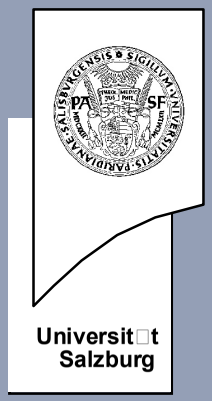


Terra:

A Virtual Machine-Based Platform for Trusted Computing

by Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum and Dan Boneh

Gerald Stieglbauer

Gerald.Stieglbauer@cs.uni-salzburg.at
Universität Salzburg

SOFTWARE RESEARCH LAB

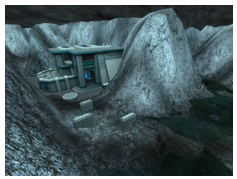
Content

- Introduction
- What is Terra?
- How does Terra work?
- Prototype implementation
- Conclusion

Facts

- The variety of applications:

- Secure bank application



- Online game

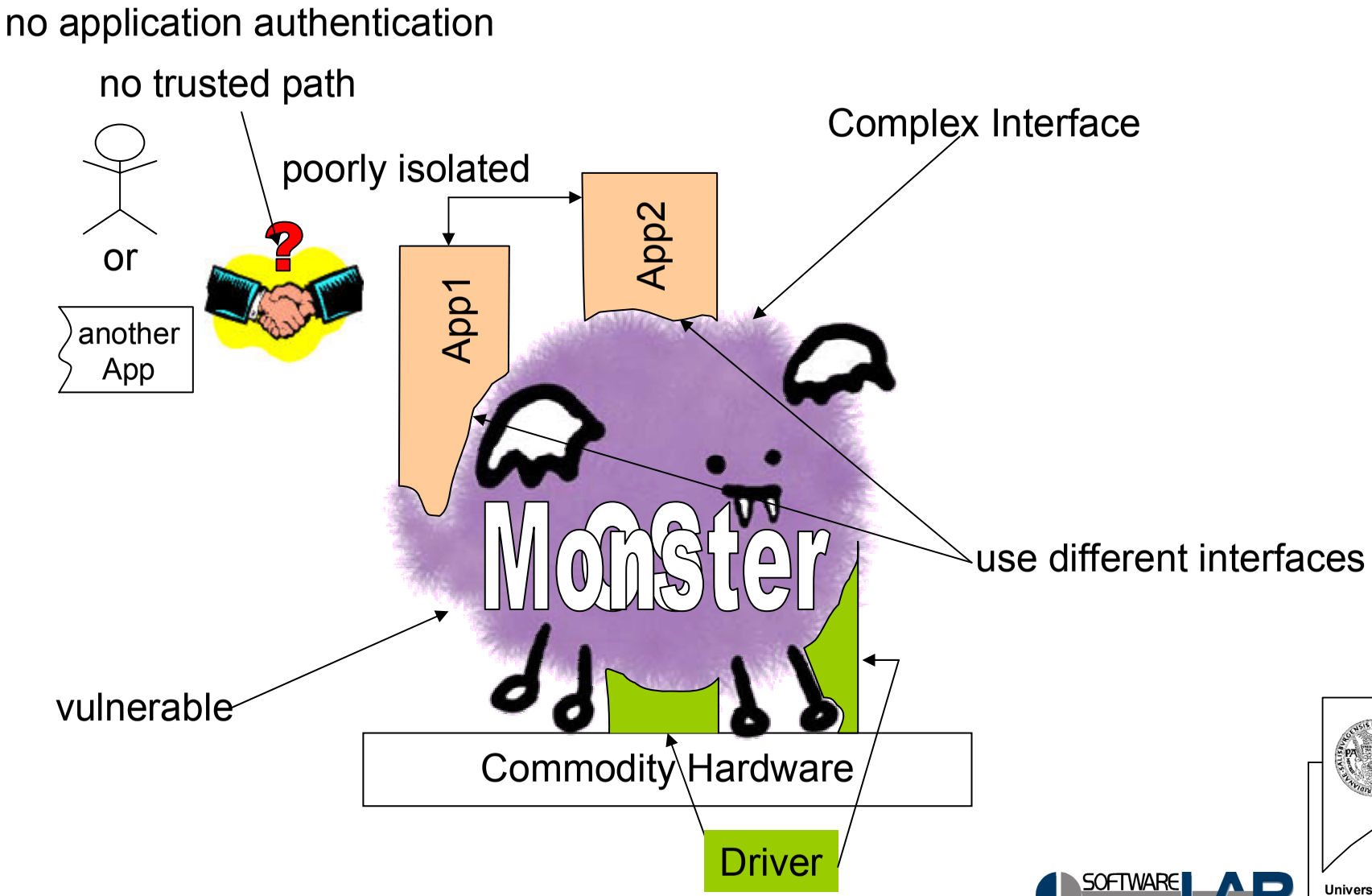
- Web application

different security requirements

SOFTWARE RESEARCH LAB

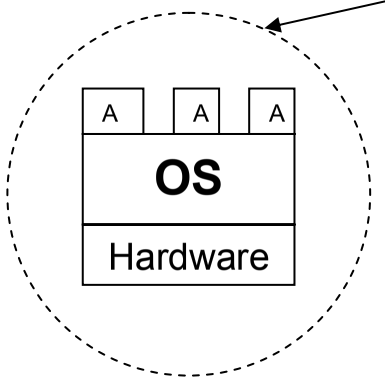
Universität Salzburg

Common situation



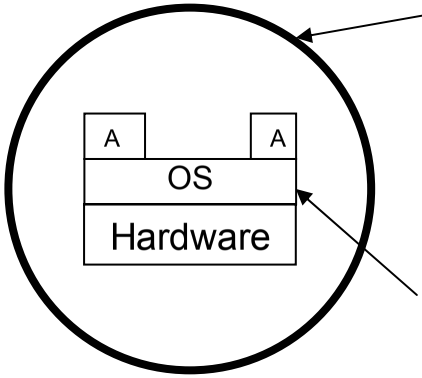
Open vs. Closed Systems (I)

Open system, e.g.



→ Plattform owner has full control

Closed System, e.g.



‚thinner', more application tailored OS

→ the developer has full control

SOFTWARE RESEARCH LAB

Universität Salzburg

Open vs. Closed Systems (II)

- Open Systems
  - Support for many application
  - Huge amount of existing code
  - Take advantage of commodity hardware

- Close Systems
  - Slimmer, more specialized OS
  - Applications are tailored to their security requirements
  - Provide hardware tamper resistance

What is Terra?

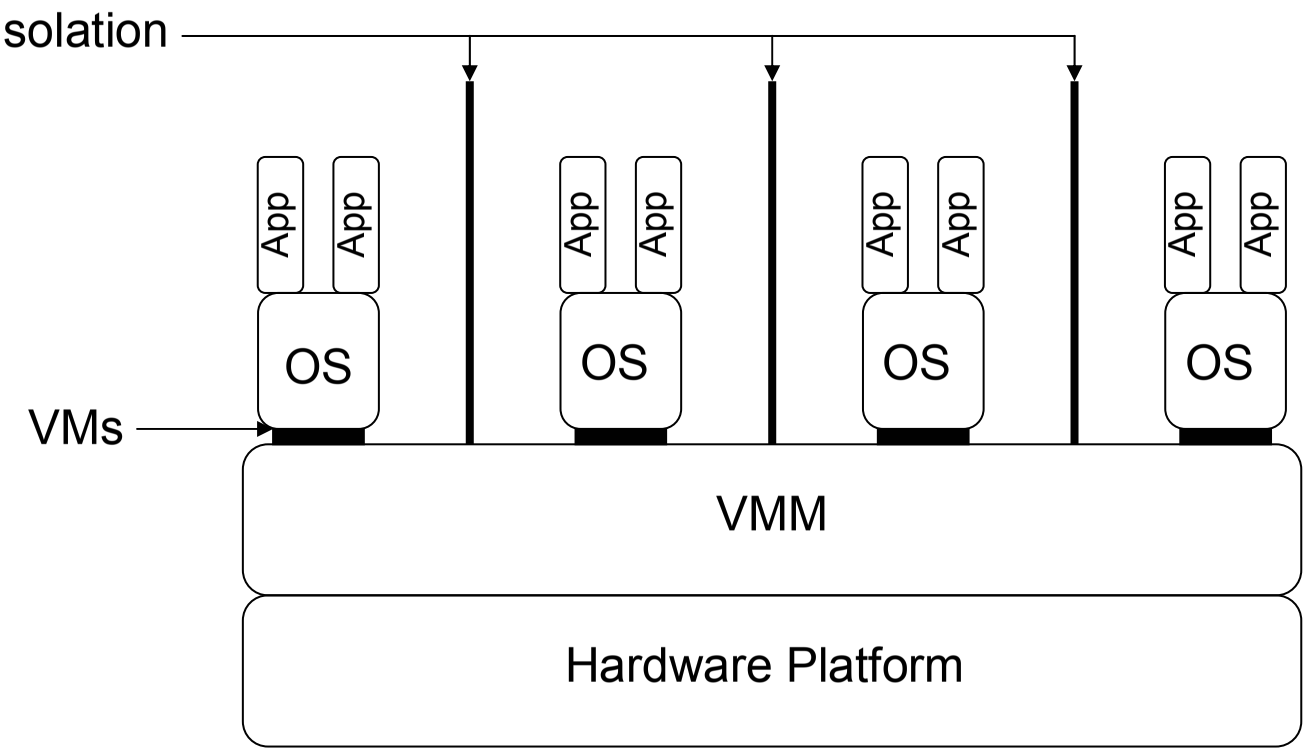
What does Terra try to achieve?

