Secure Processors in Industry

Mengjia Yan
Fall 2020

Based on slides from Christopher W. Fletcher and Jakub Szefer
Reminder

• Fill the google form
  • [https://forms.gle/G6gh6sEYJ4UY24ePA](https://forms.gle/G6gh6sEYJ4UY24ePA)

• First review will be due @ 09/27 (2.5 weeks from now)
Recommended Reading

• *Intel SGX Explained; Victor Costan, Srini Devadas*
  • Great refresh on computer architecture
  • Background on cryptographic
  • Basic SGX programming model and architecture support (next lecture)
Outline

• IBM secure coprocessor 3848 and follow-ons
• Trusted Platform Module (TPM)
• Intel TXT, AMD
• Arm TrustZone
• Intel SGX
• AMD SEV

Threat model
Trusted Computing Base (TCB)
Physical Attacks
Computing Model

Processor Chip (socket)

- core L1/L2
- core L1/L2
- ... L1/L2
- LLC

Processor Chip (socket)

- core L1/L2
- core L1/L2
- ... L1/L2
- LLC

Memory (DRAM)

Non-volatile storage device

other I/O Devices

System Bus (logically)
Computing Model

Processor Chip (socket)
- core L1/L2
- core L1/L2
- LLC

Processor Chip (socket)
- core L1/L2
- core L1/L2
- LLC

Memory (DRAM)
Non-volatile storage device
other I/O Devices

ASIC/Co-processor
System Bus (logically)

6.888 L2 - Secure Processors in Industry
Hardware Adversary

• Pre-fab adversary (HW trojans)

• Physical attacks
  • Generally require physical access
  • Classified according to cost
  • A cold boot attack example
Hardware Adversary

• Pre-fab adversary (HW trojans)

• Physical attacks
  • Generally require physical access
  • Classified according to cost
  • A cold boot attack example

Advanced Hardware Hacking Techniques; Joe Grand; DEFCON’12
Hardware Adversary

• Pre-fab adversary (HW trojans)

• Physical attacks
  • Generally require physical access
  • Classified according to cost
  • A cold boot attack example

Advanced Hardware Hacking Techniques; Joe Grand; DEFCON’12
Hardware Adversary

• Pre-fab adversary (HW trojans)

• Physical attacks
  • Generally require physical access
  • Classified according to cost
  • A cold boot attack example

Advanced Hardware Hacking Techniques; Joe Grand; DEFCON’12
Hardware Adversary

• Pre-fab adversary (HW trojans)

• Physical attacks
  • Generally require physical access
  • Classified according to cost
  • A cold boot attack example

*Advanced Hardware Hacking Techniques; Joe Grand; DEFCON'12*
A Cold Boot Attack Example

https://www.youtube.com/watch?v=vWHDqBV9yGc
A Cold Boot Attack Example

Frozen RAM retains contents for a short period

https://www.youtube.com/watch?v=vWHDqBV9yGc
A Cold Boot Attack Example

https://www.youtube.com/watch?v=vWHDqBV9yGc

Gutmann et al. “Data Remanence in Semiconductor Devices”
More Physical Attack Examples

Tap board used to intercept data transfer over Xbox's HyperTransport bus from http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html
More Physical Attack Examples

Tap board used to intercept data transfer over Xbox's HyperTransport bus from http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html

Physical Tamper Resistance

• Standalone security modules to protect cryptographic keys and personal identification numbers (PINs)
• A history lesson of physical security by IBM 4758
Physical Tamper Resistance

• Standalone security modules to protect cryptographic keys and personal identification numbers (PINs)
• A history lesson of physical security by IBM 4758

Tampering Detection

Robust metal enclosures.
Open the lid → disconnect power supply

Drill through the lid

Photocells and tilt devices
Physical Tamper Resistance

- Standalone security modules to protect cryptographic keys and personal identification numbers (PINs)
- A history lesson of physical security by IBM 4758

**Tampering Detection**
- Robust metal enclosures.
  - Open the lid → disconnect power supply
- Drill through the lid
- Photocells and tilt devices

