

SEMANTIC

REMOTE

ATTESTATION

VIVEK HALDAR

# Thesis

Remote attestation, one of the core mechanisms of Trusted Computing, can be performed in a way that:

- reasons expressively about program behavior and dynamic properties
- enables a flexible, graded notion of trust
- avoids intractable management problems at both the client and server end
- does not tie attestation to a specific executable binary

In short, remote attestation can attest program *properties*, rather than program *binaries*. I call this **semantic remote attestation**.

# Thesis

WHAT'S THAT?

Remote attestation, one of the core mechanisms of Trusted Computing, can be performed in a way that:

- reasons expressively about program behavior and dynamic properties
- enables a flexible, graded notion of trust
- avoids intractable management problems at both the client and server end
- does not tie attestation to a specific executable binary

In short, remote attestation can attest program *properties*, rather than program *binaries*. I call this **semantic remote attestation**.

## Thesis

WHY are these things PROBLEMS?

Remote attestation, one of the core mechanisms of Trusted Computing, can be performed in a way that:

- reasons expressively about program behavior and dynamic properties
- enables a flexible, graded notion of trust
- avoids intractable management problems at both the client and server end
- does not tie attestation to a specific executable binary

In short, remote attestation can attest program *properties*, rather than program *binaries*. I call this **semantic remote attestation**.

# Thesis

Remote attestation, one of the core mechanisms of Trusted Computing, can be performed in a way that:

- reasons expressively about program behavior and dynamic properties
- enables a flexible, graded notion of trust
- avoids intractable management problems at both the client and server end
- does not tie attestation to a specific executable binary

In short, remote attestation can attest program *properties*, rather than program *binaries*. I call this **semantic remote attestation**.

↳ HOW DO I DO THIS?

CAN YOU TRUST A COMPUTER?

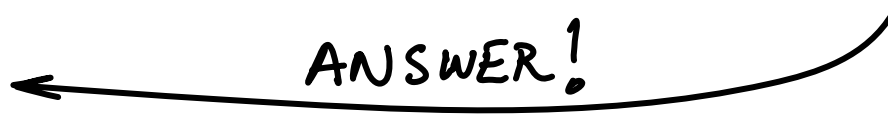
MACHINE A

MACHINE B

QUERY ?



ANSWER!



# CAN YOU TRUST A COMPUTER?

MACHINE A

MACHINE B

QUERY?



ANSWER!



WHAT COULD GO WRONG?

# CAN YOU TRUST A COMPUTER?

MACHINE A

MACHINE B

QUERY?



WHO IS THIS FROM?

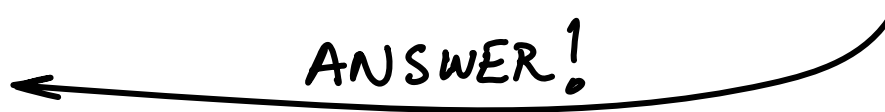
IS IT MALICIOUS?

HAS SOMEONE

TAMPHERED WITH

IT?

ANSWER!



WHAT COULD GO WRONG?

# CAN YOU TRUST A <sup>REMOTE</sup> COMPUTER?

MACHINE A

MACHINE B

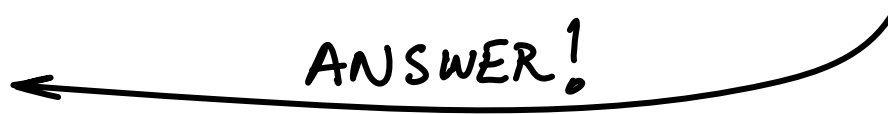
QUERY ?



WHO IS THIS FROM?  
IS IT MALICIOUS?  
HAS SOMEONE  
TAMPARED WITH  
IT?