- *Terra* tries to combine the advantages of an Open and a Closed box.
- Terra implements two principles:
  - *Isolation*
  - *Attestation*

  of applications

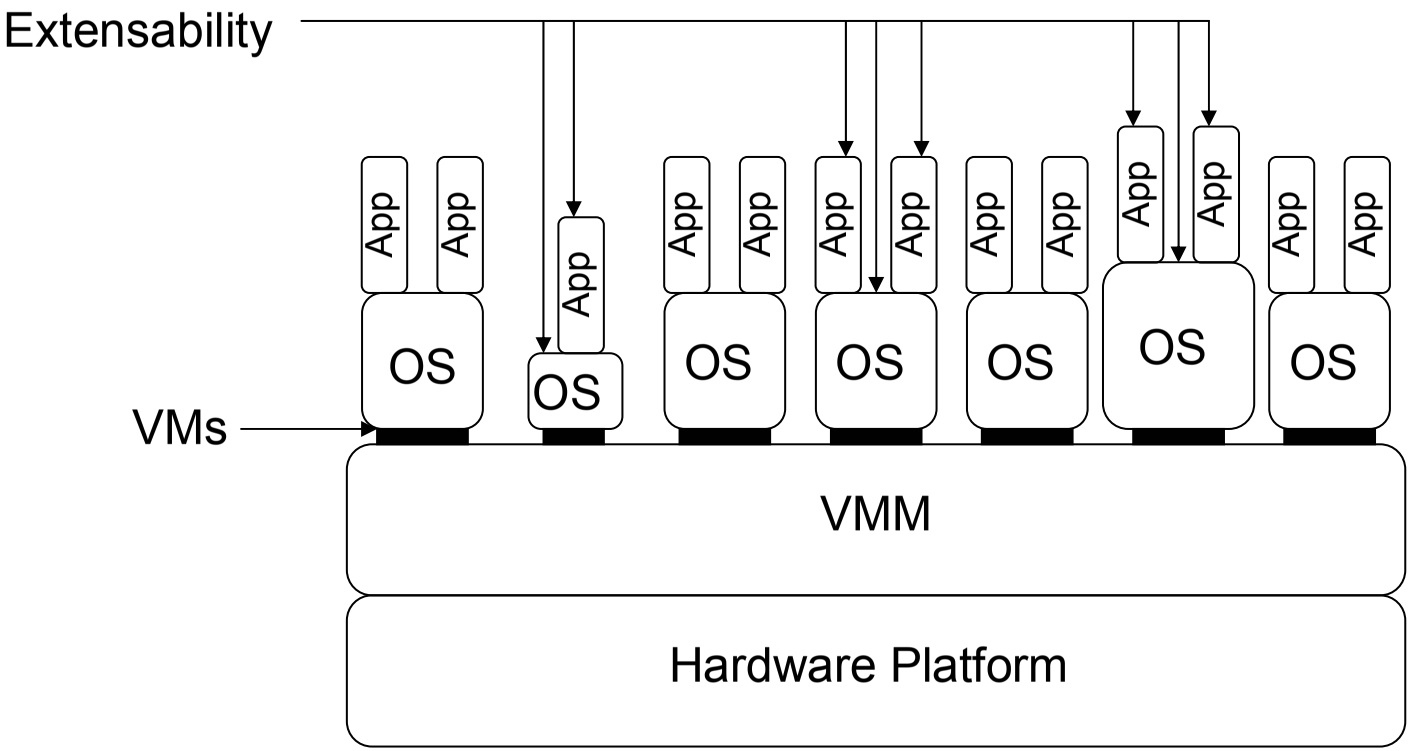
What is Terra?

- The heart of *Terra* is a *Virtual Machine Monitor* (VMM)



What is Terra?

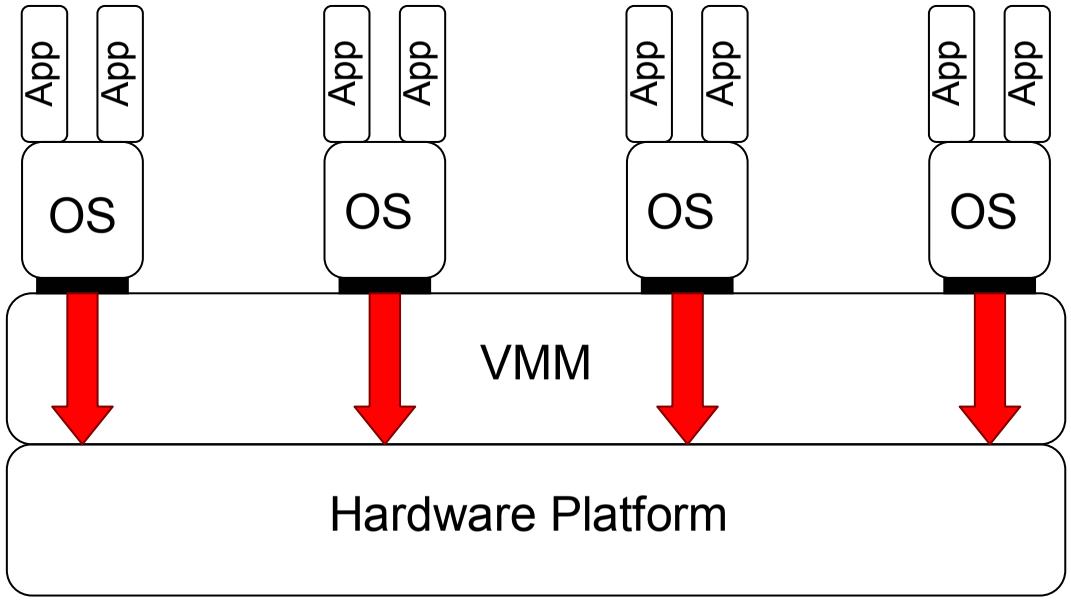
- The heart of *Terra* is a *Virtual Machine Monitor* (VMM)



What is Terra?

