Semantic Remote Attestation: A Virtual Machine Directed Approach to Trusted Computing

Vivek Haldar
Deepak Chandra
Michael Franz

Information and Computer Science
University of California, Irvine

Outline

- Quick Trusted Computing primer...
- ... and what's wrong with it
- How ***virtual machines*** can make Trusted Computing more “secure”, flexible and effective

Background: Gaining Trust

- ***Trust*** – assurance that system is doing what I want it to do, properly
- Kinds of systems:
  - ***Closed:*** trusted because of a trustworthy manufacturer, and closed interface – e.g. Game consoles, ATMs
  - ***Open***: not trusted at all, no central arbiter – e.g. PCs, PDAs
- *Question: can we bring the notion of "trust" to open systems?*

Enter: Trusted Computing

- Trust = Integrity + Authenticity
- ***Integrity***: the program/system was not changed or tampered with
- ***Authenticity:*** verifying credentials, or identity, of a person, thing, or program
- Trusted Computing: add components and mechanisms to open systems to provide “trust”
  - This component is a *Trusted Platform Module (TPM)* that is tamper-resistant and has an embedded private key

Integrity: Secure Boot Process

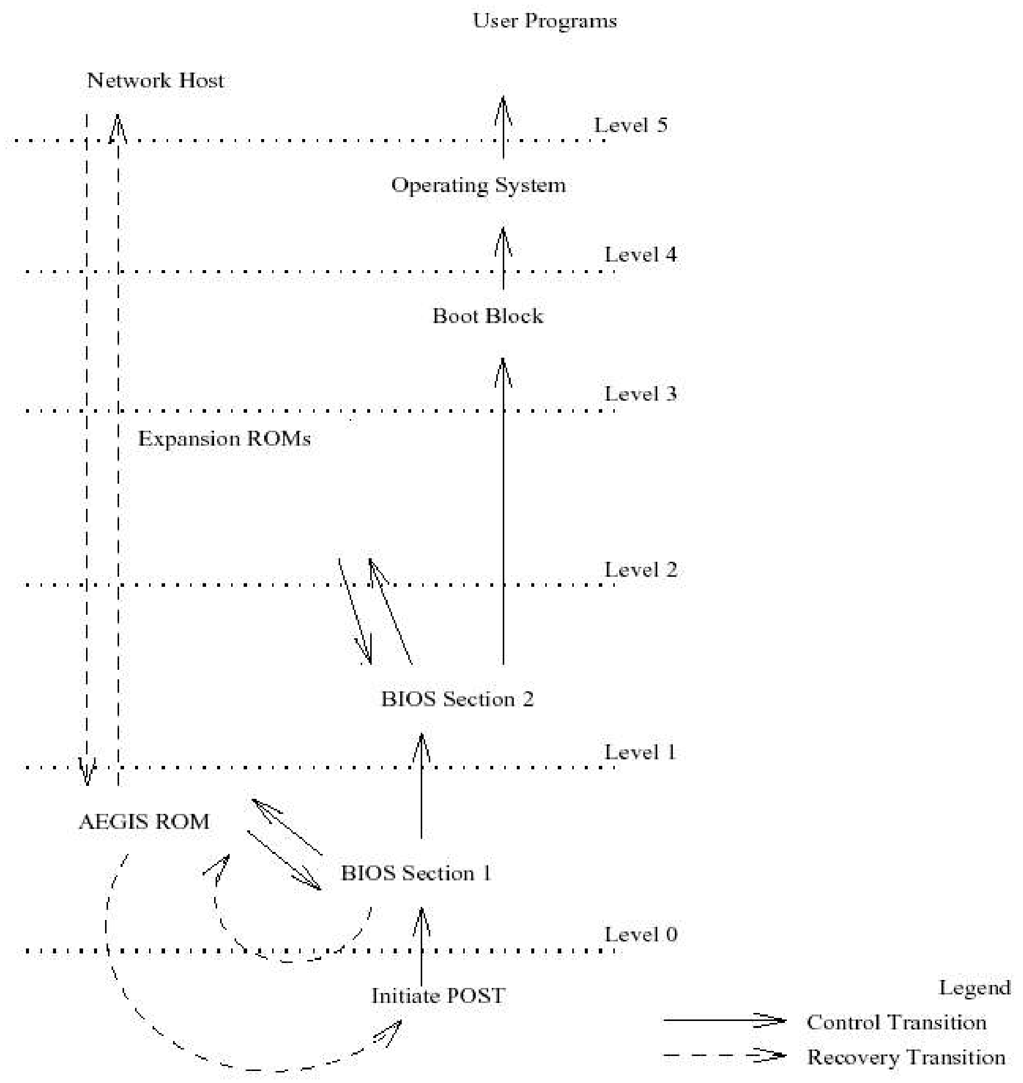


Diagram by Arbaugh et al

- **Goal**: make sure you're only booting off “trusted” software
- At every level:
- Compute hash of level above
- Compare with stored signature of hash of original software
- *Then* transfer control

