# Trusted Computing in OVAL

**Charles Schmidt**

**March 24, 2011**

HS SEDI | MITRE
Homeland Security Systems Engineering and Development Institute

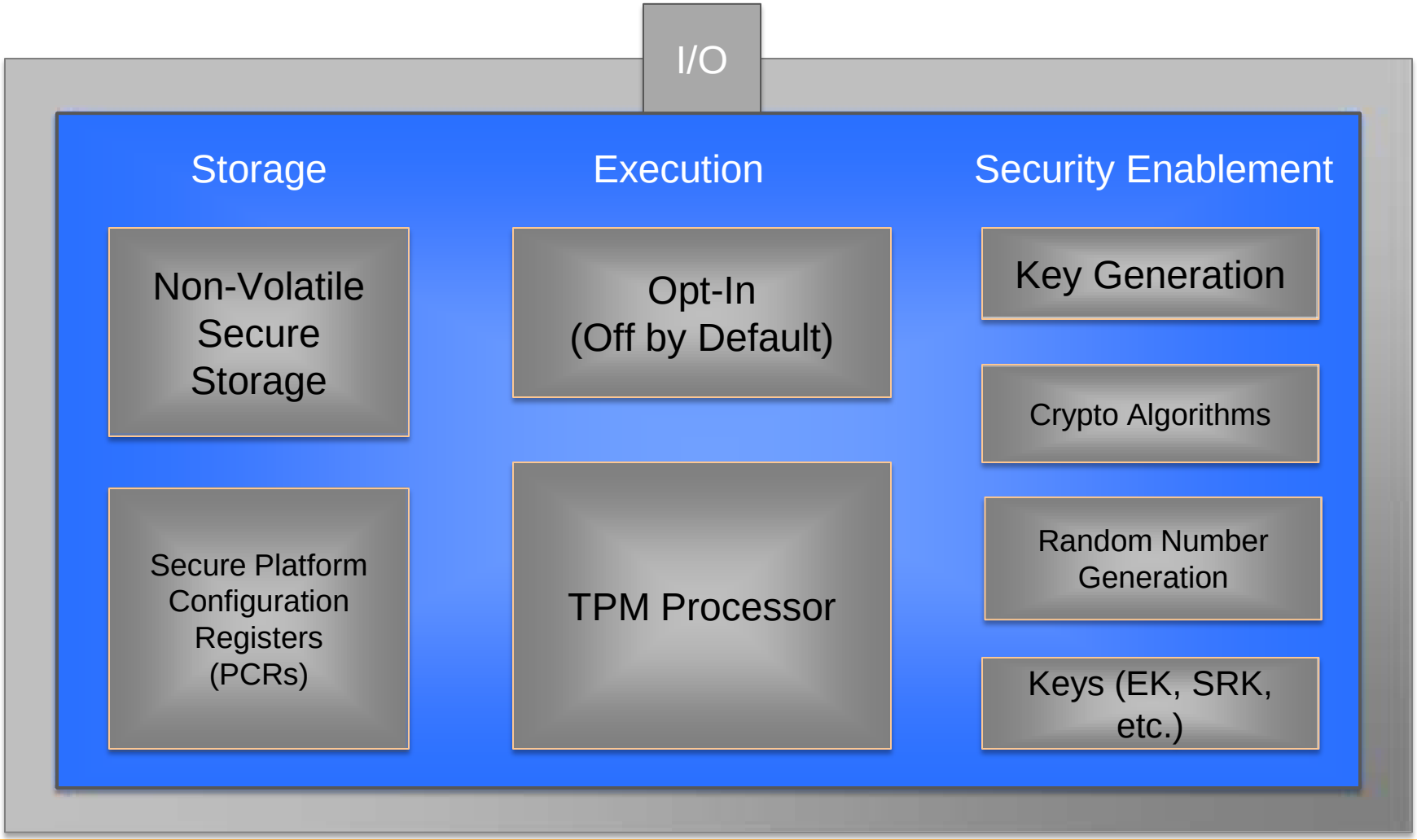
# What is Trusted Computing?

