Property-Based Attestation Approach and Virtual TPM

Ahmad-Reza Sadeghi  
sadeghi@crypto.rub.de  
Horst Görtz Institute for IT-Security  
Ruhr-University Bochum, Germany  
www.trust.rub.de  

ReTrust Meeting, Villach 2008
§ Motivation
§ Trusted Virtualization
Motivation

- Hardware virtualization a (re-discovered / re-invented) useful means to reduce to total cost
  - apparent in corporate data centers
  - however, workloads should be processed separately due to diversity of security objectives of the involved parties (see [Barham et al 2003, Sailer et al 2005])

- Combine hypervisors (Virtual Machine Monitors) with hardware-based root of trust
  - Hypervisors provide isolations of workloads
    - mediating access to physical resources by virtual machines
  - Hardware root of trust is resistant to software attacks and provides a basis for reasoning about the integrity of SW running on a platform
Possible Architecture

Applications
- Existing OS
- Security applications

Legacy OS (e.g., Linux)
- untrusted storage
- vTPM

HDDEnc

Isolation

Legacy OS (e.g., Linux)
- untrusted storage
- vTPM

Attestation Manager
- Hypervisor (IPC, Hardware Sharing, Memory Management, Scheduling, etc.) based on microkernel or Xen

Storage Manager
- Secure GUI

Component Manager
- Applications
  - Existing OS
  - Security applications

Secure VMM

Virtualization Layer

Conventional Hardware
- TPM

Hardware
- CPU
- Devices
- Trusted Computing (TC) Technology (TPM, Trusted Execution Technology (TXT), Presidio, etc.)
Components

- **TC enabled hardware**

- **Trusted Service Layer**
  - Trust Manager: controls access to TPM interface
  - Compartment Manager
    - manages creation, updates, and deletion of compartments
    - measures compartments and assigns unique IDs to them
  - Storage manager
    - guarantees trusted storage, i.e., authenticity, confidentiality and integrity (and freshness) of stored data
    - has access to configuration of clients it is communicating to over trusted channel
  - Secure GUI
    - guarantees a trusted path to application

- **Virtualization Layer**
  - provides abstraction of physical machine
  - Provides isolation between virtual machines
Use Case: Corporate Computing

Private Environment
- e.g., Protection of hard disk encryption application

Corporate Environment
- Classified: Stronger security requirement on usage of encryption keys bound to specific hardware
- Unclassified: migrate working environment at home

request/response path between vTPM-Manager, vTPMs and the hardware TPM
The Big Picture

- **Trustworthiness in distributed IT systems**
  - Different parties with potentially conflicting requirements involved
  - Cryptographic methods are of limited help
  - Example applications (signatures, Grid, online voting and banking, ….)

- **How to define „trustworthiness“?**
- **How to determine/verify it?**
- **How could common computing platforms support such functionality?**
  - Even a secure OS cannot verify own integrity

- **The role of Trusted Computing**
  - Enable the reasoning about the „trustworthiness“ of own and others
§ TCG Approach to Trusted Computing
Basic Idea for Trusted Platform

- Trusted components in hardware and software
- Provides a variety of functions that must be trusted
  - in particular a set of cryptographic and security functions
- Creates a foundation of trust for software
- Provides hardware protection for sensitive data
  - e.g., keys, counters, etc.
- Desired goals
  - Trusted Computing Base (TCB) should be minimized
  - Compatibility to commodity systems
Trusted Computing Group (TCG)

- Consortium of IT-Enterprises (since April 2003)
  - Today more than 120 members [TCG]
    - www.trustedcomputing.org/about/members/
- Focus on development of hardware-enabled trusted computing and security technology across multiple platforms and devices
- Evolved from Trusted Computing Platform Alliance (TCPA)
  - Formed by Hewlett-Packard (HP), Compaq (today part of HP), IBM, Intel and Microsoft in January 1999
- Published various specifications
- Set up various working groups
TCG Main Specification

