Legal Disclaimers

• The comments and statements are the presenter’s and not necessarily Intel’s

• Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

• No computer system can be absolutely secure.
Principles of Operation
The Basic Issue: Vulnerabilities in Privileged Code

Protected Mode (rings) protects OS from apps ...
The Basic Issue: Vulnerabilities in Privileged Code

Protected Mode (rings) protects OS from apps ...

... and apps from each other ...
The Basic Issue: Vulnerabilities in Privileged Code

Protected Mode (rings) protects OS from apps ...

... and apps from each other ...

... UNTIL a malicious app exploits a flaw to gain full privileges and then tampers with the OS or other apps

Apps not protected from privileged code attacks
The Basic Issue: Vulnerabilities in Privileged Code

Protected Mode (rings) protects OS from apps ... 

... and apps from each other ...

... UNTIL a malicious app exploits a flaw to gain full privileges and then tampers with the OS or other apps

Apps not protected from privileged code attacks
Reduced attack surface with SGX
Reduced attack surface with SGX

Application gains ability to defend its own secrets
Reduced attack surface with SGX

Application gains ability to defend its own secrets
- Smallest attack surface (app + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets
Reduced attack surface with SGX

Application gains ability to defend its own secrets
- Smallest attack surface (app + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

Familiar development/debug
Reduced attack surface with SGX

Application gains ability to defend its own secrets
- Smallest attack surface (app + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

Familiar development/debug
- Single application environment
Reduced attack surface with SGX

Application gains ability to defend its own secrets
- Smallest attack surface (app + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

Familiar development/debug
- Single application environment
- Build on existing ecosystem expertise
Reduced attack surface with SGX

Application gains ability to defend its own secrets
- Smallest attack surface (app + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

Familiar development/debug
- Single application environment
- Build on existing ecosystem expertise

Familiar deployment model
Reduced attack surface with SGX

Application gains ability to defend its own secrets
- Smallest attack surface (app + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

Familiar development/debug
- Single application environment
- Build on existing ecosystem expertise

Familiar deployment model
- Platform integration not a bottleneck to deployment of trusted apps

Scalable security within mainstream environment
What is an Enclave?

User Process

- OS
- App Data
- App Code
What is an Enclave?

Enclave: trusted execution environment embedded in a process

Diagram showing:
- OS
- Enclave
- App Data
- App Code
- User Process
What is an Enclave?

Enclave: trusted execution environment embedded in a process

- Own code and data
- Provides Confidentiality
- Provides integrity
- Controlled entry points
What is an Enclave?

Enclave: trusted execution environment embedded in a process

- Own code and data
- Provides Confidentiality
- Provides integrity
- Controlled entry points
- Supporting multiple threads
What is an Enclave?

Enclave: trusted execution environment embedded in a process

- Own code and data
- Provides Confidentiality
- Provides integrity
- Controlled entry points
- Supporting multiple threads
- Full access to app memory
How SGX Works: Protection vs. Software Attack

Application

Privileged System Code: OS, VMM, BIOS, SMM, ...
How SGX Works: Protection vs. Software Attack

1. App is built with trusted and untrusted parts.

Application

Untrusted Part of App

Trusted Part of App

Privileged System Code:
OS, VMM, BIOS, SMM, ...
How SGX Works: Protection vs. Software Attack

1. App is built with trusted and untrusted parts
2. App runs & creates enclave which is placed in trusted memory
How SGX Works: Protection vs. Software Attack

Application

Untrusted Part of App

Create Enclave

Call Trusted Func.

Trusted Part of App

Call Gate

Execute

SSN: 999-84-2611

1. App is built with trusted and untrusted parts
2. App runs & creates enclave which is placed in trusted memory
3. Trusted function is called; code running inside enclave sees data in clear; external access to data is denied

Privileged System Code:
OS, VMM, BIOS, SMM, ...
How SGX Works: Protection vs. Software Attack

Application

1. App is built with trusted and untrusted parts
2. App runs & creates enclave which is placed in trusted memory
3. Trusted function is called; code running inside enclave sees data in clear; external access to data is denied
4. Function returns; enclave data remains in trusted memory
Enclave Life Cycle – Overview

Physical Address Space
Enclave Life Cycle – Overview

Physical Address Space

- System Memory
- Enclave
- Page
- Cache

EPCM
- Invalid
- Invalid
- Invalid
- Invalid
- Invalid
Enclave Life Cycle – Overview

Virtual Address Space

ECREATE (Range)