**Tampering Evident**
- “Potting” the device in a block of epoxy resin
- Patience: Scrape away the epoxy
- Tamper-sensing barriers: nichrome wire wound around the device
IBM 4758 Secure Co-Processor

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html
IBM 4758 Secure Co-Processor

- Memory remanence
  - constant movement of values from place to place
- Cold boot
  - detects changes of temperature
- X-ray
  - a radiation sensor
- Power side channels
  - Solid aluminium shielding and a low-pass filter (a Faraday cage)

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html
IBM 4758 Secure Co-Processor

- Memory remanence
  - constant movement of values from place to place
- Cold boot
  - detects changes of temperature
- X-ray
  - a radiation sensor
- Power side channels
  - Solid aluminium shielding and a low-pass filter (a Faraday cage)

Expensive. Other secure processors only focus on a limited set of physical attacks.

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html
IBM 4758 and Follow-ons

- The first FIPS 140-1 Level 4 validation, arguably the only general-purpose computational platform validated at this level by 2001

*From Dyer et al. “Building the IBM 4758 Secure Coprocessor”*
IBM 4758 and Follow-ons

- The first FIPS 140-1 Level 4 validation, arguably the only general-purpose computational platform validated at this level by 2001
- A multipurpose programmable device
- Secure Boot and SW attacks (discussed later)

From Dyer et al. “Building the IBM 4758 Secure Coprocessor”

Bond et al. “API-Level Attacks on Embedded Systems.”
Trusted Platform Module (TPM)

- “Commoditized IBM 4758”
- Standard LPC interface – attaches to commodity motherboards
- Weaker computation capability
Trusted Platform Module (TPM)

• “Commoditized IBM 4758”
• Standard LPC interface – attaches to commodity motherboards
• Weaker computation capability

• Uses:
  • Verify platform integrity (firmware+OS)
  • Disk encryption and password protection
Software Attacks
Software Stack

User application

Host operating system/Hypervisor

Hardware
Software Stack

Intel’s Privilege Level

<table>
<thead>
<tr>
<th>Ring</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring 3</td>
<td>Application Enclave application</td>
</tr>
<tr>
<td>Ring 2</td>
<td></td>
</tr>
<tr>
<td>Ring 1</td>
<td></td>
</tr>
<tr>
<td>Ring 0</td>
<td>OS kernel</td>
</tr>
<tr>
<td>SMM</td>
<td>BIOS/firmware</td>
</tr>
</tbody>
</table>

Less Privilege

More Privilege

SMM: system management mode
Software Stack

Intel’s Privilege Level

<table>
<thead>
<tr>
<th>Ring</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring 3</td>
<td>Application/Enclave application</td>
</tr>
<tr>
<td>Ring 2</td>
<td></td>
</tr>
<tr>
<td>Ring 1</td>
<td></td>
</tr>
<tr>
<td>Ring 0</td>
<td>OS kernel</td>
</tr>
<tr>
<td>SMM</td>
<td>BIOS/firmware</td>
</tr>
</tbody>
</table>

Less Privilege

More Privilege

SMM: system management mode
Process Isolation When Sharing Hardware

• Share HW resources in SMT contexts, same processor chips, across sockets.
Virtual Address Abstraction

6.888 L2 - Secure Processors in Industry
Virtual Address Abstraction

Benefits of virtual memory abstraction:
• Over-commit memory: the illusion that they own all resources
• Security: process isolation
• Programmability: software independent of DRAM size
Virtual Address Abstraction

Benefits of virtual memory abstraction:
• Over-commit memory: the illusion that they own all resources
• Security: process isolation
• Programmability: software independent of DRAM size
Page Table

- Page table:
  - A data structure to store address translation entries
  - Multi-level trees
Page Table

• Page table:
  • A data structure to store address translation entries
  • Multi-level trees

• Page table entry attributes:
  • Writable (W), Executable (X), Supervisor (S), etc.
  • E.g., data execution prevention (DEP)
Page Table

• Page table:
  • A data structure to store address translation entries
  • Multi-level trees

• Page table entry attributes:
  • Writable (W), Executable (X), Supervisor (S), etc.
  • E.g., data execution prevention (DEP)