• Trusted Platform Module (TPM) [TPM2002, TPM2003, TPM2007]
  • Provides a set of immutable cryptographic and security functions

• Trusted Software Stack (TSS) [TSS2003, TSS2007]
  • Issues low-level TPM requests and receives low-level TPM responses on behalf of higher-level applications
§ Trusted Platform Module: Main TCG Specification
Trusted Platform Module (TPM)

- Current implementation is a cryptographic co-processor
  - Hardware-based random number generation
  - Small set of cryptographic functions
    - Key generation, signing, encryption, hashing, MAC
- Offers additional functionalities
  - Secure storage (ideally tamper-resistant)
  - Platform integrity measurement and reporting
- Embedded into the platform’s motherboard
- Acts as a “Root of Trust”
  - TPM must be trusted by all parties
- Two versions of specification available
- Many vendors already ship their platforms with a TPM [TPMMMatrix2006]
<table>
<thead>
<tr>
<th>Trusted Platform Module (TPM)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cryptographic Co-Processor</td>
<td>Input/Output</td>
</tr>
<tr>
<td>• Asymmetric en-/decryption (RSA)</td>
<td>• Protocol en-/decoding</td>
</tr>
<tr>
<td>• Digital signature (RSA)</td>
<td>• Enforces access policies</td>
</tr>
<tr>
<td>SHA-1</td>
<td>Opt-In</td>
</tr>
<tr>
<td>HMAC</td>
<td>• Stores TPM state information (e.g., if TPM is disabled)</td>
</tr>
<tr>
<td>Random Number Generation</td>
<td>• Ensures state-dependent limitations (e.g., some commands must not be executed if the TPM is disabled)</td>
</tr>
<tr>
<td>Key Generation</td>
<td>Execution Engine</td>
</tr>
<tr>
<td>• Asymmetric keys (RSA)</td>
<td>• Processes TPM commands</td>
</tr>
<tr>
<td>• Symmetric keys</td>
<td>• Ensures segregation of operations</td>
</tr>
<tr>
<td>• Nonces</td>
<td>• Ensures protection of secrets</td>
</tr>
<tr>
<td>Platform Configuration Registers (PCR)</td>
<td>Non-Volatile Memory</td>
</tr>
<tr>
<td>• Storage of integrity measurements</td>
<td>• Stores persistent TPM data (e.g., the TPM identity or special keys)</td>
</tr>
<tr>
<td></td>
<td>• Provides read-, write- or unprotected storage accessible from outside the TPM</td>
</tr>
</tbody>
</table>

System Interface (e.g., LPC-Bus)
TPM Integration into PC-Hardware

Central Processing Unit (CPU)

Graphics and Memory Controller HUB (GMCH)
Chipset (Northbridge)

System Memory

Interface Controller HUB (ICH)
Chipset (Southbridge)

Low Pin Count (LPC) Bus

System BIOS

TPM

Super I/O (Legacy Devices)

Floppy Drive

PS/2

USB Devices

Network Interface

Parallel I/O

Serial I/O
Integrity Measurement

- **Integrity Measurement**
  - Process of obtaining metrics of platform characteristics that affect the integrity (trustworthiness) of a platform and storing digests of those metrics to the TPM’s PCRs
    - Platform characteristic = digest of the software to be executed

- **Platform Configuration Registers (PCR)**
  - Shielded location to store integrity measurement values
  - Can only be extended: \( PCR_{i+1} \leftarrow SHA-1(\ PCR_i, value ) \)
  - PCRs are reset only when the platform is rebooted

- **Integrity Logging**
  - Storing integrity metrics in a log for later use
  - e.g., storing additional information about what has been measured like software manufacturer name, software name, version, etc.
Performing Integrity Measurements

1. CRTM measures entity E
2. creates Event Structure in TPM Event Log
   - SML contains the Event Structures for all measurements extended to the SM
   - SM Event Log can be stored on any storage device
     - E.g., hard disk
3. extends value into Registers
4. Executes/passes control to entity E
Core Root of Trust for Measurement (CRTM)

