"TRUSTED" COMPUTING

DISTRIBUTED OPERATING SYSTEMS

Hermann Härtig
Summer 2017
Understand principles of:

- Authenticated booting, relation to (closed) secure booting
- Remote attestation
- Sealed memory
- Dynamic root of trust, late launch
- Protection of applications from the OS
- Point to variants of implementation in HW (TPM, SGX)

Beware of terminology changes!

Non-Goal:

- Lots of TPM, TCG, Trustzone, SGX details
  → read the documents once needed
SOME TERMS

- Secure Booting
- Authenticated Booting
- (Remote) Attestation
- Sealed Memory
- Late Launch / dynamic root of trust
- Trusted Computing (Group) / Trusted Computing Base

Attention: terminology occasionally changes
Trusted Computing Base (TCB)

- The set off all components, hardware, software, procedures, that must be relied upon to enforce a security policy.

Trusted Computing (TC)

- A particular technology comprised of authenticated booting, remote attestation and sealed memory.
TC KEY PROBLEMS

- Can running certain Software be prevented?
- Which computer system do I communicate with?
- Which stack of Software is running?
  - In front of me?
  - On my server somewhere?
- Restrict access to certain secrets (keys) to certain software?
- Protect an application against the OS
Digital Rights Management:

- Provider sells content
- Provider creates key, encrypts content
- Client downloads encrypted content, stores on disk
- Provider sends key, but needs to ensure that only specific SW can use it
- Has to work also when client is off line
- PROVIDER DOES NOT TRUST CUSTOMER
Usage Examples (2)

Virtual machine provided by cloud

- Client buys Cycles + Storage (Virtual machine)
- Client provides its own operating system
- Needs to ensure that provided OS runs
- Needs to ensure that provider cannot access data
- CUSTOMER DOES NOT TRUST PROVIDER
Usage Examples (3)

Industrial Plant Control (Uranium enrichment)

- Remote Operator sends commands, keys
- Local operator occasionally has to run test SW, update to new version, ...
- Local technicians are not Trusted
Anonymity Service

- Intended to provide anonymous communication over internet
- Legal system can request introduction of trap door (program change)
- Anonymity-service provider not trusted
Measuring

- "process of obtaining metrics of platform characteristics"
- Example for metric: Hash-Codes of SW

Attestation

- "vouching for accuracy of information"

Sealed Memory

- binding information to a configuration
Principle Method:
separate critical Software
rely on small Trusted Computing Base

- Small OS kernels
  micro kernels, separation kernels, ....

- Hardware/Microcode
SMALL TRUSTED COMPUTING BASE

Hardware

Linux

X11

App.

APP !
SMALL TRUSTED COMPUTING BASE

App.

X11

Linux

Hardware

Helper

APP!

Intel SGX
NOTATION

- **H(M)**
  Collision-Resistant Hash Function H applied to content M

- **S^{pair}: S^{priv}, S^{pub}**
  Asymmetric key pair of entity S used to **conceal** or **sign** some content
  S^{pub} is published, S^{priv} must be kept secret

- **S^{symm}**
  symmetric key, must be kept secret (“secret key”)
\[ S^{\text{pair}} : \ S^{\text{priv}} \quad S^{\text{pub}} \quad \text{Asymmetric key pair of entity } S \]

\[ S^{\text{symm}} \quad \text{Symmetric Key} \]

- **“Digital Signature”:** \[ \{ M \} S^{\text{priv}} \]
  \[ S^{\text{pub}} \] can be used to verify that \( S \) has signed \( M \)
  \[ \text{is short for: } \left( M, \text{encrypt}(H(M), S^{\text{priv}}) \right) \]

- **“Concealed Message”:** \[ \{ M \} S^{\text{pub}} \]
  Message concealed for \( S \)
  \[ S^{\text{pub}} \] is needed to unconceal \( M \)
■ "Digital Signature": \( \{ M \} S^{\text{priv}} \)
   
   \( S^{\text{pub}} \) is used to verify that \( S \) has signed \( M \)
   
   is short for: \( M, \text{encrypt}(H(M), S^{\text{priv}}) \)