Physical Address Space

System Memory

Enclave

Page Cache

Invalid

Invalid

Invalid

Invalid

SECS

Valid, SECS, Range

EPCM

Enclave creation
Enclave Life Cycle – Overview

Virtual Address Space
- Enclave
  - Code/Data
  - ECREATE (Range)

Physical Address Space
- System Memory
  - Enclave
  - Page Cache
  - Plaintext Code/Data
  - EPCM
    - Valid, SECS, Range
    - Invalid
    - Invalid
    - Invalid
    - Invalid

Enclave creation
Enclave Life Cycle – Overview

Virtual Address Space

- Enclave
  - Code/Data

Physical Address Space

- System Memory
  - Enclave
    - SECS
  - Page Cache
  - Plaintext Code/Data
  - ECREATE (Range)
  - EADD (Copy Page)

EPCM

- Valid, SECS, Range
- Invalid
- Invalid
- Valid, REG, LA, SECS
- Invalid
Enclave Life Cycle – Overview

Virtual Address Space

- Code/Data

Physical Address Space

- System Memory
  - Enclave
    - Page
      - SECS
    - Cache
      - Plaintext
      - Code/Data

EPCM

- Valid, SECS, Range
- Invalid
- Invalid
- Valid, REG, LA, SECS
- Invalid
Enclave Life Cycle – Overview

Virtual Address Space

Physical Address Space

Enclave creation

ECREATE (Range)
EADD (Copy Page)
EEXTEND
Enclave Life Cycle – Overview

Virtual Address Space

- Code/Data
- Code/Data

Physical Address Space

- System Memory
- Enclave
  - E CREATE (Range)
  - E ADD (Copy Page)
  - E EXTEND
  - E INIT

- Enclave
  - SECS
  - Plaintext
  - Code/Data

EPCM

- Invalid
- Valid, SECS, Range
- Valid, REG, LA, SECS
- Valid, REG, LA, SECS

Enclave init
Enclave Life Cycle – Overview

Virtual Address Space

Enclave

Code/Data

Code/Data

Physical Address Space

System Memory

Enclave

SECS

Plaintext

Code/Data

Plaintext

Code/Data

EPCM

Valid, SECS, Range

Invalid

Valid, REG, LA, SECS

Valid, REG, LA, SECS

Invalid

Enclave active

ECREATE (Range)
EADD (Copy Page)
EEXTEND
EINIT
EENTER

10 /15
Enclave Life Cycle – Overview

Virtual Address Space

Physical Address Space

Enclave active

Enclave

Code/Data

Code/Data

System Memory

Enclave

Page

Cache

ECREATE (Range)
EADD (Copy Page)
EEXTEND
EINIT
EENTER

ECMD

Virtual Address Space

Physical Address Space

Enclave active

Enclave

Code/Data

Code/Data

System Memory

Enclave

Page

Cache

ECREATE (Range)
EADD (Copy Page)
EEXTEND
EINIT
EENTER

EPCM

Valid, SECS, Range
Invalid
Valid, REG, LA, SECS
Valid, REG, LA, SECS
Invalid
Enclave Life Cycle – Overview

Virtual Address Space

- Code/Data
- Code/Data

- Enclave

Physical Address Space

- System Memory
- Enclave
- Page
- Cache
- SECS
- Plaintext
- Code/Data
- Plaintext
- Code/Data

EPCM

- Valid, SECS, Range
- Invalid
- Valid, REG, LA, SECS
- Valid, REG, LA, SECS
- Invalid
Enclave Life Cycle – Overview

Virtual Address Space

Physical Address Space

Enclave active

Enclave

Code/Data

System Memory

Code/Data

Enclave

Code/Data

ECREATE (Range)

ECADD (Copy Page)

EEXTEND

EINIT

EENTER

EEXIT

ECACHE

Plaintext

Code/Data

Plaintext

Code/Data

EPCM

Invalid

Valid, SECS, Range

Valid, REG, LA, → SECS

Valid, REG, LA, → SECS

Invalid

Enclave active
Enclave Life Cycle – Overview

Virtual Address Space

Physical Address Space

Enclave destruction

ECREATE (Range)
EADD (Copy Page)
EEXTEND
EINIT
ENTER
EXIT
EREMOVE

System Memory

Enclave

Page Cache

SECS

EPCM

Valid, SECS, Range
Invalid
Invalid
Invalid
Invalid
Enclave Life Cycle – Overview

Virtual Address Space

Physical Address Space

ECREATE (Range)
EADD (Copy Page)
EEXTEND
EINIT
EENTER
EEXIT
EREMOVE

System Memory

Enclave Page
Cache

EPCM
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid

Enclave destruction
Enclave Life Cycle – Instantiating an Enclave (1)