- Immutable portion of the host platform’s initialization code that executes upon a host platform reset
- Trust in all measurements is based on the integrity of the CRTM
- Ideally the CRTM is contained in the TPM
- Implementation decisions may require to locate it in other firmware (e.g., BIOS boot block)
Two Possible CRTM Implementations

- **CRTM is the BIOS Boot Block**
  - BIOS is composed of a BIOS Boot Block and a POST BIOS
  - Each of these are independent components
    - Each can be updated independent of the other
  - BIOS Boot Block is the CRTM while the POST BIOS is not, but is a measured component of the Chain of Trust

- **CRTM is the entire BIOS**
  - BIOS is composed of a single atomic entity
  - Entire BIOS is updated, modified, or maintained as a single component
TPM Software Integration

Applications (local)

TCG-Application

Conventional Application

Remote Trusted Platform

Remote TCG-Application

TCG Service Provider

RPC Client

RPC Server

RPC (Remote Procedure Call)

TCG Core Services (TCS)
- key and credential management
- platform integrity measurement and reporting (TPM Event Log)
- parsing and handling of TPM commands

TCG Service Provider (TSP)
- provides object-oriented interface for TCG-enabled applications

Cryptographic Interface (e.g., MS-CAPI, PKCS#11)

TSP Interface (TSPI)

Operating System

TCG Service Provider (TSP)
- provides standard interface for TPMs of different manufacturers
- transition between user mode and kernel mode

TCS Interface (TCSI)

TCG Device Driver Library (TDDL)
- provides object-oriented interface for TCG-enabled applications

TDDL Interface (TDDLI)

Hardware

TPM

System Services

Trusted Software Stack (TSS)
TPM Keys
TPM Key Types

- **TPM provides 9 different types of keys**
  - 3 special TPM key types
    - Endorsement Key, Storage Root Key, Attestation Identity Keys
  - 6 general key types
    - Storage, signing, binding, migration, legacy and "authchange" keys
  - Most important key types explained in following slides
- **Each key may have additional properties, the most important ones are**
  - Migratable, non-migratable, certified migratable
    - e.g., whether the key is allowed to be migrated to another TPM
  - Whether the key is allowed only to be used when the
Special Keys

• **Endorsement key (EK)**
  - TPM identity
  - Generated and certified during manufacturing
  - RSA key

• **Attestation Identity Key (AIK)**
  - Used to sign to current platform configuration
  - Alias for TPM/platform identity EK
  - RSA key
  - TPM/platform may have multiple AIKs

• **Storage Root Key**
  - Secure data storage implemented as a hierarchy of keys
  - Storage Root Key (SRK) is root of this key hierarchy
  - Generated by TPM during process of installing TPM
    Owner
TPM Key Hierarchy

- Depth of hierarchy and number of TPM-protected keys only limited by size of external storage
- Storage keys (StoreK) protect all other key types
  - Attestation ID keys (AIK)
  - Signing keys (SigK)
  - Binding keys (BindK)
  - Migration Keys (MigrK)
  - Symmetric keys (SymK)
- Transitive protection
  - SRK indirectly protects
TPM Key Object – Important Fields

- **General Information**
  - Key Type
  - Algorithm
  - Authorization Secret

- **Specific Information**
  - Key Length
  - Key Data

- **Key Properties**
  - Migration
  - PCR Values

E.g., signing key, binding key, storage key, ...

E.g., RSA, DSA, HMAC, AES, ...

Authorization secret required to use the key.

Public and private key, asymmetric key. Secret key data is encrypted with the corresponding parent key.

Information about the migratability of the key:
- migratable
- certified migratable
- non-migratable

A key can be sealed to specific PCR values. This means that such a key can only be used when the platform is in a specific (trusted) state.
§ Main Functionalities
Authenticated Boot
Bootstrap Architecture in PC

The boot loader provides a link between hardware and the operating system. The execution sequence starts with the boot loader, then the operating system, followed by applications.