- The heart of *Terra* is a *Virtual Machine Monitor* (VMM)

Efficiency

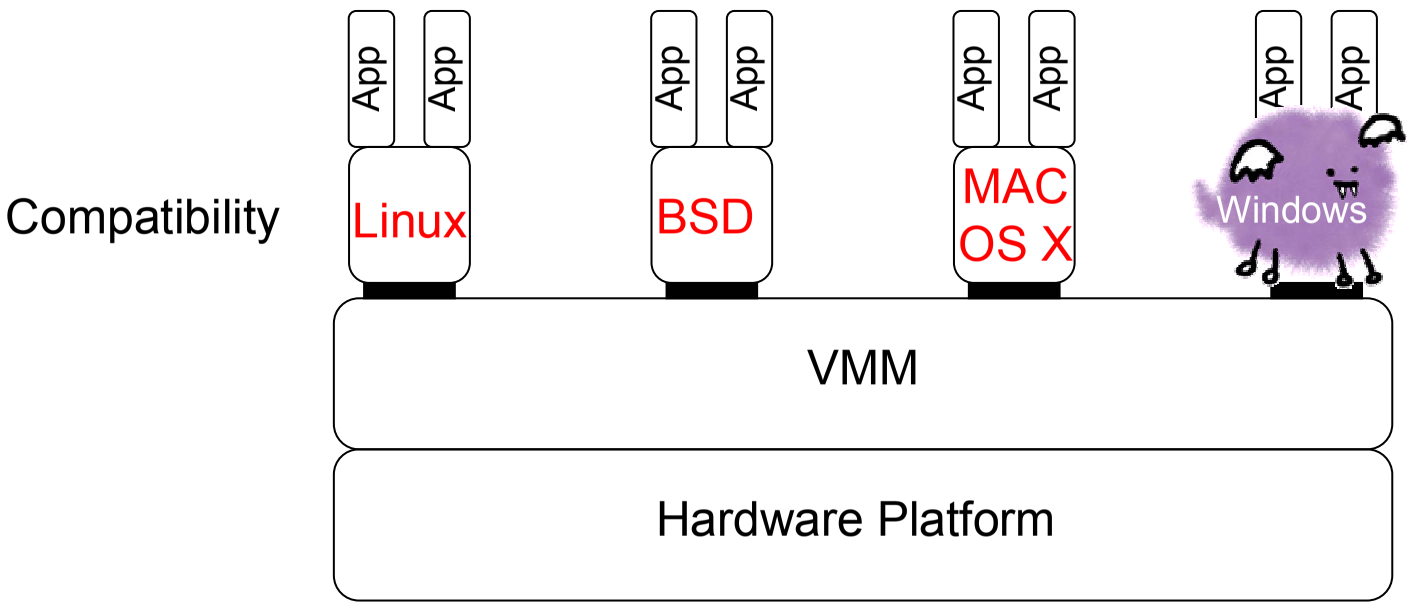


SOFTWARE RESEARCH LAB

Universität Salzburg

What is Terra?

- The heart of *Terra* is a *Virtual Machine Monitor* (VMM)

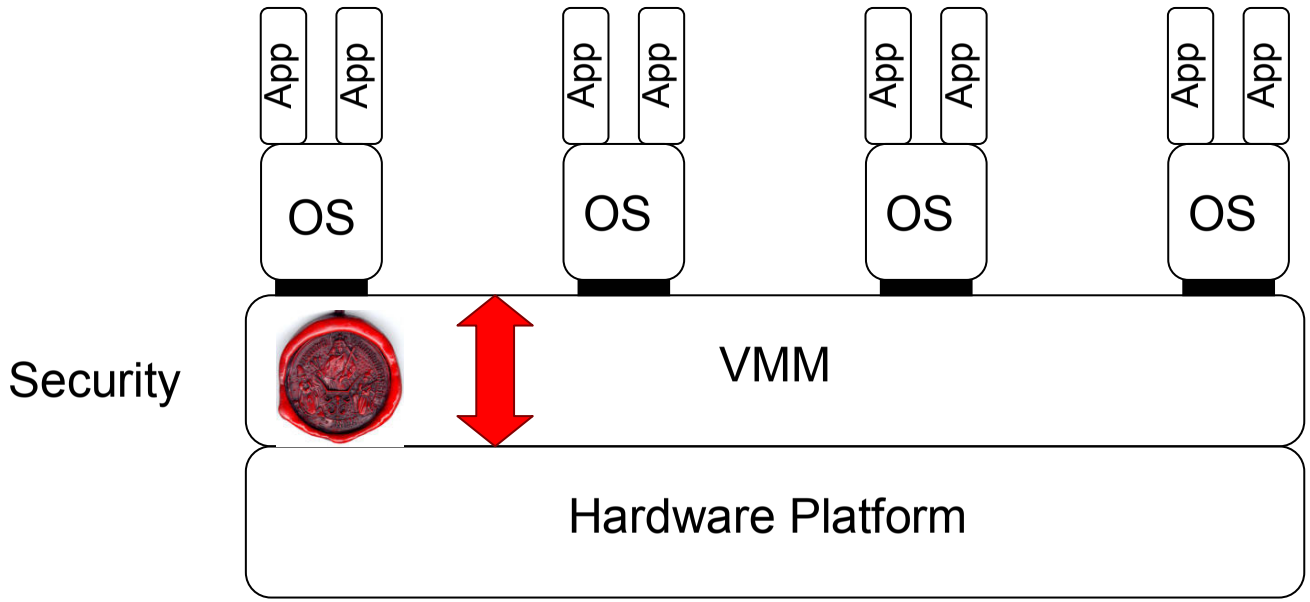


SOFTWARE RESEARCH LAB

Universität Salzburg

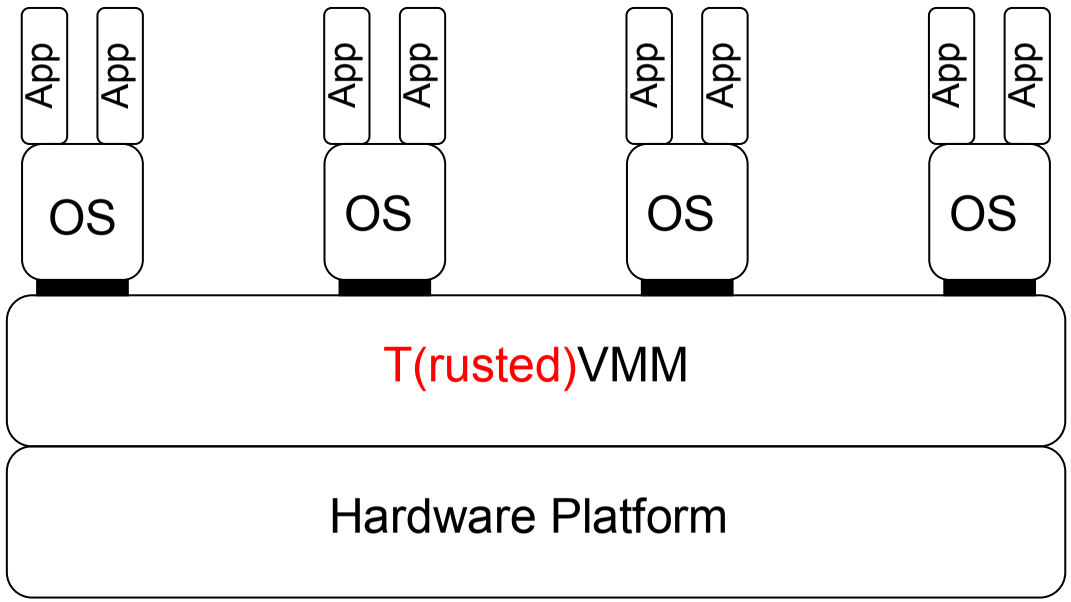
What is Terra?

- The heart of *Terra* is a *Virtual Machine Monitor* (VMM)



What is Terra?

- The heart of *Terra* is a *Trusted Virtual Machine Monitor* (TVMM)