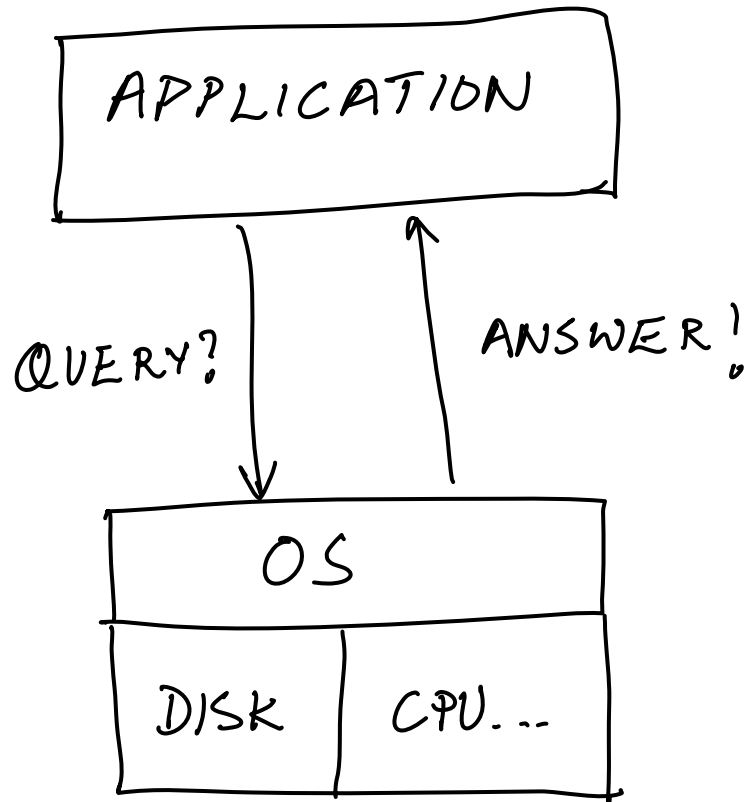
WHO IS THIS FROM?  
IS IT TRUE?  
MALICIOUS?  
TAMPARED?

ANSWER!

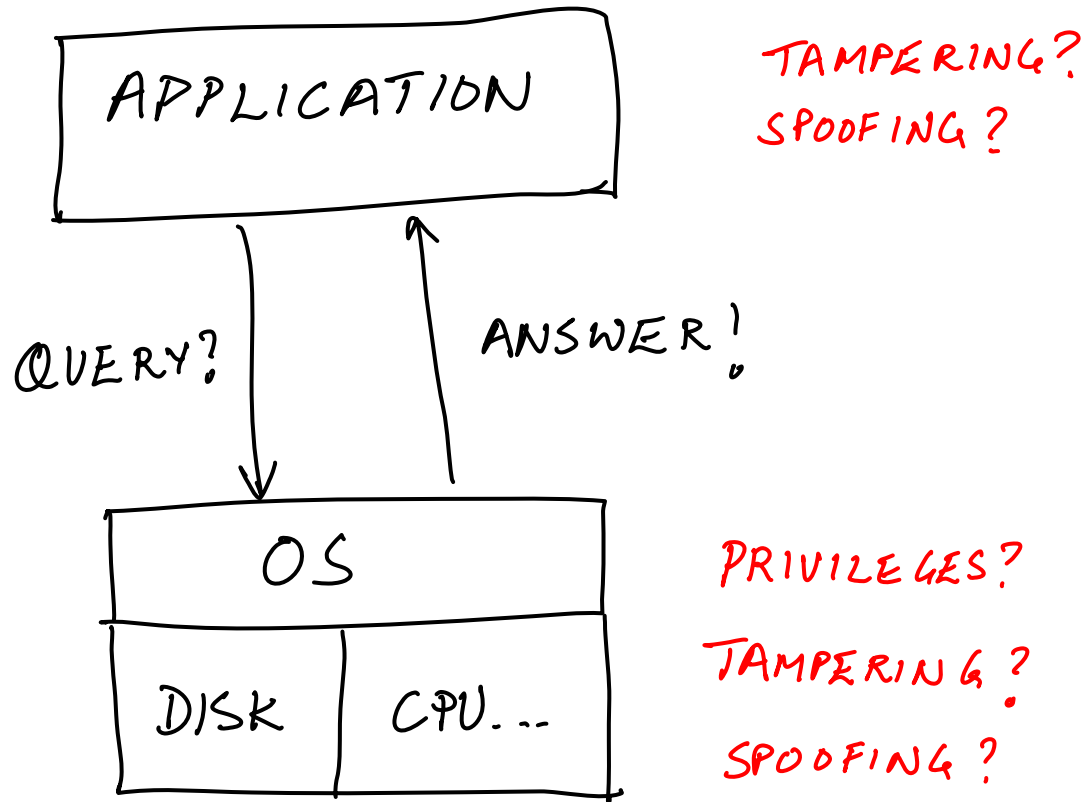


WHAT COULD GO WRONG?