• MMU (memory management unit)
  • A hardware unit performs address translation

• TLB:
  • Caches for page tables
Trusted computing base (TCB)

• Trusted computing base (TCB)
  • TCB is trusted to be correctly implemented
  • Vulnerabilities or attacks on TCB nullify TEE protections
  • TCB may not be trustworthy
Trusted computing base (TCB)

- Trusted computing base (TCB)
  - TCB is trusted to be correctly implemented
  - Vulnerabilities or attacks on TCB nullify TEE protections
  - TCB may not be trustworthy

- Attacks, e.g., Rootkit, may change the **integrity** of TCB

- How to verify platform (HW + low-level SW) integrity
Platform Initialization (Booting)

- Processor Chip (socket)
  - core L1/L2
  - core L1/L2
  - LLC

- System Bus (logically)
- Memory (DRAM)
- Non-volatile storage device
- other I/O Devices

- Processor Chip (socket)
  - core L1/L2
  - core L1/L2
  - LLC

- ME (management engine)

6.888 L2 - Secure Processors in Industry
Platform Initialization (Booting)

For remote system management. Also manage booting.

6.888 L2 - Secure Processors in Industry
Boot Process (UEFI)

1. Security (SEC)
2. Pre-EFI Initialization (PEI)
3. Driver eXecution Environment (DXE)
4. Boot Device Selection (BDS)
5. Transient System Load (TSL)
6. Run Time (RT)

- Security (SEC) measures:
- Pre-EFI Initialization (PEI) microcode firmware
- Driver eXecution Environment (DXE) DRAM Initialized
- Boot Device Selection (BDS) bootloader
- Transient System Load (TSL) OS
- Run Time (RT) Cache-as-RAM
Boot Process (UEFI)

- Security (SEC)
- Pre- EFI Initialization (PEI)
- Driver eXecution Environment (DXE)
- Boot Device Selection (BDS)
- Transient System Load (TSL)
- Run Time (RT)

Root of trust

Cache-as-RAM

microcode
firmware

DRAM Initialized

bootloader

OS
Boot Process (UEFI)

Root of trust

How to perform the measurement?
Cryptographic Hashing (e.g., SHA 1-3)

Hash(m) = h

- Message (long)
- Hash value (length depends on algorithm)

Use as fingerprints
Cryptographic Hashing (e.g., SHA 1-3)

Hash(m) = h

- One-way hash
  - Practically infeasible to invert, Difficult to find collision

Use as fingerprints
Cryptographic Hashing (e.g., SHA 1-3)

- One-way hash
  - Practically infeasible to invert, Difficult to find collision
- Avalanche effect
  - “Bob Smith got an A+ in ELE386 in Spring 2005” → 01eace851b72386c46
  - “Bob Smith got an B+ in ELE386 in Spring 2005” → 936f8991c111f2cefw

Use as fingerprints
Secure Boot using TPM

• Static root of trust for measurement (SRTM)

PCR: platform configuration register
Secure Boot using TPM

- Static root of trust for measurement (SRTM)

1. Measure (hash)

TPM + firmware

Boot Loader

OS kernel

PCR: platform configuration register
Secure Boot using TPM

- Static root of trust for measurement (SRTM)

1. Measure (hash)

   - TPM MR after reboot
   - Boot Loader
     - SHA-1( )
     - sent to TPM
   - TPM MR when boot loader executes
     - SHA-1( )
     - sent to TPM

2. Report (extend)

   - OS Kernel
     - SHA-1( )
     - sent to TPM
   - Kernel module
     - SHA-1( )
     - sent to TPM
   - OS kernel
     - SHA-1( )
     - sent to TPM

TPM + firmware

Boot Loader

OS kernel

PCR: platform configuration register
Secure Boot using TPM

- Static root of trust for measurement (SRTM)

1. Measure (hash)
2. Report (extend)
3. Load

PCR: platform configuration register
Secure Boot using TPM

• Static root of trust for measurement (SRTM)

1. Measure (hash)
2. Report (extend)
3. load

TPM + firmware

Boot Loader

OS Kernel

Kernel module

OS kernel

TPM MR after reboot

TPM MR when boot loader executes

SHA-1( ) sent to TPM

SHA-1( )

TPM MR when OS kernel executes

SHA-1( ) sent to TPM

SHA-1( )

TPM MR when Kernel Module executes

PCR: platform configuration register
Secure Boot using TPM

• Static root of trust for measurement (SRTM)

1. Measure (hash)

2. Report (extend)

3. Load

TPM + firmware

Boot Loader

OS kernel

Compared to expected values locally or submitted to a remote attester.