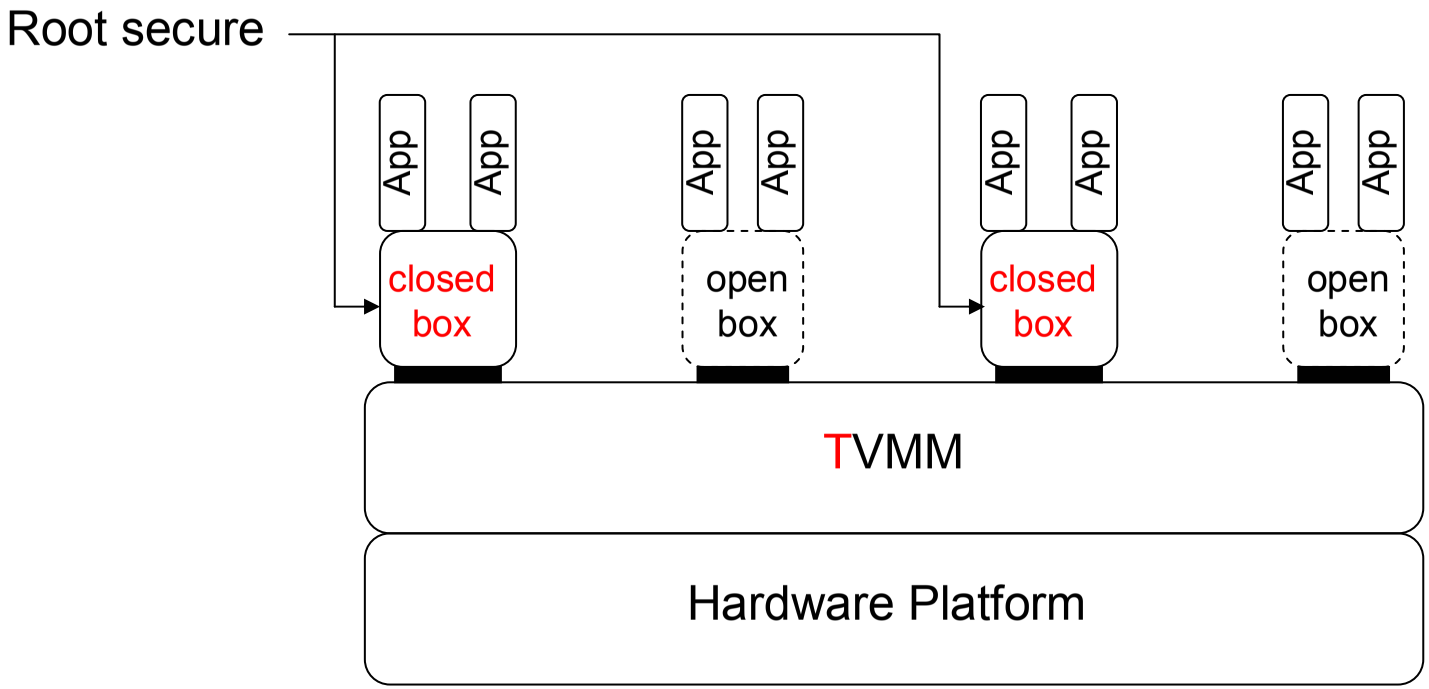


SOFTWARE RESEARCH LAB

Universität Salzburg

What is Terra?

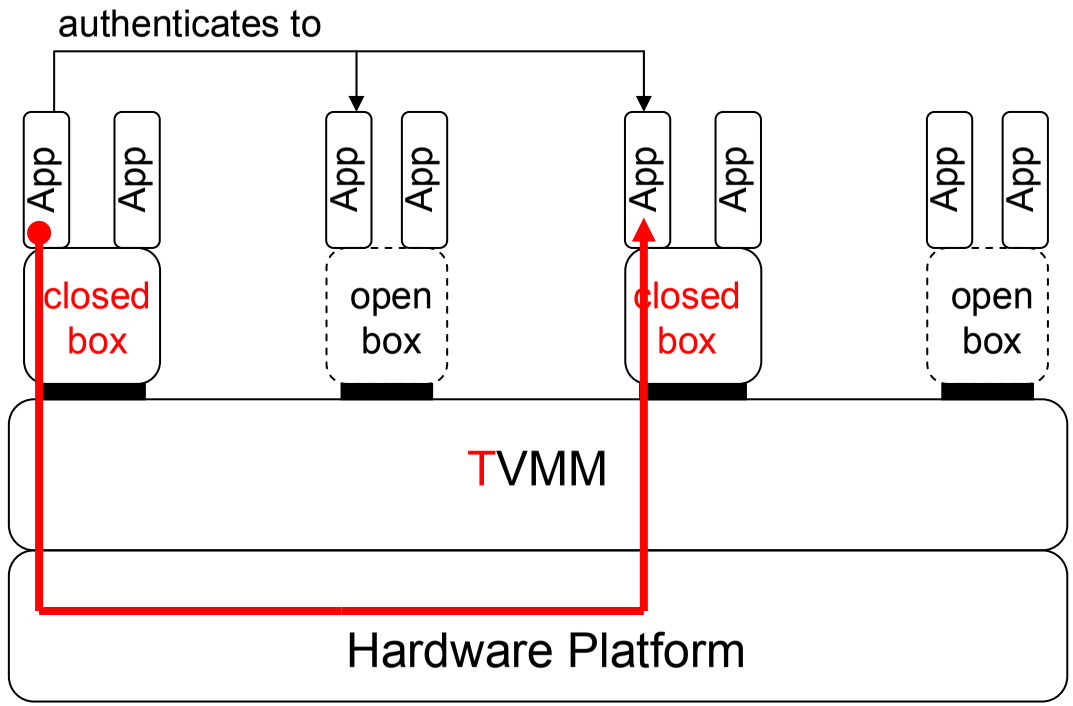
- The heart of *Terra* is a *Trusted Virtual Machine Monitor* (TVMM)



What is Terra?

- The heart of *Terra* is a *Trusted Virtual Machine Monitor* (TVMM)

Attestation

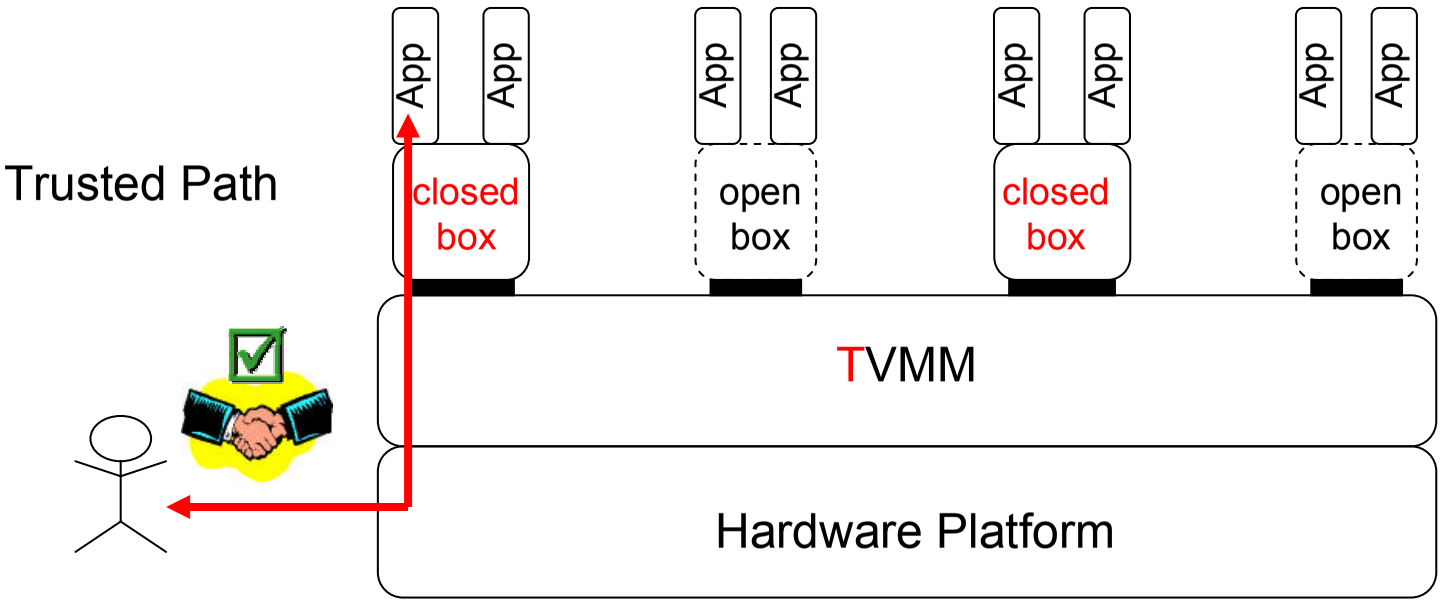


SOFTWARE RESEARCH LAB

Universität Salzburg

What is Terra?

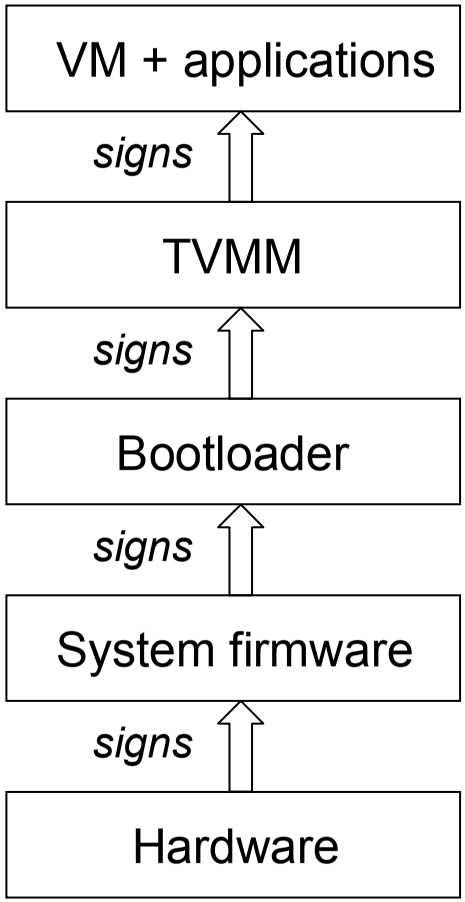
- The heart of *Terra* is a *Trusted Virtual Machine Monitor* (TVMM)



How does Terra work?