ECREATE

- Creates a unique instance of an enclave, establishes the linear address range, and sets enclave’s attributes
  - Enclave mode of operation (32/64)
  - Processor features that enclave supports
  - Debug is allowed or not

Information is stored within a Secure Enclaves Control Structure (SECS) generated by ECREATE.
Enclave Life Cycle – Instantiating an Enclave (1)

ECREATE

• Creates a unique instance of an enclave, establishes the linear address range, and sets enclave’s attributes
  – Enclave mode of operation (32/64)
  – Processor features that enclave supports
  – Debug is allowed or not
  Information is stored within an Secure Enclaves Control Structure (SECS) generated by ECREATE.

EADD

• Add Regular (REG) or Thread Control Structure (TCS) pages into the enclave
  – System software is responsible for selecting free EPC page, type, and attributes, content of the page and the enclave to which the page added to.
  – EADD sets EPCM entry to indicate type of page (REG, TCS), linear address, RWX, parent enclave etc.
Enclave Life Cycle – Instantiating an Enclave (2)

**EEXTEND**

- Generates a cryptographic hash of the content of the enclave in 256B chunks
Enclave Life Cycle – Instantiating an Enclave (2)

**EEXTEND**

- Generates a cryptographic hash of the content of the enclave in 256B chunks

**EINIT**

- Verifies the enclave’s content against the ISV’s signed SIGSTRUCT and initializes the enclave – mark it ready to be used
  - Validate SIGSTRUCT is signed using SIGSTRUCT public key
  - Enclave measurement matches the measurement specified in SIGSTRUCT
  - Enclave attributes compatible with SIGSTRUCT
  - Record sealing identity (sealing authority, product id, SVN) in the SECS
Enclave Flow Control – EENTER

- Checks that TCS is not busy

- Saves RSP/RBP for later restore on enclave Asynchronous Exit

- Changes the mode of operation to be in enclave mode
  - Flush TLB entries for enclave linear address range
  - Save CPU feature set (XCR0), load enclave’s XFRM value
  - Transfer control from outside enclave to pre-determined location inside the enclave

- Debug
  - If debug enclave and software wishes to debug then allow traps, breakpoints, single step
  - Else, set HW so the enclave appears as a single instruction
Enclave Flow Control – EEXIT

• Clears enclave mode and TLB entries for enclave addresses
• Restores CPU feature set (XCR0)
• Marks TCS as not busy
• Transfers control from inside enclave to a location outside enclave (specified by enclave code as a parameter)
• Registers state is not cleared; scrubbing of secrets is the responsibility of enclave code (run time system)
Handling Exceptions – Asynchronous Exit (AEX)

- Initiated by faults, exceptions and interrupts
- Saves registers state inside the enclave, in a pre-allocated State Save Area (SSA), and sets registers to a synthetic state
- Transfers control to a pre-set OS exception handler location
  - Before that, sets the exception handler stack so IRET will return to a pre-defined location (trampoline code) in the app code.
Handling Exceptions – ERESUME

• Called by the app’s trampoline code

• Very similar to EENTER, except:
  • Restores register state from the State Save Area (SSA) area
  • Resumes enclave operation at the RIP stored in SSA on last AEX
Hardware Operation – SGX Access Control

Linear Address → Traditional IA Page Table Checks → Physical Address
Hardware Operation – SGX Access Control

Linear Address → Traditional IA Page Table Checks → Physical Address → Enclave Access?
Hardware Operation – SGX Access Control

1. Linear Address → Traditional IA Page Table Checks → Physical Address → Enclave Access?

   - No: Non-Enclave Access
     - Address in EPC?
       - Yes: Replace Address With Abort Page
         - No: Allow Memory Access
       - No: Non-Enclave Access

   - Yes: Allow Memory Access
Hardware Operation – SGX Access Control

1. Linear Address
2. Traditional IA Page Table Checks
3. Enclave Access?
   - Yes
   - Enclave Access
   - No
   - Physical Address
4. Address in EPC?
   - Yes
   - Check EPCM
   - No
   - Replace Address With Abort Page
   - Allow Memory Access
5. Non-Enclave Access
   - Address in EPC?
     - Yes
     - Allow Memory Access
     - No
Hardware Operation – SGX Access Control

Linear Address ➔ Traditional IA Page Table Checks ➔ Physical Address ➔ Enclave Access?