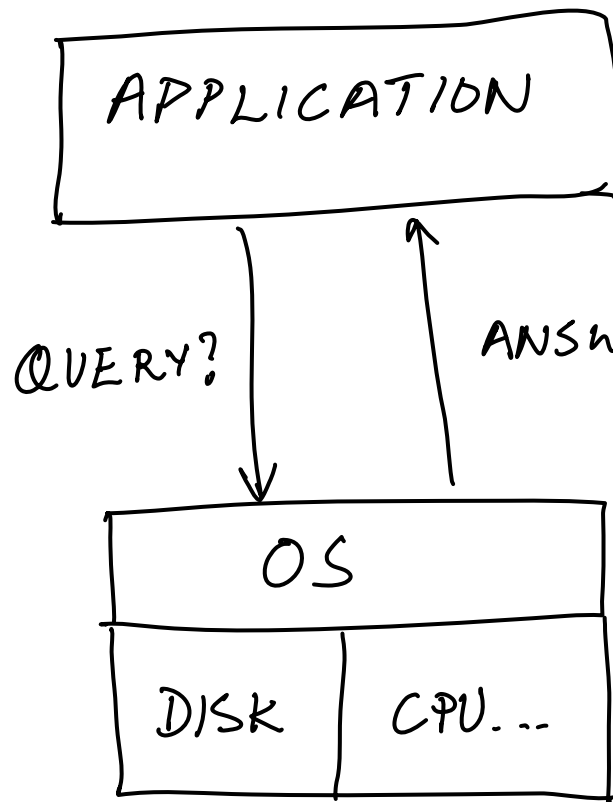
CAN YOU TRUST A <sup>LOCAL</sup> COMPUTER?



# CAN YOU TRUST A LOCAL COMPUTER?



# CAN YOU TRUST A LOCAL COMPUTER?



TAMPERING?  
SPOOFING?