Attestation (I)

- Building a certificate chain

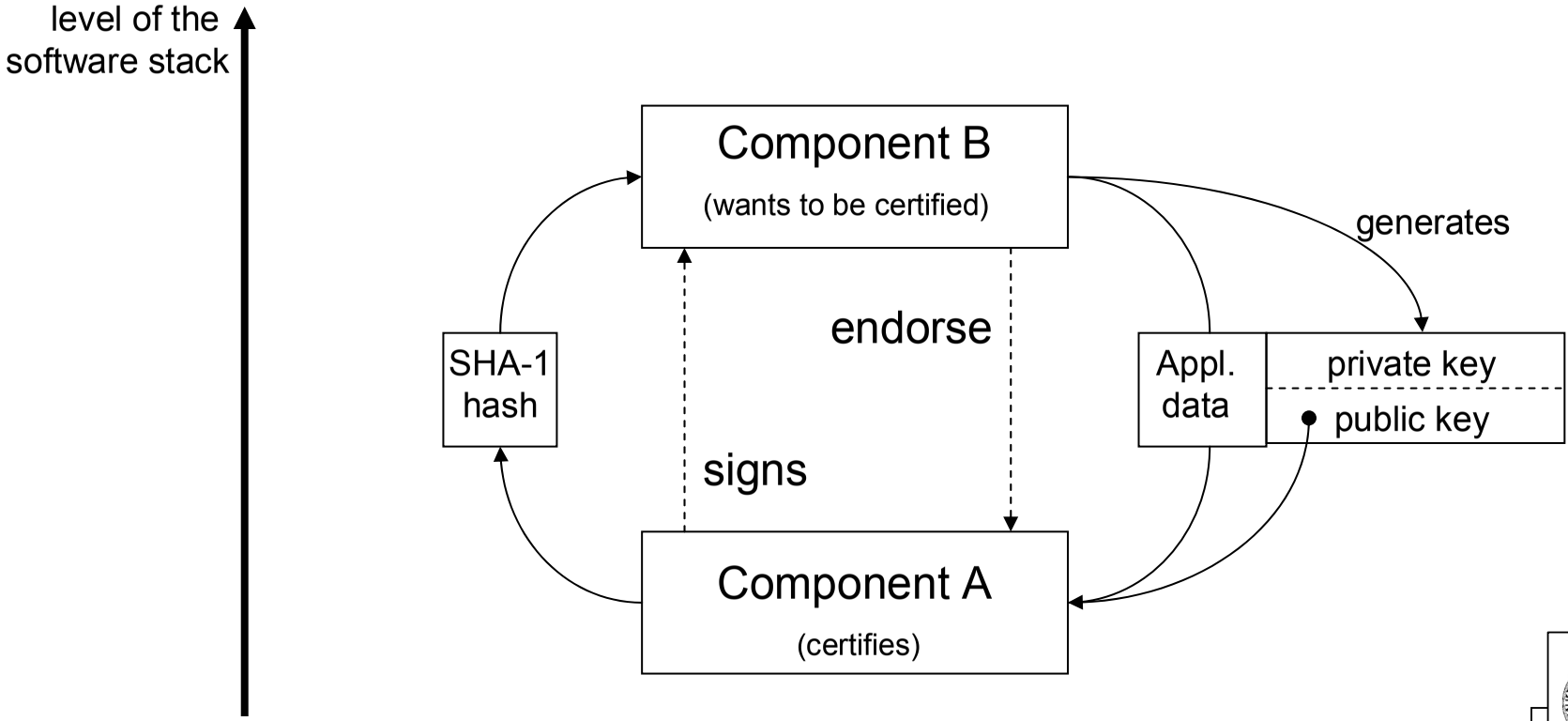


e.g. BIOS

Tamper resistant chip (e.g. TCPA)

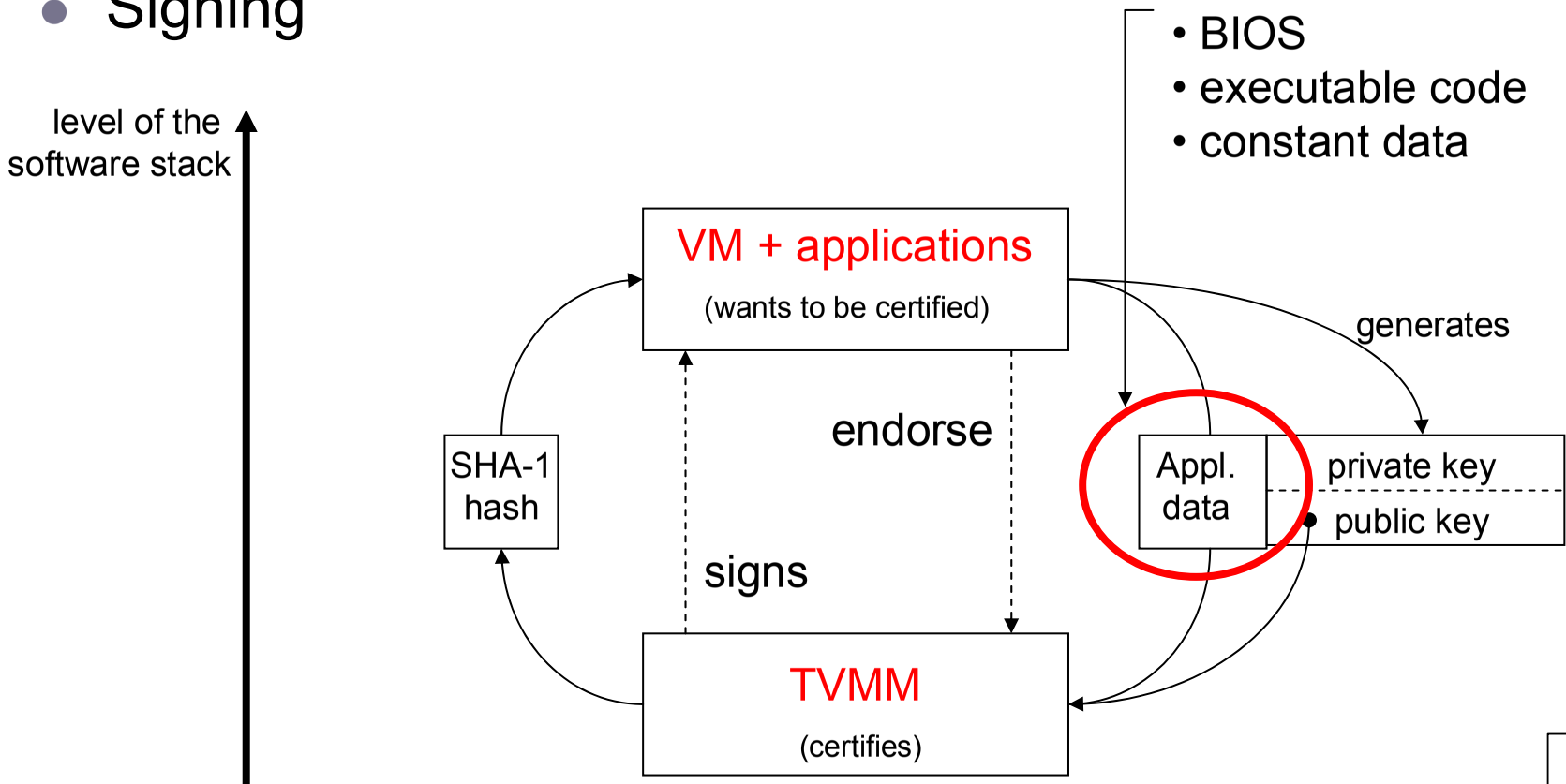
Attestation (II)