Enclave Access

- Is the page valid?
- Is the page type correct?
- Does page belong to current enclave?
- Do RWX attributes match?
- Is linear address correct?

Yes ➔ Address in EPC?

Yes ➔ Check EPCM

No ➔ Non-Enclave Access

Address in EPC?

Yes ➔ Replace Address With Abort Page

No ➔ Allow Memory Access
Hardware Operation – SGX Access Control

Enclave Access

1. Traditional IA Page Table Checks
2. Enclave Access?
3. Address in EPC?

Non-Enclave Access

1. Address in EPC?
2. Replace Address With Abort Page
3. Allow Memory Access

- Is the page valid?
- Is the page type correct?
- Does page belong to current enclave?
- Do RWX attributes match?
- Is linear address correct?
Hardware Operation – SGX Access Control

Linear Address → Traditional IA Page Table Checks → Physical Address → Enclave Access?

Yes → Address in EPC?

Yes → Replace Address With Abort Page

No → Check EPCM

Yes → Checks Pass? → Allow Memory Access

No → Signal Fault

No → Enclave Access

No → Non-Enclave Access

Yes → Address in EPC?

Yes → Allow Memory Access

No → Replace Address With Abort Page
Hardware Operation – EPC Memory Encryption

CPU Package

- Cores
- Cache

System Memory

Non-Enclave Access
Hardware Operation – EPC Memory Encryption

CPU Package

- Cores
- Cache

System Memory

Non-Enclave Access

- Security perimeter is the CPU package boundary
Hardware Operation – EPC Memory Encryption

CPU Package

- Cores
- AMEX: 3234-134584-26864

System Memory

Non-Enclave Access
- Security perimeter is the CPU package boundary
- Data and code unencrypted inside CPU package
Hardware Operation – EPC Memory Encryption

- Security perimeter is the CPU package boundary
- Data and code unencrypted inside CPU package
- Data and code outside CPU package is encrypted and/or integrity checked

Non-Enclave Access

- Security perimeter is the CPU package boundary
- Data and code unencrypted inside CPU package
- Data and code outside CPU package is encrypted and/or integrity checked
Hardware Operation – EPC Memory Encryption

- Security perimeter is the CPU package boundary
- Data and code unencrypted inside CPU package
- Data and code outside CPU package is encrypted and/or integrity checked
- External memory reads and bus snoops see only encrypted data
SGX High-level HW/SW Picture

Application Environment (Ring 3)

Privileged Environment (Ring 0)

Exposed Hardware

Enclave
- SGX Trusted Runtime
- SGX Untrusted User Runtime

Enclave
- SGX Trusted Runtime
- SGX Untrusted User Runtime

SGX Driver

Protected Memory

Page Tables

EPC

EPCM

Ring 3 Instructions
- EEXIT
- EGETKEY
- EREPORT
- EENTER
- ERESUME

Ring 0 Instructions
- ECREATE
- ETRACK
- EADD
- EWBI
- EEXTEND
- ELD
- EINIT
- EPA
- EBLOCK
- EREMOVE

H/w Data Structure
- Hardware
- Runtime
- Application
- OS Data Structure
Attestation and Sealing
The Challenge: Provisioning Secrets to the Enclave

• An enclave is in the clear before instantiation
  – Sections of code and data could be encrypted, but their decryption key can’t be pre-installed

• Secrets come from outside the enclave
  – Keys
  – Passwords
  – Sensitive data

• The enclave must be able to convince a 3\textsuperscript{rd} party that it’s trustworthy and can be provisioned with the secrets

• Subsequent runs should be able to use the secrets that have already been provisioned
Critical Features: Attestation and Sealing

- App executes on local platform
Critical Features: Attestation and Sealing

- App executes on local platform
- HW based **Attestation** provides remote platform assurance that “this is the right app executing in the right platform”
Critical Features: Attestation and Sealing

- App executes on local platform

- HW based **Attestation** provides remote platform assurance that “this is the right app executing in the right platform“
  - $\Rightarrow$ Remote platform can provision local platform with secrets
Critical Features: Attestation and Sealing

- App executes on local platform
- HW based **Attestation** provides remote platform assurance that “this is the right app executing in the right platform“
  - Remote platform can provision local platform with secrets
- App can seal secrets to platform for future use
Trustworthiness

- Intel® SGX provides the means for an enclave to securely prove to a 3rd party:
  - What software is running inside the enclave
  - Which execution environment the enclave is running at
  - Which Sealing Identity will be used by the enclave
  - What’s the CPU’s security level