PCR: platform configuration register
Software Attestation

• Report a measurement list to a remote verifier
Software Attestation

• Report a measurement list to a remote verifier
Software Attestation

• Report a measurement list to a remote verifier
• Problem: How can the verifier know the list is not faked?
Public Key Cryptography (e.g., RSA, EC)

• A pair of keys:
  • Private key ($K_{pri}$ – kept as secret); Public key ($K_{pub}$ – safe to release publicly)
Public Key Cryptography (e.g., RSA, EC)

• A pair of keys:
  • Private key ($K_{pri}$ – kept as secret); Public key ($K_{pub}$ – safe to release publicly)

• Encryption:
  • Encrypt(plaintext, $K_{pub}$) = ciphertext
  • Decrypt(ciphertext, $K_{pri}$) = plaintext
Public Key Cryptography (e.g., RSA, EC)

• A pair of keys:
  • Private key ($K_{pri}$ – kept as secret); Public key ($K_{pub}$ – safe to release publicly)

• Encryption:
  • Encrypt(plaintext, $K_{pub}$) = ciphertext
  • Decrypt(ciphertext, $K_{pri}$) = plaintext

• Digital signatures:
  • Proof that msg comes from whoever owns private key corresponding to $K_{pub}$
Public Key Cryptography (e.g., RSA, EC)

- A pair of keys:
  - Private key ($K_{pri}$ – kept as secret); Public key ($K_{pub}$ – safe to release publicly)

- Encryption:
  - Encrypt(plaintext, $K_{pub}$) = ciphertext
  - Decrypt(ciphertext, $K_{pri}$) = plaintext

- Digital signatures:
  - Proof that msg comes from whoever owns private key corresponding to $K_{pub}$
  - Sign(msg):
    - $h = \text{Hash}(\text{msg})$; signature = Encrypt($h$, $K_{pri}$)
    - Return {signature, msg}
Public Key Cryptography (e.g., RSA, EC)

• A pair of keys:
  • Private key ($K_{pri}$ – kept as secret); Public key ($K_{pub}$ – safe to release publicly)

• Encryption:
  • Encrypt(plaintext, $K_{pub}$) = ciphertext
  • Decrypt(ciphertext, $K_{pri}$) = plaintext

• Digital signatures:
  • Proof that msg comes from whoever owns private key corresponding to $K_{pub}$
  • Sign(msg):
    • $h = \text{Hash}(\text{msg}); \text{signature} = \text{Encrypt}(h, K_{pri})$
    • Return {signature, msg}
  • Verify:
    • Decrypt(signature, $K_{pub}$) $\neq \text{Hash}(\text{msg})$
Software Attestation

Processor Chip (w/ BIOS, OS, Apps)

TPM $K_{pri}$

$\text{sign}$ Measurement List

Verifier $K_{pub}$
Software Attestation

Replay attack: How can the verifier know the list is the latest?

Processor Chip (w/ BIOS, OS, Apps)

TPM

$K_{pri}$

sign

Measurement List

Verifier

$K_{pub}$
Software Attestation

• Defend against replay attack: Freshness

Processor Chip (w/ BIOS, OS, Apps)

TPM

K_{pri}

\text{sign} \quad \text{Measurement List + nonce}

Verifier

K_{pub}

nonce
Software Attestation

• Defend against replay attack: Freshness

Processor Chip (w/ BIOS, OS, Apps)

TPM

$K_{pri}$

$K_{pub}$

Verifier

How to know this key belong to a specific TPM?