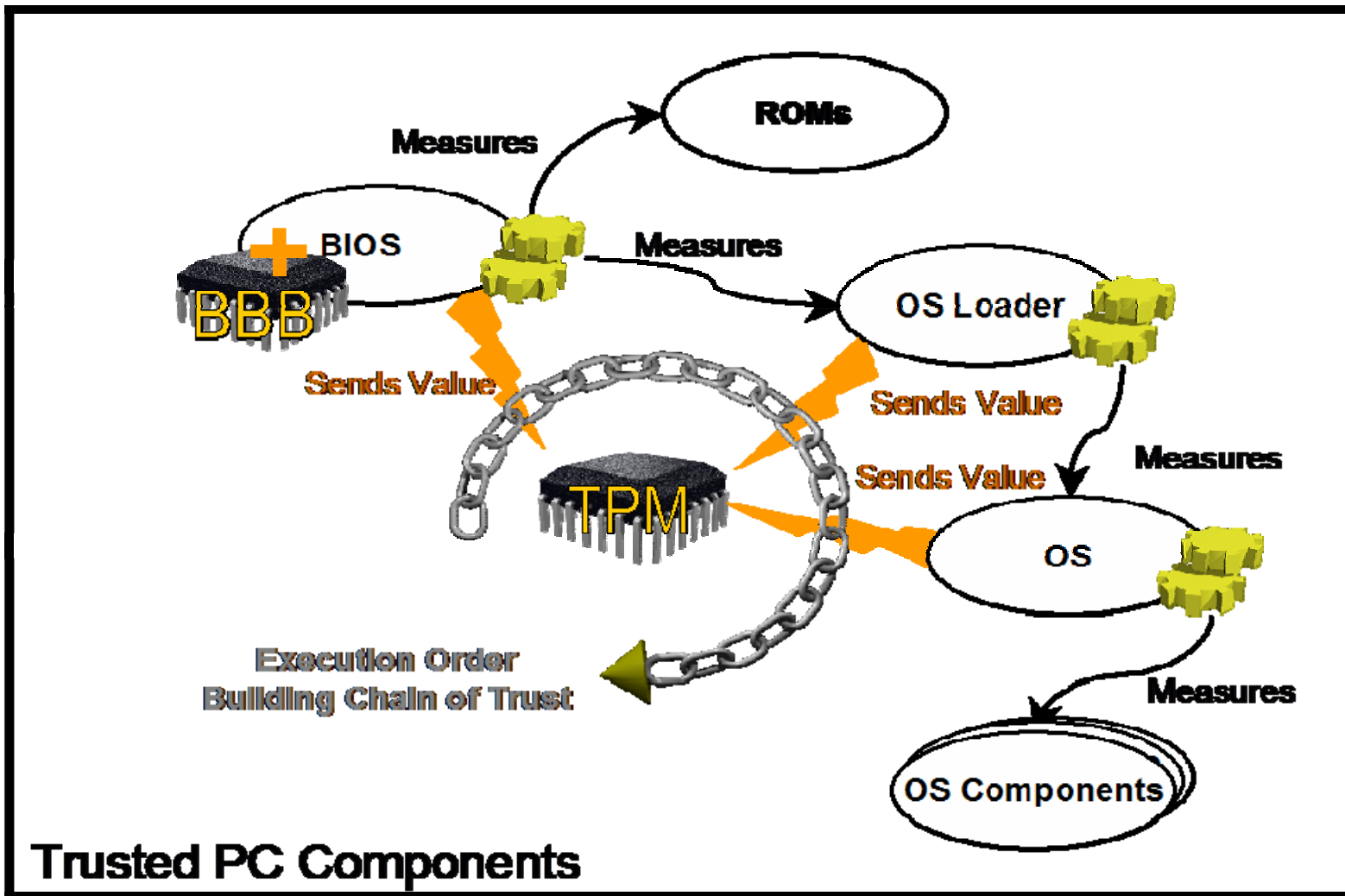
ROOTKITS WORK  
BY EXPLOITING  
WEAKNESSES AT  
THIS LEVEL

PRIVILEGES?  