- Signing



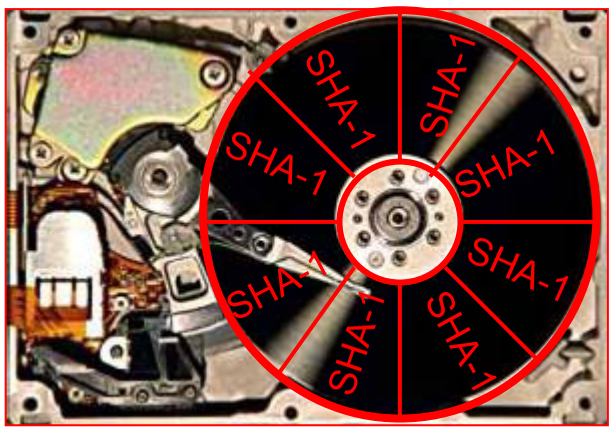
Attestation (III)

- Signing



Implementing Attestation (I)

- A VM image consists of several parts (entities
  - Mutable (NVRAM, cache, user data)
  - Immutable (BIOS, executables, constant data)
- List of hashes for attestable parts
- e.g. an attestable (virtual) disk



SHA-1
VM
descriptor

Implementing Attestation (II)

- Example:

  Hashing a 4 GB entity into 20-byte SHA-1 with a 4 kB block size

  ➔ *20 MB of hashes*

  ➔ *have to be verified against the VM descriptor hash*

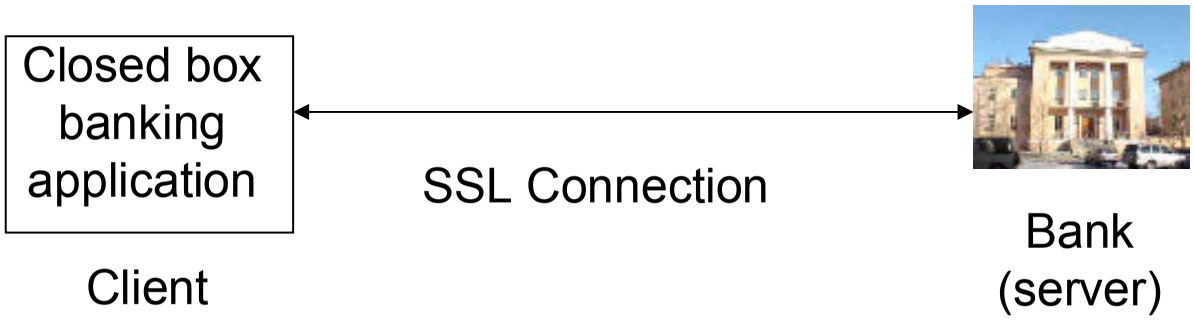
  ➔ *problem regarding memory and time*

Implementing Attestation (III)

- Ahead-of-Time Attestation
  - All components are proved during the boot process
  - Used only for small, high-assurance VMs

- Optimistic Attestation
  - If Ahead-of-Time is impractical
  - Individual blocks are „lazy“ checked

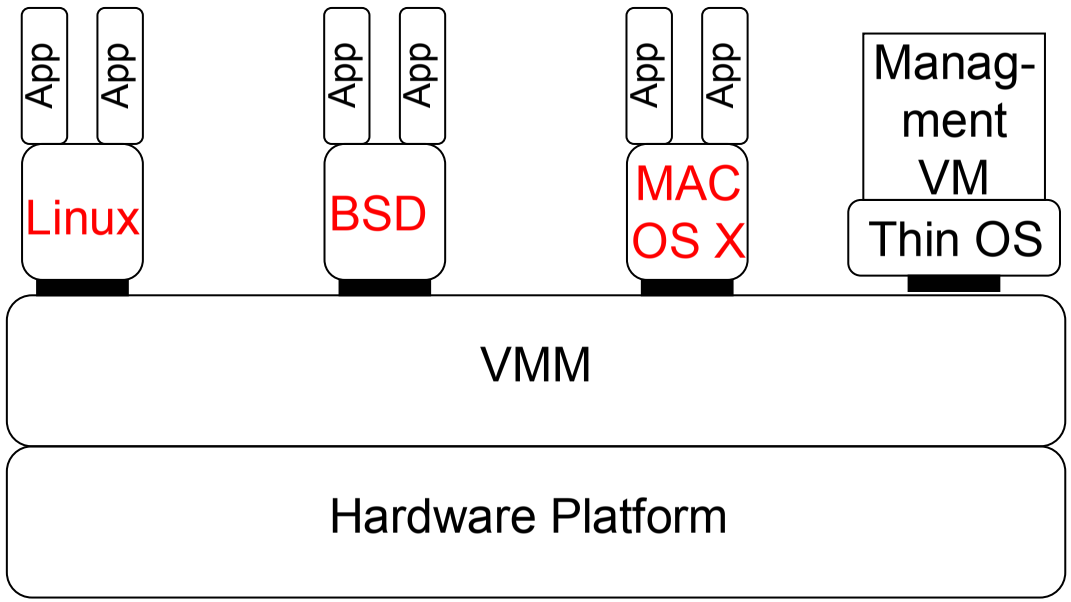
Example attestation



- Certificate is exchaned during SSL handshake
- The server verifies
  - lowest certification (hardware)
  - middle certification chain (BIOS, bootloader, TVMM)
  - highes certification (application, e.g. Quicken)

Management VM (I)

- High-Level configuration with the Management VM



Management VM (II)

- API (i)
  - device-id ← CREATE-DEVICE(type, params)
  - CONNECT(device-id-1, device-id-2)
  - DISCONNECT(…)
  - vm-id ← CREATE-VM(config)

Management VM (III)

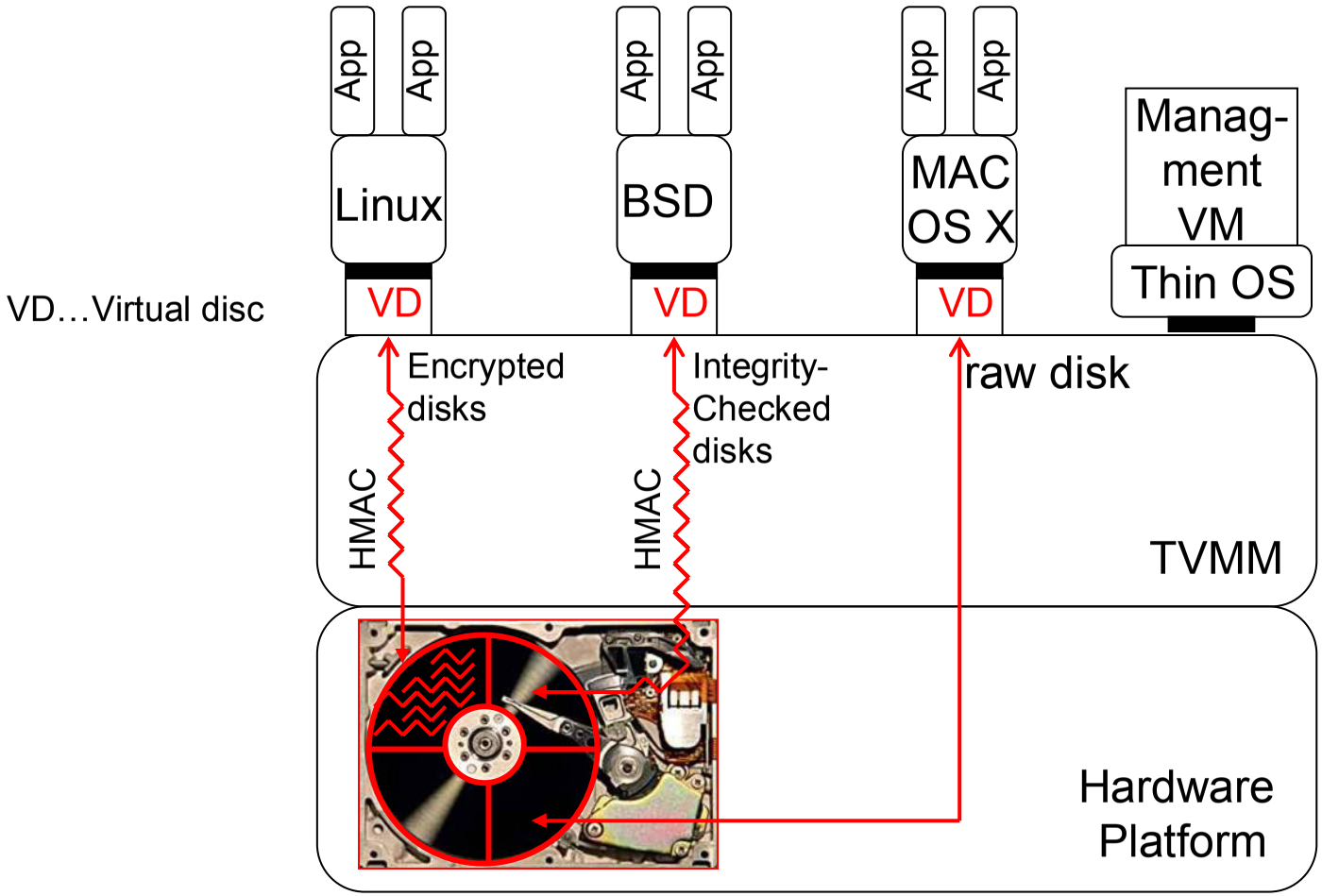
- API (ii)
  - ATTACH(vm-id, device-id)
  - DETACH(…)