- A *trusted platform* contains hardware-rooted subsystem devoted to maintaining trust & security

- Three important roots
  - *Measurement*: Reliably gathering data
  - *Storage:* Securely store data (including TPM), data tampering detectable
  - *Reporting:* Reports data in a verifiable and trustworthy way

- New hardware:
  - The Trusted Platform Module (TPM)
    - Secure storage and reporting, dirt cheap
  - "Trusted hardware extensions" (TXT, SVM)
    - Flexible root of trust for measurement

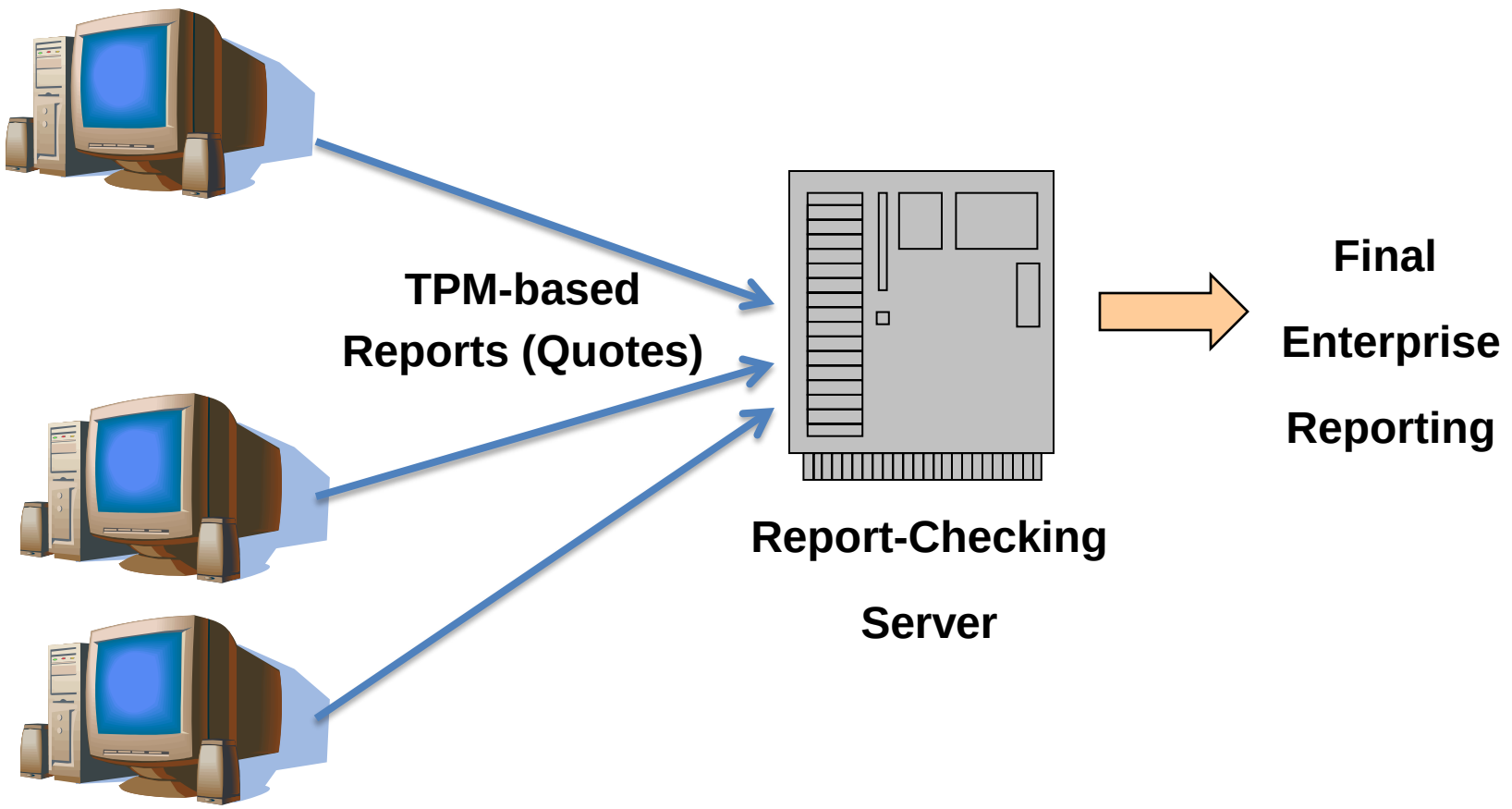
# The TPM Itself



# The TPM: What it Can Do

- **Secure Storage: Two kinds**
  - **Tiny amounts of measurement data in PCRs**
  - **Key material used to encrypt larger amounts of on-disk data**
  - ***Crucial capability: TPM residence of PCR data and storage root key***
- **Secure Reporting**
  - **Quote – PCR values signed by the TPM's core identity key**
  - **TPM's core identity key *never leaves the chip***
    - **Forms the root of a key hierarchy for reporting**
  - **PCR contents cannot be written arbitrarily**
    - **Final value a combination of multiple hashes from boot**
- **Limited cryptographic operations**

# Trusted Computing Paradigm



# Merge OVAL and TPM procedures

- **Compatible paradigms:**
  1. **Collect measurements from hosts**
  2. **Evaluate measurements against criteria to determine "compliance"**