TAMPERING?  
SPOOFING?

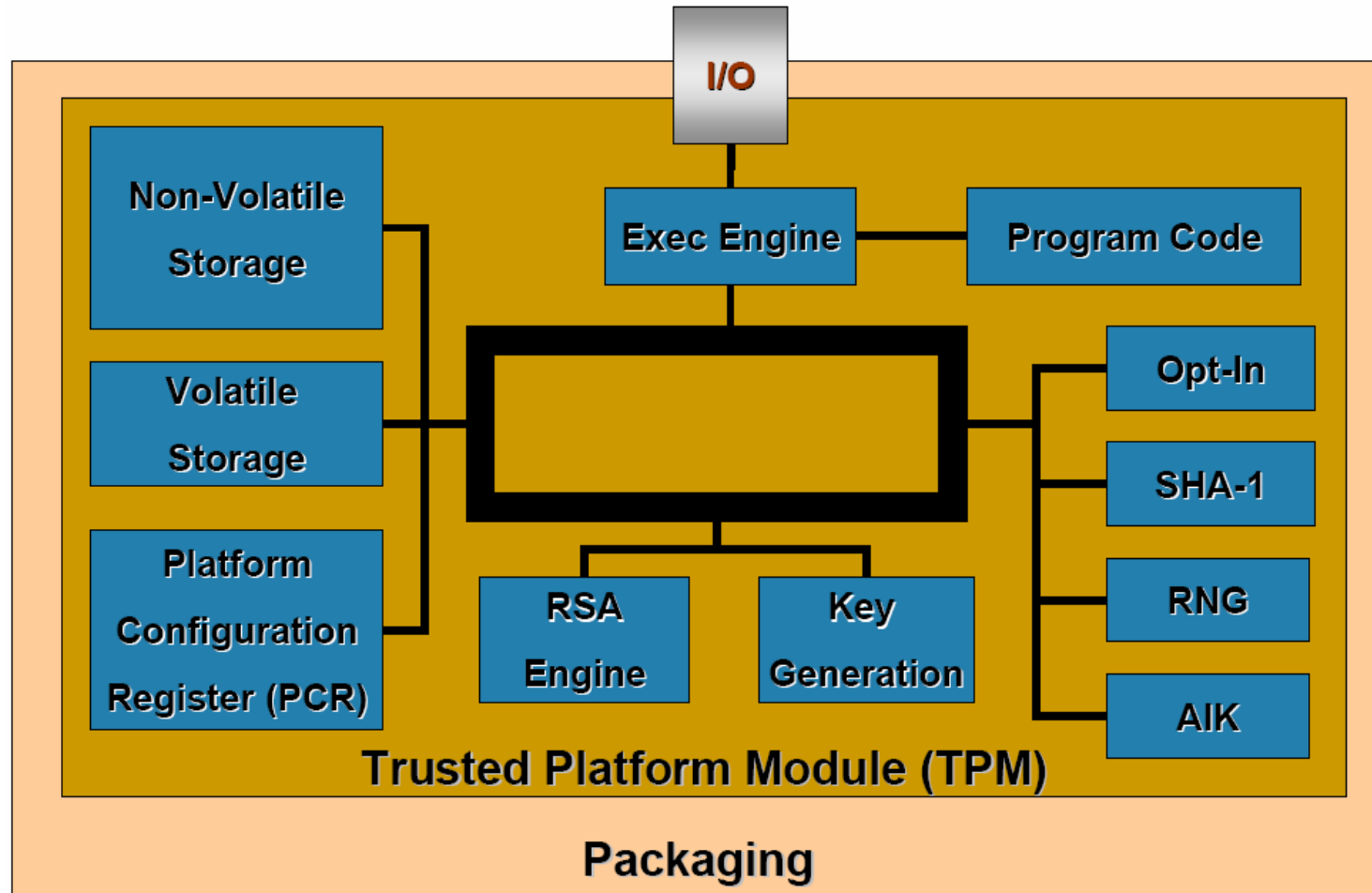
CANNOT USE  
SOFTWARE TO  
VOUCH FOR  
SOFTWARE !