  - ON(vm-id)
  - OFF(vm-id)

  - SUSPEND(vm-id)
  - RESUME(vm-id)

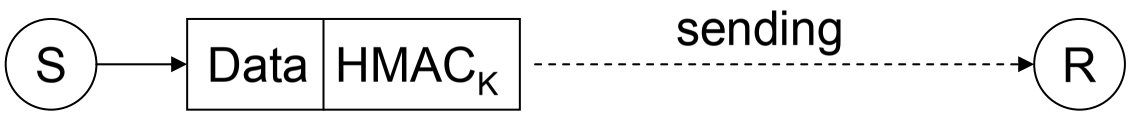
Storage Interface

- Three different types of disk access control



The HMAC Construction

- MAC – Messaging Authentication Code
- HMAC – MAC using hashes
- Using an authentication tag $HMAC_K$



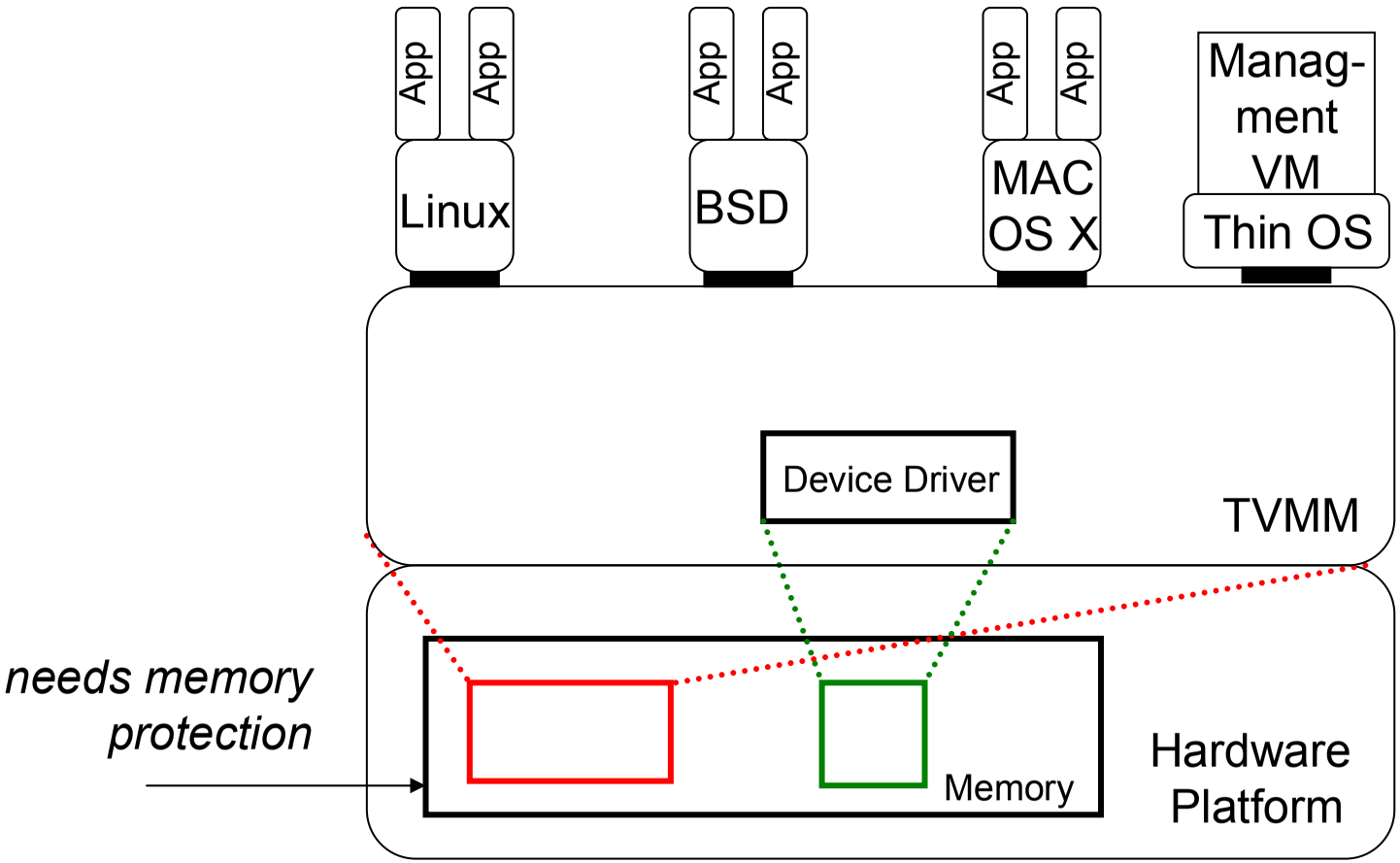
- $HMAC_K = H(K, H(K, Data))$ (H…SHA-1, K…Key)

Device Driver Security (I)

- Today´s driver can be very „large“
  (e.g. video, modem, wireless network driver)
- Worst quality code in many cases
- ➔ Device driver as part of the *TVMM´s trusted computing base*?

Device Driver Security (II)