- Accomplished by:
  - A combination of SGX instructions (local attestation)
  - Intel provided attestation enclave (remote attestation)

- Once provisioned a secret, an enclave can seal its data using SGX instructions
Enclave Measurement (Software TCB)

• When building an enclave, Intel® SGX generates a cryptographic log of all the build activities
  - Content: Code, Data, Stack, Heap
  - Location of each page within the enclave
  - Security flags being used

• MRENCLAVE ("Enclave Identity") is a 256-bit digest of the log
  - Represents the enclave’s software TCB

• A software TCB verifier should:
  - Securely obtain the enclave’s software TCB
  - Securely obtain the expected enclave’s software TCB
  - Compare the two values
Local Attestation

- “Local attestation”: The process by which one enclave attests its TCB to another enclave on the same platform
- Using Intel® SGX’s EREPORT and EGETKEY instructions
  - EREPORT generates a cryptographic REPORT that binds MRENCLAVE to the target enclave’s REPORT KEY
  - EGETKEY provides the REPORT KEY to verify the REPORT

<table>
<thead>
<tr>
<th>TCB component</th>
<th>Attestation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Firmware &amp; hardware</td>
<td>CPU REPORT KEY</td>
</tr>
<tr>
<td>Software</td>
<td>MRENCLAVE</td>
</tr>
</tbody>
</table>
Local Attestation - Flow
1. Verifying enclave sends its MRENCLAVE to reporting enclave
1. Verifying enclave sends its MRENCLAVE to reporting enclave
Local Attestation - Flow

1. Verifying enclave sends its MRENCLAVE to reporting enclave
2. Reporting enclave creates a cryptographic REPORT that includes its MRENCLAVE
1. Verifying enclave sends its MRENCLAVE to reporting enclave
2. Reporting enclave creates a cryptographic REPORT that includes its MRENCLAVE
3. Verifying enclave obtains its REPORT key and verifies the authenticity of the REPORT
Local Attestation - Flow

1. Verifying enclave sends its MRENCLAVE to reporting enclave
2. Reporting enclave creates a cryptographic REPORT that includes its MRENCLAVE
3. Verifying enclave obtains its REPORT key and verifies the authenticity of the REPORT
Remote Attestation

• “Remote attestation”: The process by which one enclave attests its TCB to another entity outside of the platform

• Intel® SGX Extends Local attestation by allowing a Quoting Enclave (QE) to use Intel® EPID to create a QUOTE out of a REPORT
  - Intel® EPID is a group signature scheme

<table>
<thead>
<tr>
<th>TCB component</th>
<th>Attestation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Firmware &amp; hardware</td>
<td>Asymmetric - Intel® EPID</td>
</tr>
<tr>
<td>Software</td>
<td>MRENCLAVE</td>
</tr>
</tbody>
</table>
Remote Attestation - Flow

1. Verifying enclave becomes the Quoting Enclave (QE).
2. After verifying the REPORT the, QE signs the REPORT with the EPIp private key and converts it into a QUOTE.
3. Remote platform verifies the QUOTE with the EPIp public key and verifies MRENCLAVE against the expected value.
Remote Attestation - Flow

1. Verifying enclave becomes the Quoting Enclave (QE).
2. After verifying the REPORT the, QE signs the REPORT with the EPID private key and converts it into a QUOTE.
3. Remote platform verifies the QUOTE with the EPID public key and verifies MRENCLAVE against the expected value.
1. Verifying enclave becomes the Quoting Enclave (QE).
2. After verifying the REPORT the, QE signs the REPORT with the EPID private key and converts it into a QUOTE
3. Remote platform verifies the QUOTE with the EPID public key and verifies MRENCLAVE against the expected value
1. Verifying enclave becomes the Quoting Enclave (QE).
2. After verifying the REPORT the, QE signs the REPORT with the EPID private key and converts it into a QUOTE.
3. Remote platform verifies the QUOTE with the EPID public key and verifies MRENCLAVE against the expected value.
1. Verifying enclave becomes the Quoting Enclave (QE).
2. After verifying the REPORT the, QE signs the REPORT with the EPID private key and converts it into a QUOTE.
3. Remote platform verifies the QUOTE with the EPID public key and verifies MRENCLAVE against the expected value.
Provisioning SGX with HW-based Keys

Intel generates and fuses a unique key during manufacturing. Intel maintains a database of these keys.

Intel generates provisioning blobs for each device specific to the device TCB & Provisioning Enclave.