- **Hardware**
  - **BIOS**
  - **CPU**
- **Boot Loader**
- **Operating System**
- **Application**
Bootstrap and Integrity Measurement

**Measurement**

- **OS measures Application**
- **BL measures OS**
- **BIOS measures BL**
- **CRTM measures BIOS**

**PC-Hardware**

- **CPU**
- **Boot Loader (BL)**
- **Operating System (OS)**
- **Application**

**Trusted Channel**

- **Remote Party**

**Chain of Trust**

- **TPM**
- **PCR[15]**
- **PCR[14]**
- **PCR[1]**
- **PCR[0]**

**Booting**

- **PCR[15]** measures BIOS
- **PCR[14]** measures BL
- **PCR[1]** measures OS
- **PCR[0]** measures Application

**CRTM Core Root of Trust for Measurement**

- **Trusted Platform Module (TPM)**
- **Trusted components**
Bootstrap and Integrity Measurement

TCG-enabled boot loader

TrustedGRUB [tGRUB2005]

TCG_HashAll; TCG_PassThroughToTPM

Platform Configuration Register
00: BIOS
01: Mainboard Configuration
02: Option ROM
03: Option ROM Configuration
04: Initial Program Loader (IPL)
05: IPL Config & Data
06: RFU (Reserved for Future Usage)
07: RFU
08: First part of „stage2“
09: Rest of „stage2“
13: Arbitrary file measurements
14: Booted system files (e.g., Kernel, modules,...)

TCG-enabled Hardware

BIOS

CRTM

CPU

TPM

Bootloader (Stage1)

TCG API | Pre-Boot-Driver

Bootloader (Stage2)

Measured by

OS Kernel

Service

Service

Service

Hand over control
Binding and Sealing

Binding

- Conventional asymmetric encryption
- May be used to bind data to a specific TPM/platform
  - Data encrypted with non-migratable key can only be recovered by TPM that knows corresponding secret key
- Usually no platform binding
  - Since binding can also be used with migratable keys

Sealing (extension of binding)

- Always binds data to a specific TPM/platform
  - Sealing can only be used with non-migratable storage keys
- Configuration of encrypting platform can be verified
  - Ciphertext includes platform’s state at the time of encryption
- May bind data to a specific platform configuration
  - Data can be decrypted only if platform is in a pre-defined
Integrity Reporting / Attestation
Attestation

• Authentic report of a platform’s state to a (remote) verifier
  • A local or remote verifier (challenger) is interested in platform configuration (e.g., hard- and software environment)
  • Verifier is able to decide whether it trusts the attested configuration
    • e.g., an online-banking client checks whether the bank’s server is in a known secure configuration (e.g., has not been tampered with)

• TPM and CRTM act as Root of Trust for Reporting
  • TPM can generate authentic reports of current integrity measurement values (current PCR content)
Requirements on Attestation

- Attest to all states of entities (machines) capable of affecting the behavior of the entity being attested
  - e.g., hard- and software environment of the attesting platform including history of all executed program code
- Attestation platform’s state report
  - Integrity, confidentiality, freshness
- Authenticity of attestor
- Privacy
  - Regarding information disclosure on system configuration and platform identity
### Simplified TCG Attestation Concept

**Trusted Platform**

**Host H**
- has configuration \( C_H \)
- RTR authentically reports \( C_H \) to TPM

**Attestor (TPM)**
- \( \text{AIK} \leftarrow \text{genkey}(l) \)
- \( \sigma_{TPM} \leftarrow \text{sign}_{\text{AIK}}(N_V, m) \)
- \( m \leftarrow \text{hash}(C_H) \)

**Verifier V**
- local or remote
- interested in configuration \( C_H \) of host H
- Decides whether \( C_H \) is trustworthy (e.g., does not violate V’s security)

**Configuration List**
- List of trusted configurations
- \( \ldots, (m, C_H), \ldots \)

\( N_V \) Nonce (anti-replay value) chosen by the verifier
\( C_H \) current configuration of host H
Related TPM-Interface

