Authenticated Booting, Remote Attestation, Sealed Memory
aka „Trusted Computing“
Goals

**Understand principles of:**

- Authenticated booting, difference to (closed) secure booting
- Remote attestation
- Sealed memory
- Dynamic root of trust
- Protection of applications from the OS
- Some variants of implementations (HW)

**Non-Goal:**

- Lots of TPM, TCG-Spec 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 has changed
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 compromised 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?

- Can I restrict access to certain secrets (keys) to certain software?

- Can I protect an application against the OS
1) End User Example

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 CLIENT
2) Cloud Example

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
- CLIENT DOES NOT TRUST PROVIDER
3) Industrial Plant Example

(Uranium Enrichment) Plant Control

- Remote Operator sends commands, keys
- Local operator occasionally has to run test SW, update to new version, ...
- Local technicians are not Trusted
4) Anonymizer example

Anonymity Service

• Intended to provide anonymous communication over internet
• Legal system can request introduction of trap door (program change)
• Service provider not trusted
Trusted Computing Terminology

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
An example application: DRM

• „Digital Content“ is encrypted using symmetric key

• Smart-Card
  • contains key
  • authenticates device
  • delivers key only after successful authentication

• Assumptions
  • Smart Card can protect the key
  • „allowed“ OS can protect the key
  • OS cannot be exchanged
Small Trusted Computing Base

- App.
- X11
- Linux
- Mini OS
- Hardware
- DRM
- GUI

Diagram showing the layers of a small trusted computing base with emphasis on Mini OS, Hardware, and related components such as Linux, X11, and DRM.
Protection of Application

Principle Method:
separate critical Software
rely on small Trusted Computing Base

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

- Hardware
Notation

● $SK^{\text{priv}}$ $SK^{\text{pub}}$ Asymmetric key pair of some entity S

$\{ M \}XK^{\text{priv}}$ Digital Signature for message M using the private key of signer X

$\{ M \}YK^{\text{pub}}$ Message encrypted using public cancellation key of Y

● $H(M)$ Collision-Resistant Hash Function

● **Certificate** by authority Ca:

$\{ ID, SK^{\text{pub}} , \text{other properties} \} \ CaK^{\text{priv}}$
Note:

- “{ M }Sk^{priv} Digital Signature”
  is short for: encrypt(H(M),Sk^{priv})

- “{ M }Sk^{pub} Message concealed ...”
  does not necessarily imply public key encryption
  for full M
  (rather a combination of symmetric
  and asymmetric methods)
Identification of Software

• Program vendor: Foosoft FS

• Two ways to identify Software: Hash / public key
  • H(Program)
  • {Program, ID- Program}FSK^{priv}
    use FSK^{pub} to check the signature must be made available, e.g. shipped with the Program

• The „ID“ of SW must be made available somehow.
Tamper-resistant black box

- CPU
- Memory
- Non-Volatile Memory:
- Platform Configuration Registers:
Ways to “burn in” the OS or secure booting

- Read-Only Memory

- Allowed H(OS) in NV memory preset by manufacturer
  - load OS- Code
  - compare H(loadded OS code) to preset H(OS)
  - abort if different

- Preset $FSK_{\text{pub}}$ in NV memory preset by manufacturer
  - load OS- Code
  - check signature of loaded OS-Code using $FSK_{\text{pub}}$
  - abort if check fails
Authenticated Booting, using HASH

Steps:

- Preparation by Manufacturers (TRB and OS)
- Booting & “Measuring”
- Remote attestation
Authenticated Booting, using HASH

CPU

Memory

Non-Volatile Memory:
“Endorsement Key” EK preset by Manufacturer

Platform Configuration Registers:
PCR:
Hash-Code obtained during boot
Vendors of TRB and OS

- TRB generates key pair: „Endorsement Key“ (EK)
  - stores in TRB NV Memory: \( EK_{\text{priv}} \)
  - emits: \( EK_{\text{pub}} \)

- TRB vendor certifies: \{“a valid EK”, \( EK_{\text{pub}} \)\( TVK_{\text{priv}} \)

- OS-Vendor certifies: \{“a valid OS“, \( H(OS) \)\( OSVK_{\text{priv}} \)

- serve as identifiers: \( EK_{\text{pub}} \) and \( H(OS) \)
Booting & Attestation, using HASH

Booting:

- TRB “measures” OS- Code (computes $H(\text{OS-Code})$)
- stores in PCR
- no other way to write PCR

Attestation:

- Challenge: nonce
- TRB generates Response: $\{\text{PCR, nonce}', \text{EK}^{\text{priv}}\}$
Authenticated Booting, using public key

Non-Volatile Memory:
“Endorsement Key” EK preset by Manufacturer

Platform Configuration Registers:
PCR:
$\text{OSK}^{\text{pub}}$ used to check OS
Vendors of TRB and OS, using Key

- TRB generates key pair:
  - stores in TRB NV Memory: $E_{K^{\text{priv}}}$
  - emits: $E_{K^{\text{pub}}}$

- TRB vendor certifies: \{“a valid EK”, $E_{K^{\text{pub}}}$\}$_{TVK^{\text{priv}}}$

- OS-Vendor certifies: \{„a valid OS“, $OS_{K^{\text{pub}}}$\}$_{OSVK^{\text{priv}}}$