■ "\( \{ M \} S^{\text{pub}} \)" Message concealed for \( S \)
   
   does not necessarily imply public key encryption for full \( M \)
   
   (rather a combination of symmetric and asymmetric methods)
TAMPERRESISTANT BLACK BOX (TRB)

TRB Conceptional View

CPU

Memory

Non-Volatile Memory (NVM):

Platform Configuration Regs (PCR):
WAYS TO “BURN IN” THE OS OR SECURE BOOTING

- Read-Only Memory
- H(OS) in NVM preset by manufacturer
  - load OS- Code
  - compare H(loaded OS code) to preset H(OS)
  - abort if different
- FSKpub in NVM preset by manufacturer
  - load OS- Code
  - check signature of loaded OS-Code using FSKpub
  - abort if check fails
Steps:

1. Preparation by TRB and OS Vendors
2. Booting & “Measuring”
3. Remote attestation
TAMPERRESISTANT BLACK BOX (TRB)

Conceptional View

CPU

Memory

NVM:

PCR:
TAMPERRESISTANT BLACK BOX (TRB)
TRB generates key pair: „Endorsement Key“ $E_{K}^{\text{pair}}$
stores $E_{K}^{\text{priv}}$ in TRB NVM
emits $E_{K}^{\text{pub}}$
TRB AND OS VENDOR

- TRB vendor certifies:
  \{"a valid EK\", EK_{pub}\}^{\text{TRB}_\text{Vendor}_{priv}}

- OS-Vendor certifies:
  \{"a valid OS\", H(OS)\}^{\text{OS}_\text{Vendor}_{priv}}

- serve as identifiers:
  EK_{pub} and H(OS)
TRB:
- resets TRB
- measures OS code $H(\text{OS})$
- stores $H(\text{OS})$ in PCR

PCR not (directly) writable by OS
more later
Challenge: send NONCE

Response: \{\text{NONCE}', \text{PCR}\}_{E^{K_{\text{priv}}}}
■ boot Linux
  - challenge
  - response “Linux”

■ reboot Windows
  - send data

add one step of indirection:
create keypairs at each reboot
At booting, TRB:

- Computes H(OS) and stores in PCR
- Creates 2 keypairs for the booted, “active” OS:
  - $\text{ActiveOSAuth}^{\text{pair}}$ /* for Authentication
  - $\text{ActiveOSCons}^{\text{pair}}$ /* for Concellation
- Certifies: $\{ \text{ActiveOSAuthK}^{\text{pub}},$ 
  $\text{ActiveOSConsKpub},$ 
  $\text{H(OS)}\} \ EK^{\text{priv}}$
- Hands over ActiveOSKeys to booted OS
Remote Attestation:

- Challenge: nonce

- Active OS generates response:
  \[
  \{ \text{ActiveOSCons}^{\text{pub}}, \text{ActiveOSAuth}^{\text{pub}}, \text{H(OS)}^{\text{EK}}^{\text{priv}} \} \times \text{see previous slide}
  \{\text{nonce}'\} \text{ActiveOSAuth}^{\text{priv}}
  \]

Secure channel:

\[
\{ \text{message} \} \text{ActiveOSCons}^{\text{pub}}
\]
ASSUMPTIONS

- TRB can protect: $E_{K^{\text{priv}}}$, PCR
- OS can protect: “Active OS keys”
- Rebooting destroys content of
  - PCR
- Memory Holding “Active OS keys”
2 Problems:

- Very large Trusted Computing Base for Booting (Drivers etc)
- Remote attestation of one process (leaf in tree)
“Extend” Operation:

- stack: \( PCR_n = H(PCR_{n-1} \ || \ next\text{-}component) \)
- tree: difficult (unpublished?)

Key pairs per step:

- OS controls applications → generate key pair per application
- OS certifies
  - \{ Application 1, App1Kpub \} ActiveOS^{priv}
  - \{ Application 2, App2Kpub \} ActiveOS^{priv}
Problem: huge Software to boot system !!!