• Reporting of PCR values signed by the TPM
  • Command: `TPM_Quote2` and `TPM_Quote` (deprecated)
  • May be called by an attestation system service that handles attestation requests

• Input to `TPM_Quote2` / `TPM_Quote`
  • AIK to be used to sign current PCR values
  • Nonce (anti-replay value)
  • Selection of PCRs to be reported
  • Indicator whether the TPM version and revision should be added to the signed report of PCR values
  • Authorization data for using the AIK
More Details about TCG Attestation

**Trusted Platform**

**Attestation System Service**

cred ← ( pc, cc, ac )

**Attestor (TPM)**

TPM ← sign_{AIK}( PCR[ S_{PCR} ], ver_{TPM}, N_{V} )

**Verifier V**

Request( S_{PCR}, I_{ver}, N_{V} )

ver_{TPM}, σ_{TPM}, log, cred

- verify cred
- verify σ_{TPM} using ver_{TPM} and by re-computing PCR[ S_{PCR} ] from log
- decide whether the events listed in log violate V’s security requirements

*S_{PCR} selection of PCR values V is interested in indicator whether V is interested in TPM version information
*I_{ver} platform credential
*N_{V} Nonce (anti-replay value) chosen by the verifier
*h_{AIK} pointer (handle) to the AIK to be used
*A_{AIK} authorization secret required to use AIK
*σ_{TPM} TPM version information
*pc platform credential
*cc Conformance Credential
*ac Attestation Credential (e.g., from Privacy CA)
*log TPM Event Log
**AIK Certification: Privacy CA I**

**TPM Owner**
- Prove to third parties that it’s platform is in a trustable state
  - E.g., by reporting platform integrity measurements signed with a certified key
- Colluding third parties should not be able to track platform’s transactions
  - E.g., by signing every integrity measurement report with a (ideally) different AIK for each

**Privacy CA**
- Trusted Third Party
- Attests that an AIK belongs to a valid TPM (Attestation Credential)
  - Protocol for certification of an AIK requires disclosure of public EK to Privacy CA
Property-Based Attestation
Problems of Binary Attestation/Sealing

• Discrimination
  • Binding/Sealing allows content and application providers to enforce usage of a specific platform configuration
  • Application vendors can exclude alternative software

• Availability
  • Changed binaries renders sealed data inaccessible

• Privacy
  • Verifiers can gain information on platform configurations

• Management
  • Hugh number of patches, various compiler options, software versions, development environment
  • Changes in binary values (digests) renders bound/sealed data inaccessible
Overview

• Verifier usually interested in properties not configuration

• Property (informally)
  • Describes an aspect of the behaviour of an object with respect to certain requirements (e.g., security-related)

• Properties can be defined on different abstraction levels
  • Privacy-preserving (built-in measures conform to the privacy laws)
  • Provides Multi-Level Security (MLS)
  • Evaluated by a governmental organisation

• Choice of a useful property set and its definition depends on the use case and its requirements
Abstract Model of PBA

Trusted Platform P
provides integrity of host H

Host H (untrusted)
- firmware
- operating system
- applications

Attestor A
- trusted component (hard- and software)
- Securely stores $C_H$

D $\xrightarrow{}$ execute($C_H$)

Veriﬁer V
- local or remote
- can decide whether $C_H$ violates its security requirements
- can “bind/seal” data D to a speciﬁc (probably secure) conﬁguration/state of H

Challenge / Veriﬁer

Verifier V
- $\xleftarrow{}$ challenge
- $\xrightarrow{}$ response
- $\xleftarrow{}$ attest( P)
- $\xrightarrow{}$ bind(P, D)
- $\xleftarrow{}$ init( $C_H$)

Attestor A
- $\xrightarrow{}$ attest( P)
- $\xrightarrow{}$ bind(P, D)
- $\xleftarrow{}$ init( $C_H$)

User / Adversary

$C_H$ initial conﬁguration/state of host H when platform P has been booted

D data to be revealed only if host H is in the (secure) conﬁguration $C_H$
A Possible Approach