nonce

Measurement List + nonce

sign
Software Attestation

• Need public key infrastructure
Software Attestation

• Need public key infrastructure

Processor Chip (w/ BIOS, OS, Apps)

TPM

AIK\textsubscript{pri}

\begin{itemize}
  \item sign
  \item Measurement List + nonce
  \item Verifier
\end{itemize}

Chip Manufactory

\begin{itemize}
  \item Root Key
    \begin{itemize}
      \item RK\textsubscript{pri}
      \item RK\textsubscript{pub}
    \end{itemize}
\end{itemize}

Attestation Identity Key

AIK\textsubscript{pub}

6.888 L2 - Secure Processors in Industry
Software Attestation

- Need public key infrastructure

Processor Chip (w/ BIOS, OS, Apps)

TRM

AIK_{pri}

chip manufacturer

Root Key

RK_{pri}  RK_{pub}

Attestation identity key

AIK_{pub}

Verifier

Measurement List + nonce

Sign

Works as Certificate Agent
Software Attestation

• Need public key infrastructure

Processor Chip (w/ BIOS, OS, Apps)

TRM

AIK_{pri}

sign

Measurement List + nonce

Verifier

AIK_{pub}

Chip Manufactory

Root Key

RK_{pri}  RK_{pub}

Attestation Identity Key

sign

Works as Certificate Agent
Security Objectives Summary

• Privacy
  • Alice sends msg $m$ to Bob. Only Bob should be able to read $m$. (asymmetric or symmetric encryption)

• Integrity
  • Alice sends msgs $m_1 \ldots m_n$ to Bob.
  • Authenticity: Bob receives msg $p$. Bob can verify $p \in m_1 \ldots m_n$. (Hash)
  • Freshness: Bob has received msgs $p_1 \ldots p_n$. Bob can verify $p_i = m_i$. (Hash+nonce)

• Identity
  • Bob wants to know if Alice is really Alice.

• Availability
  • Does Bob ever see the $n$ messages?

Protocols can be constructed using crypto primitives and infrastructures
Intel TXT

• Uses TPM for software attestation
• Dynamic root of trust for measurement (DRTM)
  • PCRs 17-22 are reset by the SINIT ACM, every time a TXT VM is launched
• Marketed as more secure, but there are various attacks targeting TXT
Open-source Choice: Google Titan

Security Vulnerabilities of Using TPM

Han et al. A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping. Usenix Security’18
Wojtczuk et al. Attacking Intel TXT® via SINIT code execution hijacking. 2011
Security Vulnerabilities of Using TPM

- Vulnerable to bus sniffing attacks
- TPM Reset attacks
  - SW reports hash values
- Bugs in the trusted software

Han et al. A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping. Usenix Security’18
Wojtczuk et al. Attacking Intel TXT® via SINIT code execution hijacking. 2011
So Far ......
So Far ......

The trend: shrink TCB. Why?

Ring 0
Host OS
SMM
Hardware
App

Ring 3

Ring 0
Guest OS
Hypervisor
SMM (firmware)
Hardware
App

Guest OS

Trusted

6.888 L2 - Secure Processors in Industry 33
Shrink Trusted Computing Base (TCB)
Shrink Trusted Computing Base (TCB)

Arm TrustZone
Shrink Trusted Computing Base (TCB)

6.888 L2 - Secure Processors in Industry
Shrink Trusted Computing Base (TCB)

Arm TrustZone

Intel SGX

AMD SEV

Guest OS

App

Hypervisor

SMM

Hardware

Trusted

enclave

6.888 L2 - Secure Processors in Industry
Arm TrustZone

from Hua et al. vTZ: Virtualizing ARM TrustZone. Usenix’17
Arm TrustZone

Recommend not to have secret data in DRAM.
But just recommendation.
Cache is dynamically partitioned by tagging each cache line with a security bit.

from Hua et al. vTZ: Virtualizing ARM TrustZone. Usenix’17
Shrink Trusted Computing Base (TCB)

Arm TrustZone

Intel SGX

AMD SEV

Guest OS

App

Hypervisor

SMM

Hardware

Trusted

enclave

6.888 L2 - Secure Processors in Industry
Next Lecture:
Intel SGX