure the right environment
► Generating temporal IDs for privacy

The Protocol: Manufacture Phase (One Time)

Collaborate and Integrate into Vehicle On-Board unit
The Protocol: Registration Phase (One Time)

- **Signature Keys:** $sk_{BMV}, pk_{BMV}$
- **Fused Keys:** $K_{Seal}, K_{PV}$
- **ID:** VIN

**Abbreviations:**
- BMV
- TEE Designer Server
- TEE-based V2V Protocol
The Protocol: Registration Phase (One Time)

- **Signature Keys:** $sk_{BMV}, pk_{BMV}$
- **Enclave Initialize Request**
- **Fused Keys:** $K_{Seal}, K_{PV}$
- **ID:** VIN

**Identification:**

- **TEE Designer Server**
  - $K_{PV}$
- **Ohio BMV**
  - $K_{Seal}$
- **Intel**
The Protocol: Registration Phase (One Time)

Fused Keys: $K_{Seal}, K_{PV}$

Enclave Initialize Request

ID: VIN

Signature Keys: $sk_{BMV}, pk_{BMV}$

TEE Designer Server

K$_{PV}$

TLS Connect

Enclave Initialize Request
The Protocol: Registration Phase (One Time)

Fused Keys: \( k_{\text{Seal}}, k_{PV} \)

Signature Keys: \( s_{BMV}, p_{BMV} \)

Enclave Initialize Request

TLS Connect

Session Key: \( K \)
The Protocol: Registration Phase (One Time)

- **Fused Keys:** $K_{\text{Seal}}, K_{\text{PV}}$
- **ID:** VIN
- **Signature Keys:** $sk_{BMV}, pk_{BMV}$
- **Session Key:** $K$

Diagram:
- Enclave Initialize Request
- TLS Connect
- TEE Designer Server
- {Enclave Program}$K$
The Protocol: Registration Phase (One Time)

**Fused Keys:** $K_{Seal}, K_{PV}$

**Signature Keys:** $sk_{BMV}, pk_{BMV}$

**ID:** VIN

Enclave Initialize Request

TLS Connect

Session Key: $K$

{Enclave Program}$K$

Enclave Install

Boot-Up Application

Check Enclave Integrity
The Protocol: Registration Phase (One Time)

Fused Keys: $K_{\text{Seal}}, K_{\text{PV}}$

ID: VIN

Signature Keys: $s_{\text{BMV}}, p_{\text{BMV}}$

TEE Designer Server

Enclave Initialize Request

TLS Connect

Session Key: $K$

(Enclave Program)$K$

Enclave Install

Boot-Up Application

Check Enclave Integrity

Remote Attest SGX Platform
The Protocol: Registration Phase (One Time)

**Fused Keys:** $K_{Seal}, K_{PV}$

**Signature Keys:** $sk_{BMV}, pk_{BMV}$

**ID:** VIN

---

**Enclave Initialize Request**

**TLS Connect**

**Session Key:** $K$

**{Enclave Program}$K$

**Enclave Install**

**Boot-Up Application**

**Check Enclave Integrity**

---

**Remote Attest SGX Platform**

**{Enclave Approve}$K$**

**TEE Designer Server**

**Key:** $K_{PV}$
The Protocol: Registration Phase (One Time)

Signature Keys: $s_{BMV}$, $p_{BMV}$

Enclave Initialize Request

TLS Connect

Fused Keys: $K_{Seal}$, $K_{PV}$

ID: VIN

Session Key: $K$

{Enclave Program}$K$

Enclave Install

Boot-Up Application

Check Enclave Integrity

Remote Attest SGX Platform

{Enclave Approve}$K$

{VIN, Owner Info}$K$
The Protocol: Registration Phase (One Time)
The Protocol: Registration Phase (One Time)

**Signature Keys:** $sk_{BMV}, pk_{BMV}$

**Enclave Initialize Request**

**TLS Connect**

**Session Key:** $K$

**{Enclave Program}$K$

**Enclave Install**

**Boot-Up Application**

**Check Enclave Integrity**

**Remote Attest SGX Platform**

**{Enclave Approve}$K$

**{VIN, Owner Info}$K$

**{VRK}$K$

**Seal (VRK)**

**VIN Database**

**ADD (VIN, VRK, ...)**
The Protocol: DS Key Renewal (On Demand)
The Protocol: DS Key Renewal (On Demand)

Fused Keys: $K_{Seal}, K_{PV}$
ID: VIN

Signature Keys: $sk_{BMV}, pk_{BMV}$

DS Key Database
ENTRY ($K_{DS}, \text{ActiveGeo, ActiveDate}$)

TEE Designer Server
$K_{PV}$
The Protocol: DS Key Renewal (On Demand)

Fused Keys: $K_{\text{Seal}}, K_{PV}$
ID: VIN

Signature Keys: $sk_{BMV}, pk_{BMV}$

TEE Designer Server

DS Key Database
ENTRY ($K_{DS}, \text{ActiveGeo}, \text{ActiveDate}$)