NEED HARDWARE

# INDUCTIVE TRUST



# TRUSTED PLATFORM MODULE



# AUTHENTICATING SOFTWARE

Prover

Challenger

# AUTHENTICATING SOFTWARE

Prover

Show me you have  
← version x.y of FooBar

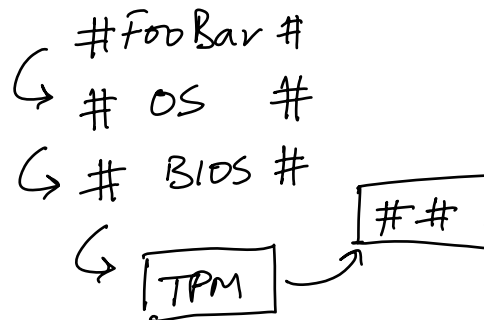
Challenger

# AUTHENTICATING SOFTWARE

Prover

Challenger

Show me you have  
← version x.y of FooBar

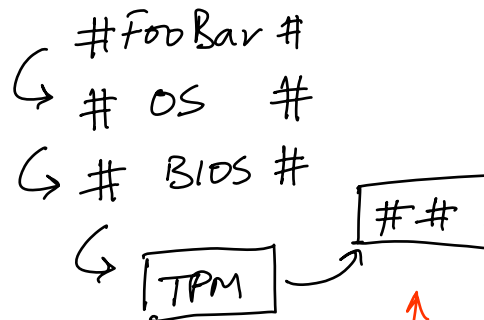


# AUTHENTICATING SOFTWARE

Prover

Challenger

Show me you have  
← version x.y of FooBar



Issued  
key by  
well-known  
CA

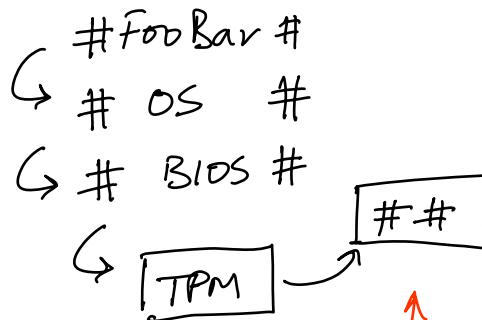
Signature  
chain

# AUTHENTICATING SOFTWARE

Prover

Challenger

Show me you have  
← version x.y of FooBar



Issued  
key by  
well-known  
CA

Signature  
chain

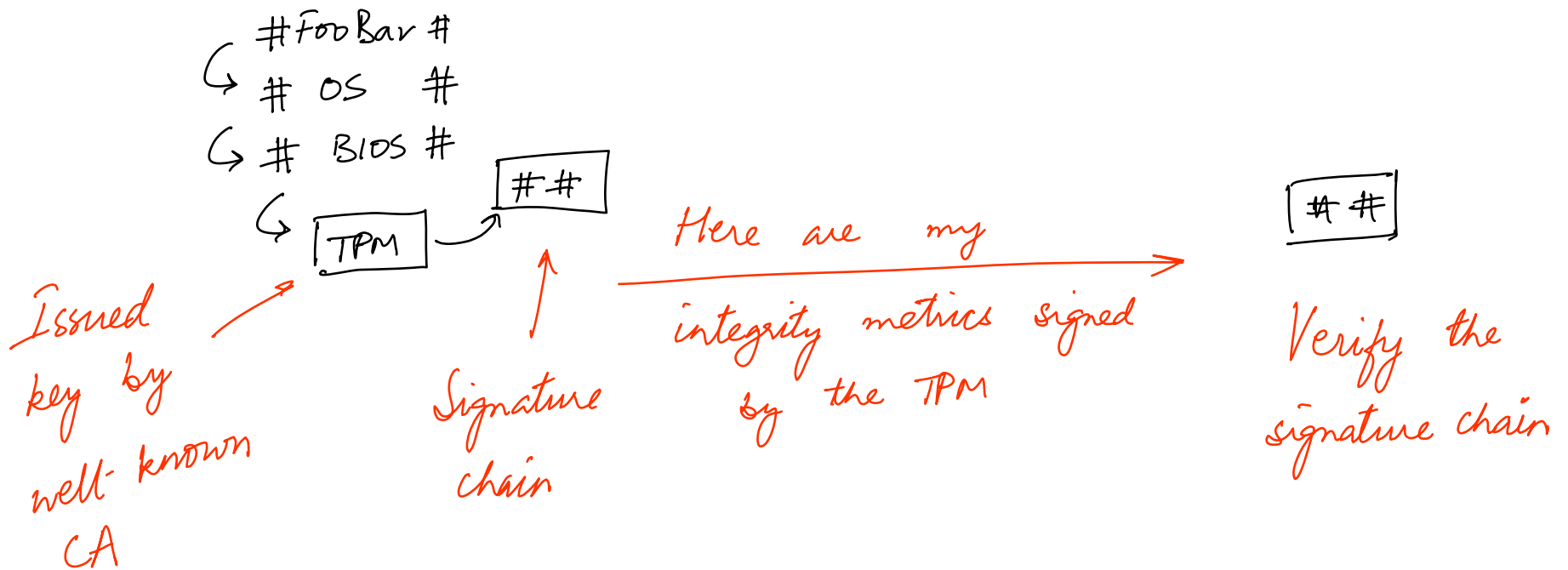
Here are my  
integrity metrics signed  
by the TPM

# AUTHENTICATING SOFTWARE

Prover

Challenger

Show me you have  
← version x.y of FooBar



# REMOTE ATTESTATION PROBLEMS

PROVER

CHALLENGER

Are you good?



Of course!

0x badd00de



# REMOTE ATTESTATION PROBLEMS

① DOES NOT

ATTEST OR

REASON ABOUT

PROGRAM

BEHAVIOR

# REMOTE ATTESTATION PROBLEMS

① DOES NOT

ATTEST OR

REASON ABOUT

PROGRAM

BEHAVIOR

• certifies a  
particular binary

↳ could have bugs!

TRUST



entities, crypto

vs SECURITY



behavior  
verification  
enforcement

# REMOTE ATTESTATION PROBLEMS

② STATIC

— CANNOT

CONVEY

DYNAMIC

INFORMATION

# REMOTE ATTESTATION PROBLEMS

② STATIC

— CANNOT

CONVEY

DYNAMIC

INFORMATION

- Runtime state of program

- Dynamic input

—————> Done ONE TIME

at the start!

# REMOTE ATTESTATION PROBLEMS

③ UPGRADES  
& PATCHES  
  
& VARIOUS  
PLATFORMS

# REMOTE ATTESTATION PROBLEMS

③ UPGRADES  
& PATCHES

Exponential  
space!

& VARIOUS

PLATFORMS

Binaries for various  
platforms?

Sources of

VARIABILITY in S/W

→ config files

→ shared libs

⋮

# REMOTE ATTESTATION PROBLEMS

PROVER

CHALLENGER

Are you good?



Of course!

0x badd00dc



How can we attest

relevant program behavior

while allowing a

range of implementations?

How can we partition

trust more flexibly,

away from the

trusted server / untrusted client

model?