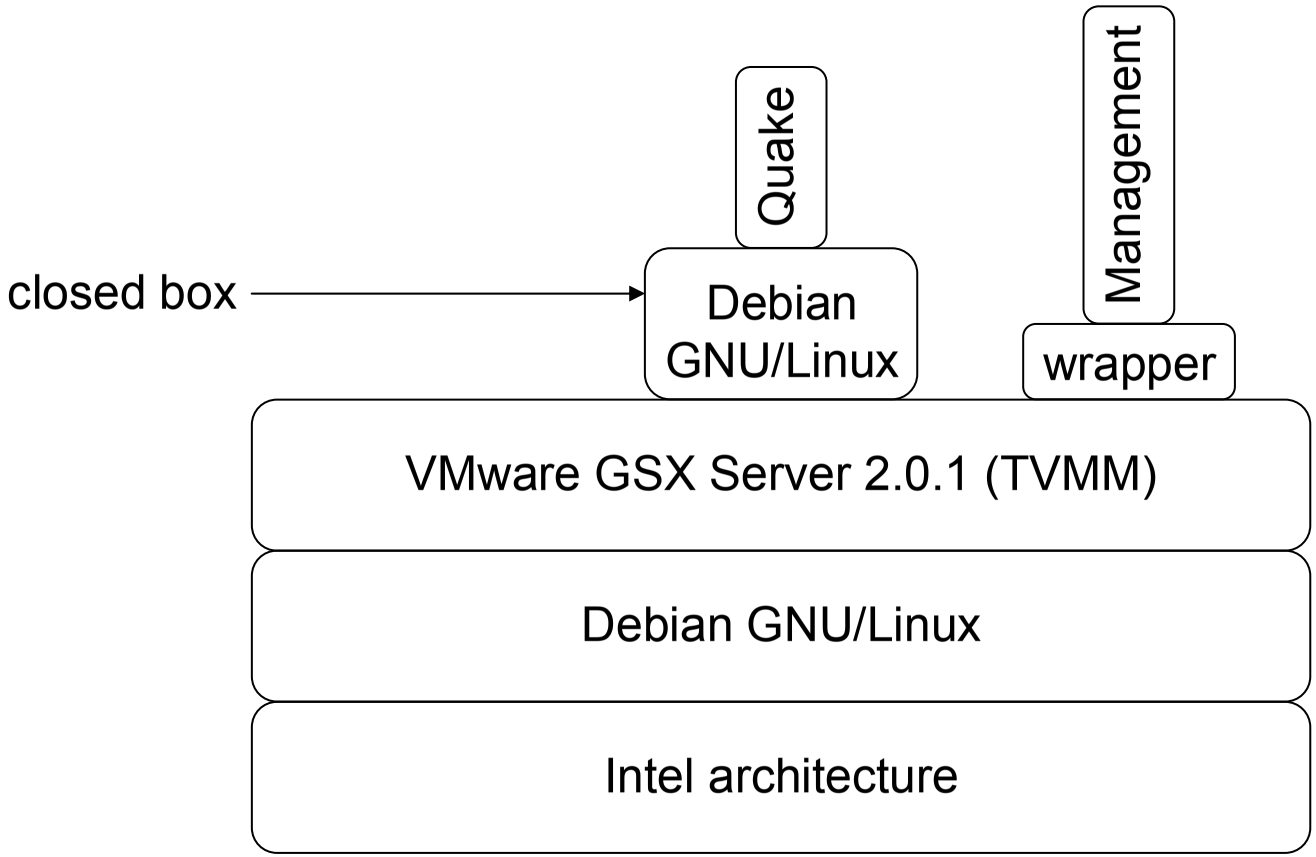
- High level configuration management



Prototype implementation

Prototype implementation (I)

- Used Architecture



Prototype implementation (II)

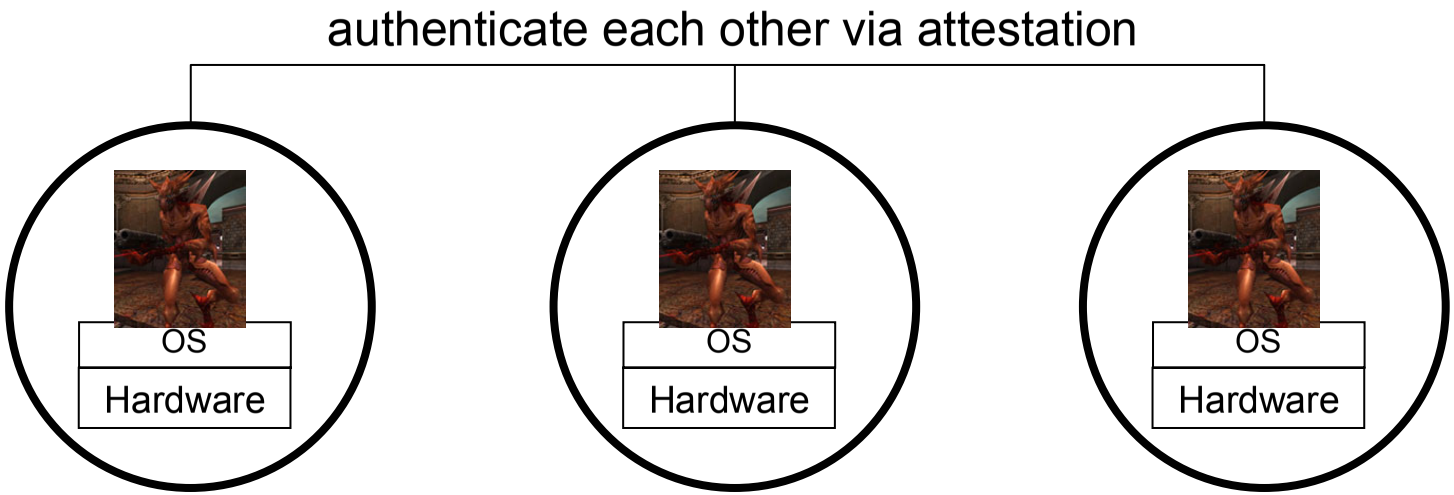
- Secure storage
  - Using a dynamic preloaded library
  - Library implements
    - Ahead-of-time attestation
    - Optimistic attestation
    - Integry-checked storage
    - Encrypted storage

Prototype implementation (III)

- System management
  - VMware GSX Server provides configuration interface
  - Python wrapper implement the management API
  - OpenSSL library for certification management

Prototype implementation (IV)

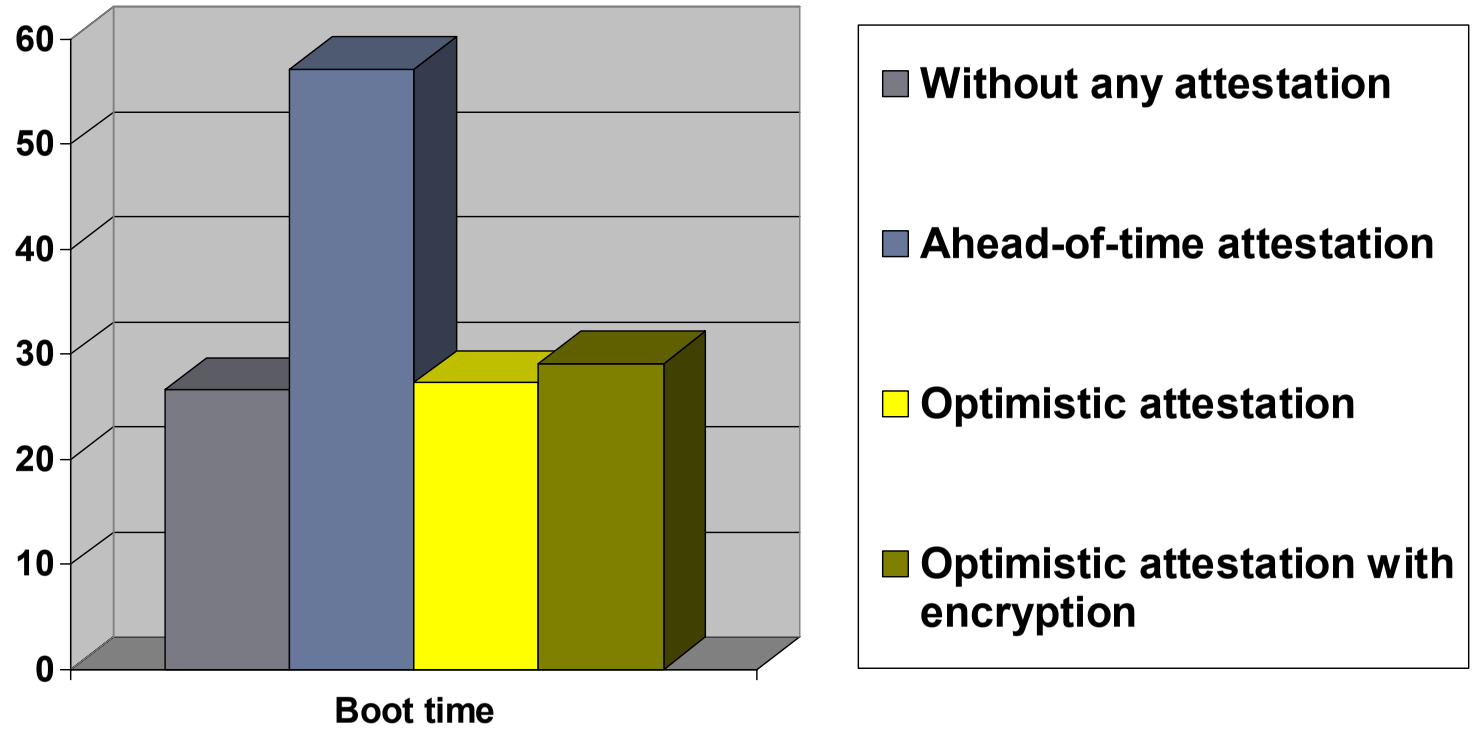
- Trusted Quake



- Linux boots directly into quake
- Attestation via dynamic preloaded library
- Using 160-bit SHA-1 HMAC and 56-bit DES keys

Prototype implementation (V)

- Measurement results



Conclusion

- Flexible architecture for trusted computing
- Allowing open box and closed box VMs
- Attestation and Isolation as basis for application authenticaion
- Application security can be tailored to their needs
- Still some „new architectural environment“ is needed