DS Key Request
The Protocol: DS Key Renewal (On Demand)

Fused Keys: $K_{Seal}, K_{PV}$
ID: VIN

Signature Keys: $sk_{BMV}, pk_{BMV}$

TEE Designer Server
$K_{PV}$

DS Key Database
ENTRY ($K_{DS}, \text{ActiveGeo, ActiveDate}$)

DS Key Request

Attestation: Verify SGX Platform
The Protocol: DS Key Renewal (On Demand)

**Fused Keys:** $K_{Seal}, K_{PV}$

**ID:** VIN

**Signature Keys:** $sk_{BMV}, pk_{BMV}$

**TEE Designer Server**

$K_{PV}$

**DS Key Database**

ENTRY ($K_{DS}$, ActiveGeo, ActiveDate)

**DS Key Request**

**Attestation: Verify SGX Platform**

**Derive Session Key $K_{RA}$**
The Protocol: DS Key Renewal (On Demand)

Fused Keys: $K_{Seal}, K_{PV}$
ID: VIN

Signature Keys: $sk_{BMV}, pk_{BMV}$

TEE Designer Server $K_{PV}$

Attestation: Verify SGX Platform

DS Key Request

Derive Session Key $K_{RA}$

ENTRY ($K_{DS}$, ActiveGeo, ActiveDate)

{$K_{DS}$, VRL, Config}$K_{RA}$
The Protocol: DS Key Renewal (On Demand)

Fused Keys: $K_{Seal}, K_{PV}$

ID: VIN

Signature Keys: $sk_{BMV}, pk_{BMV}$

TEE Designer Server: $K_{PV}$

Derive Session Key $K_{RA}$

Attestation: Verify SGX Platform

DS Key Request

TS ← TrustedTime()
Loc ← TrustedLoc()
Seal ($K_{DS}, TS, Loc, VRL, Config$)

ENTRY ($K_{DS}, ActiveGeo, ActiveDate$)

{ $K_{DS}, VRL, Config$ }$K_{RA}$
The Protocol: V2V (On Demand)
The Protocol: V2V (On Demand)

Vehicle V1

Derive $\text{TID} \leftarrow \text{HKDF}($VRK, RIID$)$

Vehicle V2
The Protocol: V2V (On Demand)

Vehicle V1

Derive $TID \leftarrow HKDF(VRK, RIID)$

Vehicle V2

$\{TS, loc, TID, BSM\}K_{DS}, CMAC$
The Protocol: V2V (On Demand)

Vehicle V1

| Derive | TID ← HKDF(VRK, RIID) |

{TS, loc, TID, BSM}KDS, CMAC

Vehicle V2

Verify CMAC
Decrypt TS, loc, TID, BSM
Verify TS with error tolerance
Verify Loc with error tolerance
Check TID =? HKDF(VRK_{VRL}, RIID)
Process BSM
Preliminary Results: TEE vs. PKI

(a) Key Exchange Latency

(b) BSM Broadcast Latency

(c) End To End Latency
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

- 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 [S&P22]
vSGX: Virtualizing SGX Enclaves on AMD SEV [S&P22]

vSGX works like an SGX module plugged onto an SEV machine

- Binary compatibility
- Comparable security guarantee with both SGX and SEV
- Reasonable performance
vSGX: Virtualizing SGX Enclaves on AMD SEV [S&P22]

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

The source code has been made available at https://github.com/OSUSecLab/vSGX
A TEE-based V2V Protocol [AutoSec’22]

Objectives

1. **(Instant) Authentication**
2. Message Integrity (Tamper Resiliency)
3. Message Confidentiality (Privacy)
4. **Anti-tracking** (Privacy)
5. Revocation and **Accountability**
6. Scalability

https://github.com/OSUSecLab/tee-v2v
A TEE-based V2V Protocol [AutoSec’22]

Objectives

1. (Instant) Authentication
2. Message Integrity (Tamper Resiliency)
3. Message Confidentiality (Privacy)
4. Anti-tracking (Privacy)
5. Revocation and Accountability
6. Scalability
A TEE-based V2V Protocol [AutoSec’22]

Our Approach

- Leveraging TEEs to protect secrets
- Using a shared Daily Symmetric (DS) Key among vehicles
- Storing DS key in the enclave
- Using Remote Attestation to ensure the right environment
- Generating temporal IDs for privacy
Our Approach

- Leveraging TEEs to protect secrets
- Using a shared Daily Symmetric (DS) Key among vehicles
- Storing DS key in the enclave
- Using Remote Attestation to ensure the right environment
- Generating temporal IDs for privacy

https://github.com/OSUSecLab/tee-v2v
Towards Distributed and Virtualized Trusted Execution Environments

Zhiqiang Lin
zlin@cse.ohio-state.edu

07/14/2022
References I


References