- **Delegation-based PBA (DB-PBA)**
  - Property attester proves that another party has certified the desired properties (e.g., certificates [SaSt2004,KuSeSt2007])
  - Hybrid approach since current TPM is used
  - Offline determination of provided properties
  - Well-suited for enterprise environment

- **How to represent property certificates?**
  - Trusted Third Party T with key pair (SK$_T$, PK$_T$)
  - Set of properties $p = \{p_0, \ldots, p_n\}$
  - Code with configuration $C_H$ and property $p$
  - $\text{cert}_T(p, S) := \text{Hash}(C_H), p, \text{Sign}_T(p, cs)$
Possible Instantiations of DB-PBA I

• **Trusted Attestation Service**
  • A trusted software service performs necessary actions
  • Software service is binary attested
  • Service must be fully trusted by the verifier

• **Hardware-based certificate verification**
  • TPM evaluates property certificates
  • New TPM commands required
Possible Instantiation of DB-PBA II

• Group signatures
  • Public group signature key represents a property while private keys represent different configurations
  • Trusted Third Party generates keys
  • TPM functionality has to be extended

• Prove possession of a valid certificate
  • Proof of membership
  • Verifier does not need to trust host
Trusted Platform

cert_{CI} is kept secret from V

Verifier V

1. obtain property certificate cert_{CI}
2. prove possession of a valid certificate that is conform to C_{H}

Hybrid approach since binary measurements are associated with properties.

C_{H} current configuration of host H
p (security) property of C_{H}

cs := Hash(C_{H})
cert_{CI} := sign_{CI}(p, cs)
Prove Possession of Valid certificate II

1. TPM commits to $cs$
   \[ \text{com}_c \leftarrow \text{Commit}(cs) \]
2. TPM chooses nonce $N_{TPM}$ and signs
   \[ \sigma_{TPM} \leftarrow \text{sign}_{TPM}(N_V, N_{TPM}, \text{com}_c) \]
3. H blinds certificate $\text{cert}_C$ and builds proof
   \[ \text{cert}_{CI}^* \leftarrow \text{Blind}(\text{cert}_C), \text{proof} \]

$C_H$ current configuration of host $H$
$cs := \text{Hash}(C_H)$
$p$ (security) property of $C_H$
$\text{cert}_C := \text{sign}_C(p, cs)$

non-interactive zero-knowledge proof of
Cryptographic Proof III

**Trusted Platform**

- Cert_{CI} is kept secret from V

**Verifier V**

- Choose nonce N_V
- Verify: cert_{CI}^*, σ_{TPM}, proof

**Attestor A_p**

- Verify string proof
  - Is a signature Proof of Knowledge (SPK)
  - (Non-interactive Zero-Knowledge Proof of Knowledge)
  - Proves that the content cs of commitment com_cs is in cert_{CI}^* without revealing cs

C_H current configuration of host H
p (security) property of C_H

\[
\begin{align*}
\text{cs} & := \text{Hash}( C_H ) \\
\text{cert}_{CI} & := \text{sign}_{CI}( p, \text{cs} )
\end{align*}
\]
Exploring Other Approaches

• **Code control**
  - Property attester is trusted to enforce that a machine can only behave as expected
  - e.g., reference monitor to attest both OS and the enforced security policy (e.g., [MaSmBaSt2004] for SE Linux [LoSm2001])

• **Code analysis**
  - Property attester derives the machine’s properties or verifies proof of properties
  - e.g., proof-carrying codes and semantic code analysis (e.g., [Necu2002], [HaChFr2003])
Mapping Properties to PCRs

• Requirement
  • reuse existing mechanisms based on PCRs
    • Remote Attestation
    • Sealing, Binding

• Solution
  • Use PCRs to store properties instead of binary hash values
  • Instead of extending PCR with $H(S)$:
    • 1) Extend PCR$_i$ with $H(PK_T)$
    • 2) Find $C = cert_T(p, S)$
    • 3) Validate signature in $C$
    • 4) If valid, extend PCR$_j$ with $p$, otherwise by $0^{160}$
PBA with Bootloader I