- and signs OS-Code: \{OS-Code\}$_{OSK^{\text{priv}}}$

- serve as identifiers: $E_{K^{\text{pub}}}$ and $OS_{K^{\text{pub}}}$
Booting:

- TRB checks OS- Code using some $OSK^{pub}$
- stores $OSK^{pub}$ in PCR
- no other way to write PCR

Attestation:

- Challenge: nonce
- TRB generates Response: $\{PCR, \text{ nonce}' \}E^{priv}$
A Race condition

Nonce → new active OS

Content → new active OS

Content → new active OS

Boot Linux

Boot Windows

Boot Linux
Auth. Booting considering reboot

- attestation required at each request
- Do not use EK

This is one way of doing it:

create new keypairs on every reboot
Booting (AB considering reboot)

Booting:

- TRB checks OS- Code using some $\text{OSK}^{\text{pub}}$
- store $\text{OSK}^{\text{pub}}$ in PCR

- create 2 keypairs for the booted OS (“Active OS”):
  - $\text{ActiveOSAuthK}$ /* for Authentication
  - $\text{ActiveOSConsK}$ /* for Concellation

- certifies: $\{\text{ActiveOSAuthK}^{\text{pub}}, \text{ActiveOSConsK}^{\text{pub}}, \text{OSK}^{\text{pub}}\}E_{\text{EK}^{\text{priv}}}$
- Hand over ActiveOSKeys to booted OS
Attestation (AB considering reboot)

Remote Attestation:

- Challenge: nonce
- Active OS generates response:

{ ActiveOSConsK_{pub}, ActiveOSAuthK_{pub}^{\text{OSK}} \text{EK}_{\text{priv}} }^{\text{ nonce' }} \text{ActiveOSAuthK}_{\text{priv}}^{\text{ nonce' }}

Encrypted Channel via the active OS:

- \{ message \} \text{ActiveOSConsK}_{\text{pub}}^{\text{ message}}
Assumptions

TRB can protect: EK, PCR

OS can protect: ActiveOSAuthK_{priv}  ActiveOSConsK_{priv}

Rebooting destroys content of
  • PCR
  • Memory Holding ActiveOSAuthK_{priv}  ActiveOSConsK_{priv}
Software stacks and trees

Application

GUI

OS Code

OS Loader

ROOT

Application

GUI

OS Code

Application

GUI

OS Loader

ROOT
Software stacks and trees

2 Problems:

- Very large Trusted Computing Base for Booting
- Remote attestation of one process (leaf in tree)
Software stacks and trees

- “Extend” Operation
  - stack: $\text{PCR}_n = H(\text{PCR}_{n-1} || \text{next-component})$
  - tree: difficult (unpublished?)

- Key pairs per step:
  - OS controls applications → generate key pair per application
  - OS certifies
    - $\{ \text{Application 1, App1K}^{\text{pub}} \} \text{ ActiveOSK}^{\text{priv}}$
    - $\{ \text{Application 2, App2K}^{\text{pub}} \} \text{ ActiveOSK}^{\text{priv}}$
Late Launch

- 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)
Sealed Memory

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
Sealed Memory

Tamper-resistant black box

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

Add / delete entry
Read / write
Sealed Memory

Tamper-resistant black box

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

Add / delete entry
Read / write
Sealed Memory

Tamper-resistant black box

- PCR: \( H(\text{SUSE}) \)
- Add / delete entry
- Read / write

- Win 7
- SUSE-Linux
- L4-Test-Version
Sealed Memory: Seal Operation

Tamper-resistant black box

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

Message
Sealed Message
Sealed Memory: Unseal Operation

Tamper-resistant black box

PCR: H(Win-7)

Win 7

SUSE-Linux

L4-Test-Version

Sealed Message

Message
Tamperresistant black box (TRB)

CPU

Memory

Non-Volatile Memory:

S: Storage key created by my manufacturer seen by nobody

Platform Configuration Register:

PCR: „SW-config“
Sealed Memory

- **Seal(message):**
  
  encrypt(“PCR, message”, Storage-Key)
  
  → “sealed message”;
  
  emit sealed message

- **Unseal(sealed_message):**
  
  decrypt( “sealed_message”, Storage-Key)
  
  → “SW config, message”;
  
  If SW config == PCR then emit message else abort fi
Sealed Memory for future configuration

- Seal(message, \texttt{FUTURE\_Config}):
  \[\text{encrypt(“FUTURE\_Config, message”, Storage-Key)}\]
  \[\rightarrow \text{“sealed message”;}
  \text{emit sealed\_message}\]

- “seals” information such that it can be unsealed by a future configuration (for example: future version)
Example

- 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 → ok
Tamper Resistant Box?

- Ideally, includes CPU, Memory, ...

- In practice
  - Additional physical protection, for example IBM 4758 ... look it up in Wikipedia
  - Recent HW versions
    - TPM: separate “Trusted Platform Modules” (replacing BIOS breaks TRB)
    - Add a new privilege mode:
      - ARM TrustZone
      - Intel SGX
TCG PC Platforms: “Trusted Platform Module” (TPM)
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
- http://www.slideshare.net/daniel_bilar/intel-sgx-2013
- ARM Trustzone