# Virtual machines and remote attestation

- **Goal:** attest *behavior*, not a particular executable image
- Two observations:
  - VMs that execute high-level, platform-independent code have access to *meta-data about code* (e.g. Class hierarchy), and the *internals of program state*
  - Code runs under *complete control of a virtual machine*

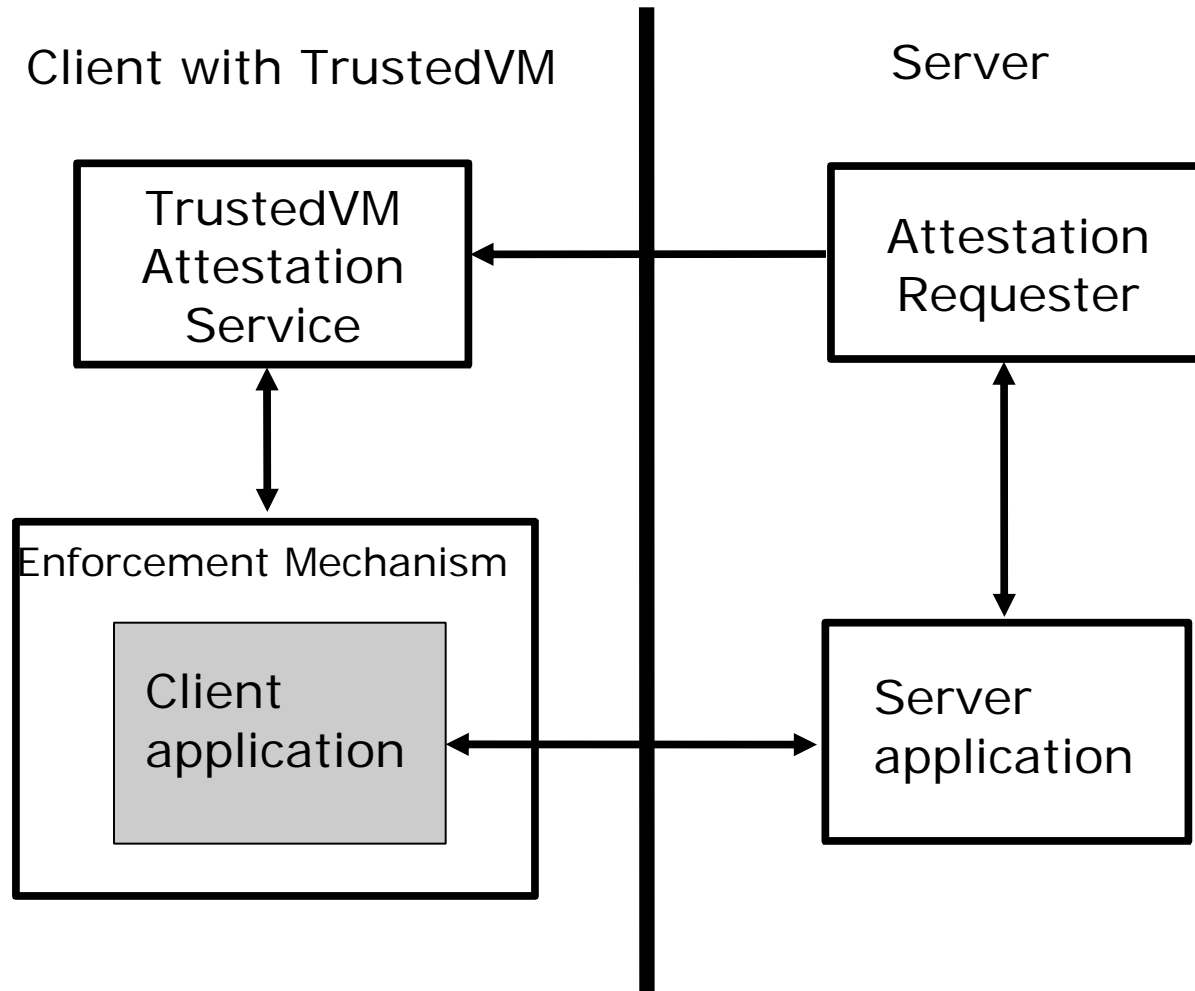
# Semantic Remote Attestation

- 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 properties 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