- BIOS measures boot loader and extends PCRs (binary measurement)
- For every module to be loaded by the boot loader
  1) Extend PCR_{b+i} with H(\text{PK}_T )
  2) Find C = \text{cert}_T(p,S)
  3) Validate signature in C
  4) If valid, extend PCR_{b+j} with p, otherwise by 0^{160}
- PCR_{b+1}...PCR_{23} represent properties \textit{p certified by } T
- Attestation, Binding, and Sealing can be used as usual
- After a software update, only a new
Solution with Bootloader II

**Execution**

- **Application**
- **Operating System (OS)**
- **Boot Loader (BL)**
  - certificate list \((m_A, p_A)\)
  - 1. measures OS / applications
  - 2. gets corresponding properties from cert. list
  - 3. reports properties to TPM
  - 4. loads OS / properties

**Hardware**

- **CPU**
- **BIOS**
- **CRTM**
- **TPM**

**Verification**

**Remote Verifier**

- Is able to verify claimed properties of the platform according to the digest authentically reported by the TPM

**Certificate Issuer Cl**

- **issues property certificate** \(\text{cert}_{CI}\)

**Measurement**

- **BL measures properties of OS and Applications**
- **BL measures**
- **BIOS measures BL**
- **CTRM measures BIOS**

- **PCR[15]**
- **PCR[14]**
- **PCR[1]**
- **PCR[0]**

- **m_{BL}**, **m_{BIOS}**, **p_{k_{CI}}**, **m_{pk}**, **p_{OS}, p_{App}**

**Trusted components**
Virtual TPM (vTPM)
Overview

• Enables virtual machines (VM) on a single hardware platform to use the same physical TPM
• Full software implementation of the TPM specification with additional functionalities to manage virtual TPM (vTPM) instances
Requirements on vTPM I

- Confidentiality and integrity of vTPM state
  - it includes Endorsement Key (EK), the Storage Root Key (SRK), the owner's authorization data (i.e., password), and monotonic counters

- Secure link to chain of trust
  - Linkage between hardware and software TPM

- Unclonability and Secure Migration

- Freshness

- Distinguishability of hardware and software TPM
Requirements on vTPM II

• Data availability
  • Sealed data must be accessible if the security policy is fulfilled, this includes migration

• Privacy protection
  • User can decide over the information disclosure on platform’s concrete configuration

• Flexible key types
Existing Solutions on vTPM and Open Problems
Possible vTPM Architecture

vTPM Manager
- creates vTPM instances
- multiplexes requests from virtual machines to their associated vTPM instances

Linking vTPM & TCB
- Lower sets of vPCRs contains values from PCR
- Upper set of vPCRs contain values specific to vTPM

[Berger et al 06]

request/response path between vTPM-Manager, vTPMs and the hardware TPM
Linking vTPM to TPM

- Certified AIK signs vEK and vAIK is certified by PrivacyCA
  - Conform to TCG model but needs communication with Privacy CA
- Certified AIK signs vAIK
  - Not standard
- In above solutions AIKs must be invalidated once VM is resumed on the target platform
- Alternative: rely on a local authority to issue certificates for EKs of vTPM instances
  - No direct connection to the TPM
  - But the service can be attested
Migration of vTPMs

1. encrypt source vTPM instance using a symmetric key K
2. compute digest of source vTPM instance
3. encrypt key sk of hardware TPM
4. lock vTPM instance
5. delete vTPM instance
6. migrate sk according to TCG
7. create new vTPM instance
8. recover vTPM instance
9. unlock vTPM instance

• decrypt K using previously migrated sk
• decrypt source vTPM instance using K
• verify digest of source vTPM instance
Problems Not Solved

• Migration to platforms with different binary VMM
  • Binary-based integrity measurement
  • Problems with mapping lower PCRs to lower vPCRs