- Use arbitrary SW to start system and load all SW
- provide specific instruction to enter “secure mode”
  - set HW in specific state (stop all processors, IO, …)
  - Measure “root of trust” SW
  - store measurement in PCR

- AMD: “skinit” (Hash) arbitrary root of trust
- Intel: “senter” (must be signed by chip set manufacturer)
Problem:

- Send information using secure channels
- Bind that information to Software configuration
- Work offline:
  How to store information in the absence of communication channels?
- For example DRM:
  bind encryption keys to specific machine, specific OS
Tamper-resistant black box

- PCR: $H(OS)$
- Win 7
- SUSE-Linux
- L4-Test-Version

Add / delete entry
Read / write

SEALED MEMORY

Hermann Härtig, TU Dresden, Distributed OS, Trusted Computing, SS 2017
Tamper-resistant black box

PCR: $H(Win-7)$

Win 7
SUSE-Linux
L4-Test-Version

Add / delete entry
Read / write
Tamper-resistant black box

- PCR: $H(\text{SUSE})$
- Win 7
- SUSE-Linux
- L4-Test-Version

Add / delete entry
Read
Write
SEALED MEMORY: SEAL OPERATION

Tamper-resistant black box

PCR: $H(\text{Win-7})$

Win 7

SUSE-Linux

L4-Test-Version

Message

Sealed Message
Tamper-resistant black box

PCR: $H(Win-7)$

Win 7

SUSE-Linux

L4-Test-Version

Sealed Message

Message
TRB generates symmetric Storage Key (S) never leaves chip
Seal(message):
  encrypt("PCR, message", S) \rightarrow "sealed_message";
  emit sealed_message

Unseal(sealed_message):
  decrypt(sealed_message, S) \rightarrow "SealTime_PCR, message";
  If SealTime_PCR == PCR then emit message
  else abort
Seal(message, FUTURE_Config):
    encrypt(“FUTURE_Config, message”, S)
    → “sealed_message”;
    emit sealed_message

“seals” information such that it can be unsealed by a future configuration (for example: future OS version)
■ Win8: Seal ("SonyOS, Sony-Secret")
  → SealedMessage (store it on disk)

■ L4: Unseal (SealedMessage)
  → SonyOS, Sony-Secret
  → PCR#SonyOS
  → abort

■ SonyOS: Unseal(SealedMessage)
  → SonyOS, Sony-Secret
  → PCR==SonyOS
  → emit SonySecret
Ideally, includes CPU, Memory, ...

Current practice

- Additional physical protection, for example IBM 4758 …
  look it up in Wikipedia

- HW versions
  - TPM:
    separate “Trusted Platform Modules”
    (replacing BIOS breaks TRB)
  - Add a new privilege mode: ARM TrustZone
  - raise to user processes: Intel SGX
TCG PC PLATFORMS: “TRUSTED PLATFORM MODULE” (TPM)
TPM

IO

NV Store  PCK  EK / AIK  Internal Program

SHA-1  RSA  Key gen  Random number
ARM TRUSTZONE

Normal World

PL0
User
User

PL1
Kernel
Kernel

PL2
Hypervisor

Secure World

Monitor

PL0
Trusted Services

PL1
Trusted OS

Hermann Härtig, TU Dresden, Distributed OS, Trusted Computing, SS 2017
INTEL SGX

CPU

Memory

Non-Volatile Memory (NVM):

Platform Configuration Regs (PCR):

Conceptional View

TRB
INTEL SGX

bound to application “enclaves”
- established per special new instruction
- measured by HW
- provide controlled entry points
- resource management via untrusted OS
Important Foundational Paper:

Authentication in distributed systems: theory and practice
Butler Lampson, Martin Abadi, Michael Burrows, Edward Wobber
ACM Transactions on Computer Systems (TOCS)
MORE REFERENCES

- TCG Specifications: https://www.trustedcomputinggroup.org/groups/TCG_1_3_Architecture_Overview.pdf
- ARM Trustzone & Intel SGX vendor sources