- **Two benefits identified**
  - **use security automation standards for the collection and transport of attestation data**
  - **use attestation to provide a root of trust to existing uses of security automation standards**

# Using Standards to Improve TPM Paradigm

- Quote transport protocols are still under active development
  - Several proposals, but nothing that is universally accepted
  - Propose something we can use them effectively within our existing SCAP infrastructures
- TPM measurements in quotes lack granularity
  - PCRs are hashes of multiple measurements – virtually impossible to trace a bad hash to a single cause
  - OVAL measurements tell us what went wrong
- Goal of the new OVAL TPM Probe
  - Identify and transmit useful information about the TPM itself
  - Do this in conjunction with the regular, granular OVAL assessments

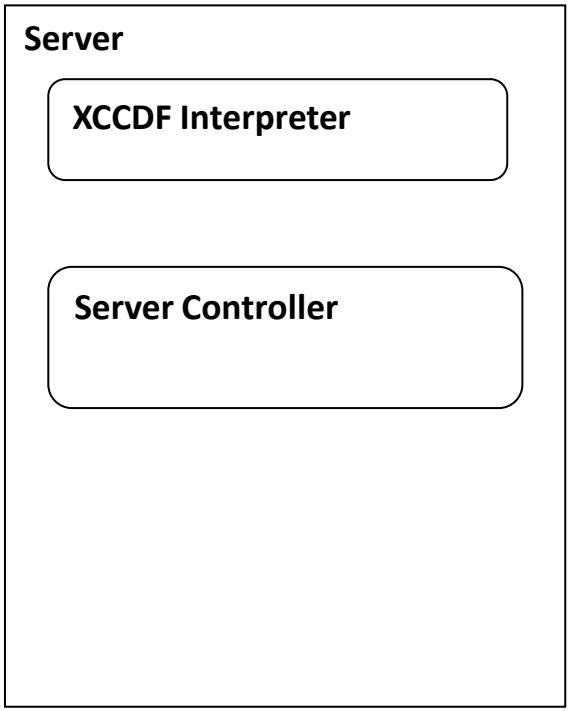
# Using TPMs to Improve OVAL Operation

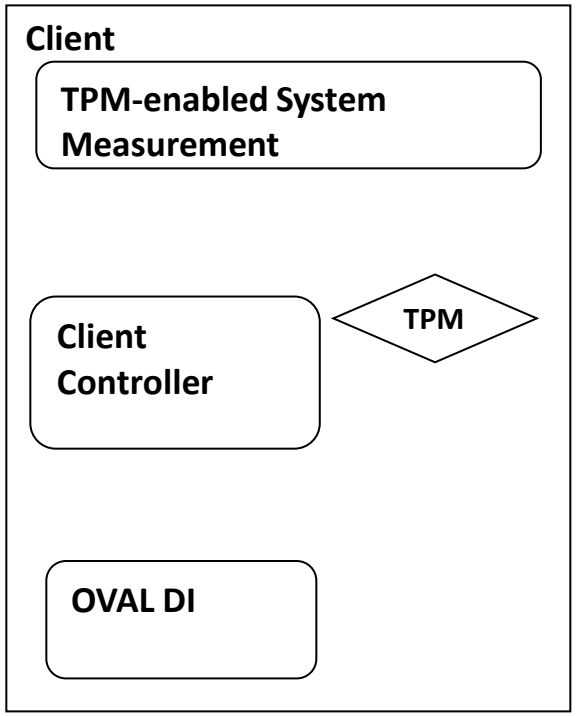
- Currently
  - OVAL can evaluate a vast array of settings on many platforms
  - However, OVAL evaluations have no trusted base
    - Only as trustworthy as the software OVAL interpreter + libraries
- Attesting to correctness of the OVAL interpreter allows us to trust OVAL measurements
  - The core of an OVAL interpreters is stateless (between evaluations) and can be quite small
- The TPM allows OVAL assessments themselves to be rooted in the hardware of the assessed system
- This is slightly different from the current OVAL paradigm
  - Quote information not evaluated by OVAL interpreter because it requires special operations (signature verification, etc.)
  - Instead, TPM quote's SC Item acts as a certification of the correct operation of the OVAL infrastructure

# The New TPM Component Schema

- quotereport_test
  - Purpose is to collect a TPM quote
  - Most fields cannot be usefully evaluated in OVAL; full evaluation of the quote results would need to take place externally using the information in the OVAL System Characteristics file
    - mask, aikblob, nonce, quotetype – required input to retrieve a quote
    - pcr, locality – measurements returned in a quote
    - signature, pcrcomposite – data structures provided in a quote for integrity
- tpminfo_test
  - Test inherent characteristics of a TPM chip
    - version, revision, errata_number, manufacturer, manufacturer_info, command_support_level, pcr_count, pcr_attributes, buffer_size