Protocol between Provisioning Enclave & Provisioning Service:
• Platform proves it has a key that Intel put in a real processor.
• Server certifies an Attestation Key for the platform.
Intel® EPID

Intel® Enhanced Privacy Identifier (EPID) key is used for platform identity.

EPID is a Group based anonymous signature scheme.

EPID requires 3 keys:
- Master Issuing Key
- Group Public Key
- Member Private Key

Member private keys will be generated on platform.

Public keys and revocation lists to need to be issued to EPID verifiers.
Sealing

• “Sealing”: Cryptographically protecting data when it leaves the enclave.

• Enclaves use EGETKEY to retrieve an enclave, platform persistent key and encrypts the data

• EGETKEY uses a combination of enclave attributes and platform unique key to generate keys
  – Enclave Sealing Authority
  – Enclave Product ID
  – Enclave Product Security Version Number (SVN)
Sealing Authority

- Every enclave has an Enclave Certificate (SIGSTRUCT) which is signed by a Sealing Authority (typically the enclave writer)
- SIGSTRUCT includes:
  - Enclave’s Identity (represented by MRENCLAVE)
  - Sealing Authority’s public key (represented by MRSIGNER)
- \textit{EINIT} instruction verifies the signature over SIGSTRUCT prior to enclave initialization
Example: Secure Transaction

1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls EREPORT to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls EGETKEY to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
Example: Secure Transaction

1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls EREPORT to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls EGETKEY to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls *EGETKEY* to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
Example: Secure Transaction

1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls $EREPORT$ to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls $EGETKEY$ to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls **EREPORT** to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls **EGETKEY** to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls *EREPORT* to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls *EGETKEY* to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
Example: Secure Transaction

1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls EREPORT to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls EGETKEY to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
Example: Secure Transaction

1. Enclave built & measured against ISV’s signed certificate
2. Enclave calls EREPORT to obtain a REPORT that includes enclave specific data (ephemeral key)
3. REPORT & user data sent to Quoting Enclave who signs the REPORT with an EPID private key
4. QUOTE sent to server & verified
5. Ephemeral key used to create a trusted channel between enclave and remote server
6. Secret provisioned to enclave
7. Enclave calls EGETKEY to obtain the SEAL KEY
8. Secret is encrypted using SEAL KEY & stored for future use
Links


• Joint research poster session: http://sigops.org/sosp/sosp13/


• HASP Workshop: https://sites.google.com/site/haspworkshop2013/workshop-program

• ISCA 2015 Tutorial: http://sgxisca.weebly.com/
Thank You
Backup
SGX Paging Introduction

Requirement:

• Remove an EPC page and place into unprotected memory. Later restore it.
• Page must maintain same security properties (confidentiality, anti-replay, and integrity) when restored

Instructions:

• EWB: Evict EPC page to main memory with cryptographic protections
• ELDB/ELDU: Load page from main memory to EPC with cryptographic protections
• EPA: Allocate an EPC page for holding versions
• EBLOCK: Declare an EPC page ready for eviction
• ETRACK: Ensure address translations have been cleared
Page-out Example

EPC

- SECS
- Enclave Page
- VA Page
- Enclave Page
- Enclave Page

EWB

System Memory

39
Page-out Example

EWB Parameters:
- Pointer to EPC page that needs to be paged out
- Pointer to empty version slot
- Pointers outside EPC location
Page-out Example

EWB Parameters:
- Pointer to EPC page that needs to be paged out
- Pointer to empty version slot
- Pointers outside EPC location

EWB Operation
- Remove page from the EPC
- Populate version slot
- Write encrypted version to outside
- Write meta-data, PCMD
EWB Parameters:

- Pointer to EPC page that needs to be paged out
- Pointer to empty version slot
- Pointers outside EPC location

EWB Operation

- Remove page from the EPC
- Populate version slot
- Write encrypted version to outside
- Write meta-data, PCMD

All pages, including SECS and Version Array can be paged out
Page-in Example

EPC

SECS

Enclave Page

Free Enclave Page

Enclave Page

ELD

System Memory

Encrypted Page

MD

BUILD
Page-in Example

ELD Parameters:
- Encrypted page
- Free EPC page
- SECS (for an enclave page)
- Populated version slot
Page-in Example

ELD Parameters:
- Encrypted page
- Free EPC page
- SECS (for an enclave page)
- Populated version slot

ELD Operation:
- Verify and decrypt the page using version
- Populate the EPC slot
- Make back-pointer connection (if applicable)
- Free-up version slot