• Large TCB
  • VMM, vTPM Manager, vTPM instances
  • and a whole guest OS in one VM

• Only one type of TPM keys supported

• Software Updates?
Endorsement Credentials for vTPMs

Information to be included into the Endorsement Credential (EC) for vTPMs:

- certInfo( EK_vTPM ) ← ( credType, pk_{EK_vTPM}, tpmModel, certIssuer, tpmVer, ml )
- Information to be included into the Endorsement Credential (EC) for vTPM
  - public Endorsement Key of vTPM
  - identifier for vTPM (manufacturer-specific)
  - identifies issuer of the EC
  - revision of TPM spec. the vTPM implements
  - measurement list of the environment of the VM

Certification of vTPM’s EK using an AIK and the TPM_Quote command:

\[ N \leftarrow \text{SHA-1}( \text{certInfo}( EK_{vTPM} ) ) \]

TCG intended \( N \) to act as anti-replay value (nonce)

\[ \sigma_{TPM} = \text{sign}_{AIK}( PCR[S_{PCR}], N ) \leftarrow \text{TPM_Quote}( h_{AIK}, A_{AIK}, S_{PCR}, N ) \]

- pointer to the AIK to be used to sign \( S_{PCR} \)
- authorization data for using the AIK
- selection of PCR registers which should be signed (e.g., those containing digests of environment of the VM)

Resulting EC for vTPM:

\[ \text{cert}_{AIK}( EK_{vTPM} ) \leftarrow ( \text{certInfo}( EK_{vTPM} ), \sigma_{TPM} ) \]
Flexible Key Generation and Usage

- Both hardware and software keys should be usable by a compartment
  - Example: Corporate computing at home
    - Boot802.1x (non-migratable) requires a hardware key
    - Other compartments (migratable) require software keys

- Security kernel should enforce privacy requirements
  - AIK/DAA management and usage by security service
    - Example: Service creates new AIK for every attestation procedure

- Decisions should be made by mandatory privacy and security policy!
Alternative Approach

The diagram illustrates a microkernel-based system with multiple virtual machines and client-side TPM drivers. The system includes components such as application layers, virtual machines, client-side TPM drivers, vTPM interfaces, key managers, storages, random number generators, counters, service interfaces, memory managers, I/O managers, and TPM drivers. The request/response paths are indicated by arrows connecting these components.
Possible Alternative Solution: Logical vTPM Architecture
Property Based Approach

Binary Measurement

TPM_Extend

h:= hash(binary)

h

vPCR

Property-based Measurement

TPM_Extend

h:= hash(binary)

h

vPCR

Each PropertyProvider maps the given hash value to a property certificate and stores the certificate in the vPCR (i.e., the hash of the certifying key).

c_a := propertyCert(h)

c_b := propertyCert(h)
Migration of vTPMs

Source vTPM

Migrate()

Source vTPM-Manager

InitiateMigration()

Target vTPM-Manager

Create()

Target vTPM

requestTrustedChannel()

(pk_{bind}, cert_{bind})

- verify (pk_{bind}, cert_{bind})
- sk := creatkey()
- esk := bind[pk_{bind}](sk)
- s := getState()
- es := encrypt[sk](s)
- deleteKey(sk), deleteState()

transfer(es, esk)

- destroy()

- sk := unbind[sk_{bind}](esk)
- s := decrypt[sk](es)
- setState(s)
Conclusion and Future Work

- Trusted Computing is an emerging technology, however, many challenges remain:
  - Property-based Attestation
    - What are useful properties? How to formalize them? What are the most efficient approaches for property-based Attestation in practice?
  - TPM
    - How to design a minimum TPM? How to detect trapdoors and Trojans? More effective revocation and maintenance mechanisms
  - Privacy protecting and anti-discriminating designs
  - Enabling security protocols with TC functionalities
  - Trusted Virtual Domain
  - Applications of multiparty computation for real world
  - Lightweight TC for embedded systems
  - Formal proofs in security models