Authenticity: Remote Attestation

- **Goal:** a *remote party* wants to know what software I'm running before it'll talk to me
- It asks me for my *integrity metrics*
- The trusted module signs my integrity metrics, which I send to the remote party
- The remote party verifies the digital signature
- This integrity metric is just the signed *hash of the executable image!*

Problems with remote attestation

- Says nothing about program behavior
- Static, inexpressive and inflexible
- Upgrades and patches?
- Heterogeneity of devices/platforms?
- Revocation

Program behavior not attested

- Remote attestation certifies ***what particular binary*** is running on a remote machine
- Attested programs can be insecure, or have bugs
- Most vendors will attest their programs regardless
- Assurances about program behavior – “because vendor says so, and he signed it”
- ***Trust vs Security***
  - ***Trust*** – identity, certification
  - ***Security*** – behavior, verification, enforcement

Remote attestation is static

- Cannot convey ***dynamic information***
  - Runtime state of program
  - Properties of program input
- One time operation done at the beginning of a network transaction/protocol

Upgrades and patches?

- Verifier needs an “approved list” of software
- Most software has a steady stream of patches and upgrades
  - Can be applied in any order, some may not be applied
  - Exponential blowup in “version” space
- Creates problems at both ends of the network
  - Servers need to manage intractable list
  - Clients may need to hold off on upgrades/patches

Accommodating platform heterogeneity

- Wide variety of computing platforms
  - Popularity of cross-platform solutions like Java and .NET
- Standard remote attestation certifies ***specific binaries***
- Just as with patches and upgrades, intractable to manage programs across various platforms

Some questions

- How to remotely attest ***relevant program behavior***, while allowing a ***range of implementations***?
- Stuck with lop-sided network model
  - A lot (most?) ***work*** done on *untrusted* clients
  - All the ***trust*** still resides at the server
  - ***How can we partition trust more flexibly?***
- How to reconcile ***security*** and ***trust***?

Virtual machines and remote attestation

- **Goal**: attest *behavior*, not a particular executable image
- Two observations:
  - VMs that execute high-level, platform-independent code have a lot of meta-information about code – e.g. Class hierarchy
  - Code runs under complete control of a virtual machine