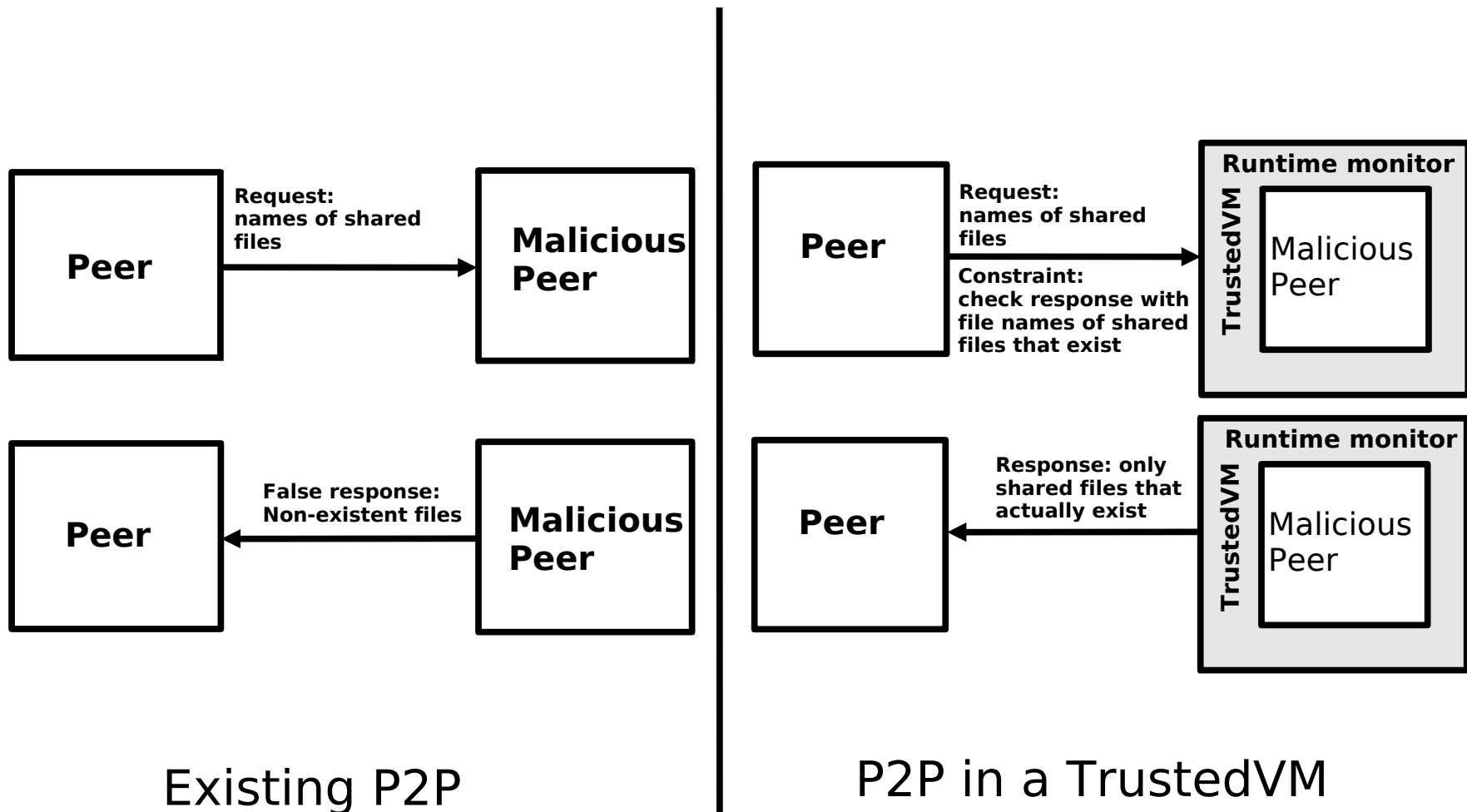
# Framework for Semantic Remote Attestation



# 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
  - Want to test a platform's capabilities before handing over a computation bundle
  - Evaluate quality-of-service metrics for application service providers – e.g. send “test suites”

# Example: 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

# Conclusion

- Currently proposed mechanisms for Trusted Computing are severely limited
  - Deploying TPMs is the right start...
  - ...but the currently proposed software infrastructure on top of that is too brittle and not scaleable
- Leveraging language-level virtual machine technologies can make Trusted Computing more flexible and effective

THANK YOU!

<http://vivekhaldar.com/sra>

<http://vivekhaldar.com/blog>