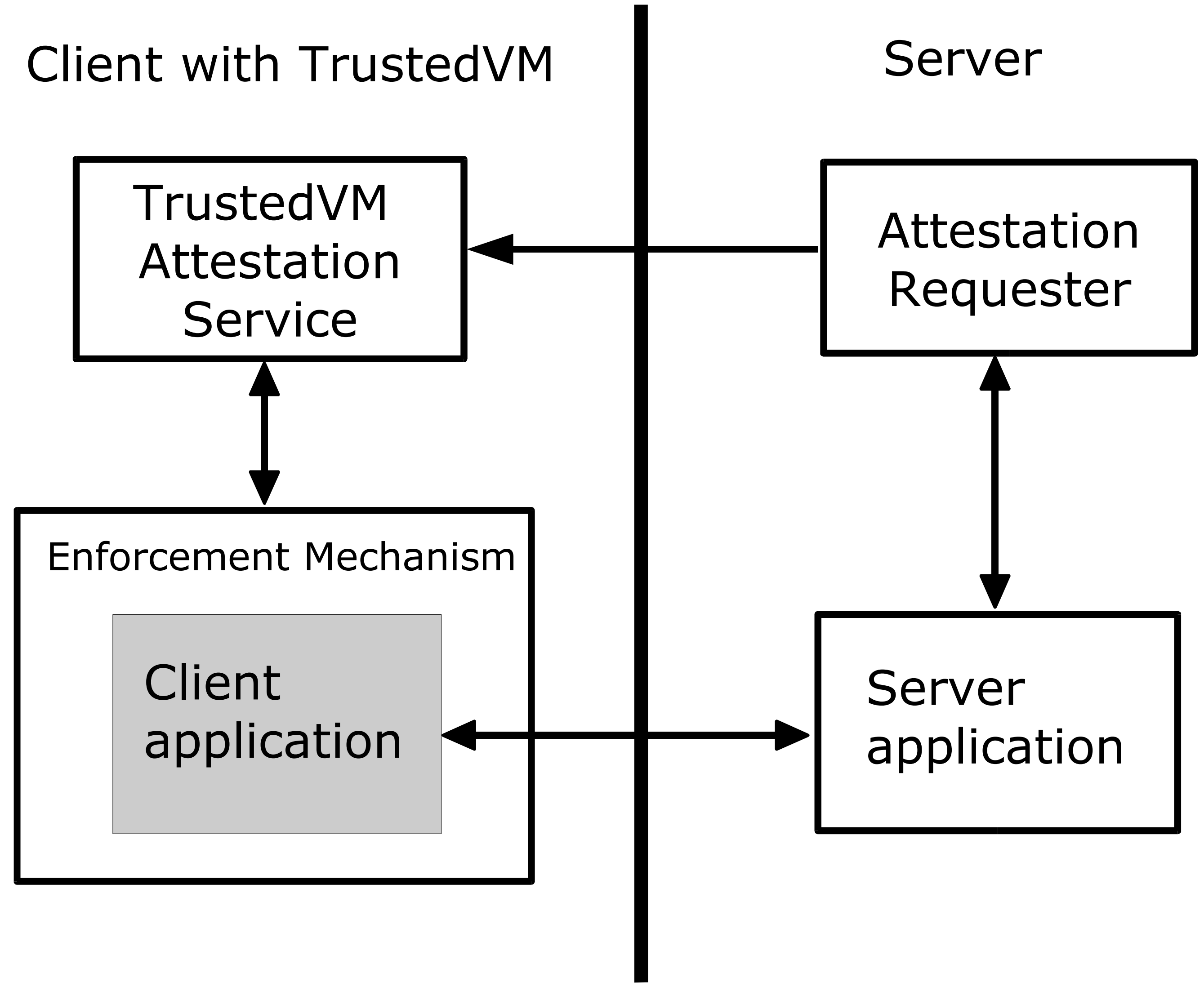
Semantic Remote Attestation

- Use a ***trusted virtual machine*** (TrustedVM) to attest properties of code running on it
- This is a *much more fine-grained and semantically richer operation than signing the hash of an executable* – ***semantic remote attestation***

What can a TrustedVM attest?

- ***Properties of classes*** – class hierarchies, restricted interfaces
- ***Dynamic properties*** – runtime state of program, information about input; install runtime monitors
- ***System properties*** – testing system abilities before running distributed computations
- In general, send code to TrustedVM to evaluate properties; test suites, static analyses

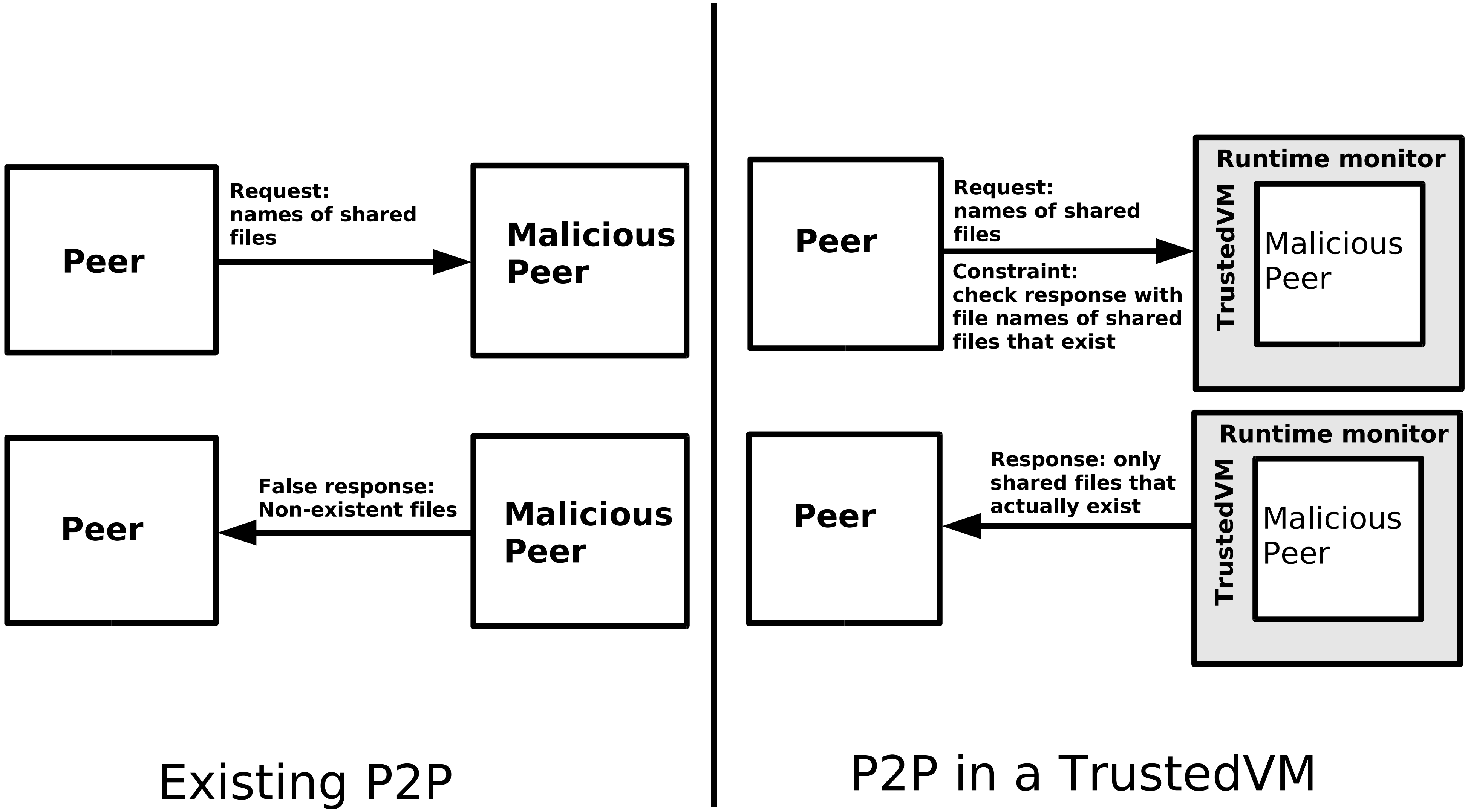
Framework for Semantic Remote Attestation



Semantic Remote Attestation: Examples

- Peer to peer networks
  - Depend on each client respecting the rules of the protocol
  - With semantic remote attestation, can ***explicitly check security requirements***
- Distributed computation – e.g. Mersenne Primes
  - Want to test a platform's capabilities
    - Before handing over a computation bundle
    - To evaluate “goodness” of results

Example: P2P in a TrustedVM


Peer
Request: names of shared files
Malicious Peer
Peer
False response: Non-existent files
Malicious Peer
Peer
Request: names of shared files
Constraint: check response with file names of shared files that exist
Runtime monitor
TrustedVM
Malicious Peer
Peer
Response: only shared files that actually exist
Runtime monitor
TrustedVM
Malicious Peer

Existing P2P

P2P in a TrustedVM

Advantages of semantic remote attestation

- It certifies ***program behavior*** – not a specific binary
- Allows ***various implementations***, as long as they satisfy required security criteria
- ***Dynamic*** – can attest runtime properties
- ***Flexible*** – can attest wide range of properties

Advantages of semantic remote attestation

- Trust relationships between nodes are made ***explicit***
- These are actually ***checked*** and ***enforced***
- ***Finer-grained trust:*** Degree of trustworthiness –
  - can know which properties were not satisfied – traditional attestation is all-or-nothing
  - Allows nodes to dynamically adjust trust relationships

In the works…

- Ways to attest ***information flow*** – both statically and dynamically
  - ***Statically*** – static analysis of Java bytecode – results can be easily checked
  - ***Dynamically*** – Mandatory Access Control (MAC) at the object-level inside the JVM
- Simulator for TCPA hardware module
  - Understand (debug?) the TCPA spec
  - Research vehicle for TC research

Conclusion

- Currently proposed mechanisms for Trusted Computing are severely limited
- Leveraging virtual machine technologies can make Trusted Computing more flexible and effective

Thank You

Interpreters, Virtual Machines and Emulators (IVME) 2004
(co-located with PLDI)
June 7, 2004, Washington, DC

http://www.ics.uci.edu/~franz/ivme