# Demonstration Architecture




**XCCDF Interp – Processes policies**

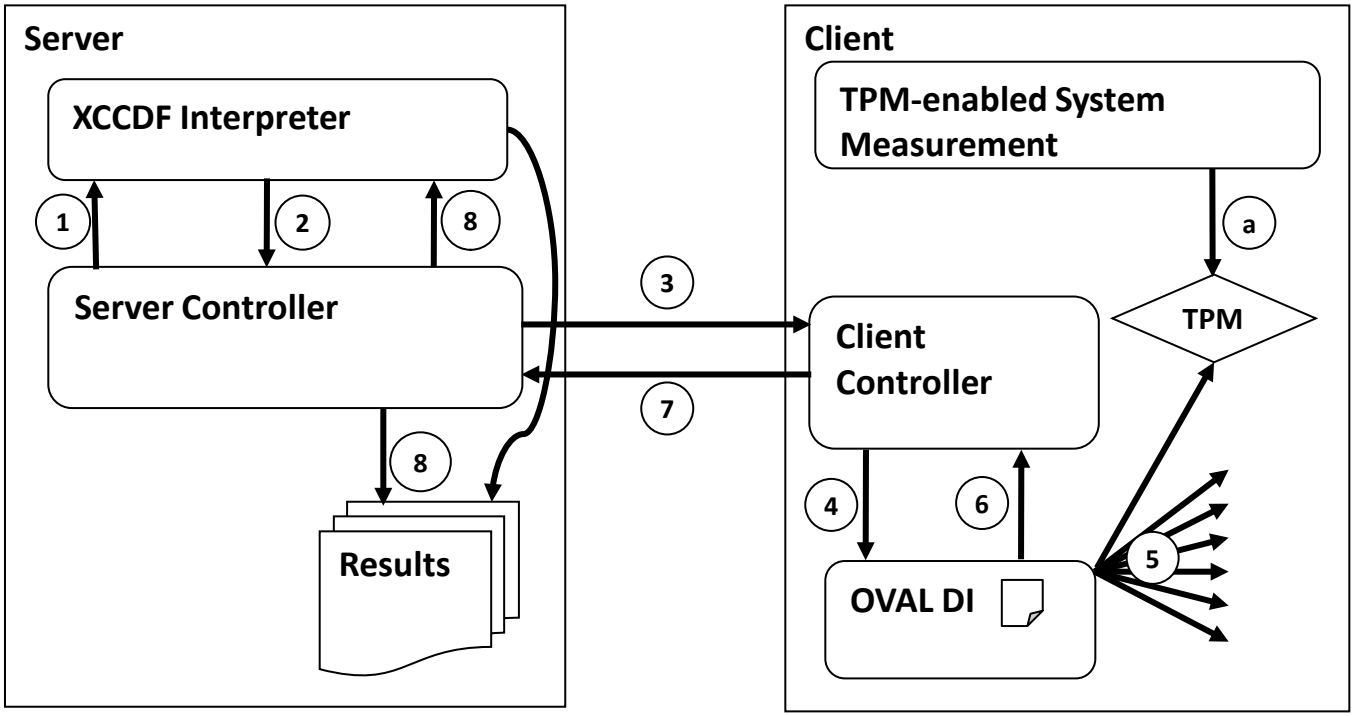
**Server Controller – Orchestrates interactions between OVAL DI, XCCDF Interp., and client**

**Measurement – Measures system, including OVAL DI**

**OVAL DI – Collects/evaluates findings**

**Client Controller – Orchestrate between server and local OVAL DI**

# Demonstration Architecture




1. The user initiates an assessment
2. XCCDF Interp. processes the guidance, which calls for OVAL definitions
3. The Server Controller passes them to the Client Controller…
4. … which passes them to the Client OVAL DI
5. OVAL DI collects system findings, including a TPM quote, and evaluates to produce results, as normal
6. The results are sent to the Client Controller
7. The Server Controller passes results to the XCCDF Interp. to get the configuration results and also stores raw results
8. The Server Controller sends the assessment results to the Server Controller

# Demo status

- Current status
  - Dynamically insert TPM query definition into OVAL files
    - Can be done starting with XCCDF or raw OVAL
  - Collect TPM data and store in OVAL results
    - Specifically, in the system-characteristics part of the results
  - Use collected data to verify integrity of the data
    - Ensure quote itself has not been tampered with
- Next step
  - Insert measurements of OVAL Interpreter into TPM
  - Verify those measurements on the server

# Questions for the Community

- Is this new capability useful?
  - Would vendors be interested in having TPM measurements of their assessment engines?
  - Is OVAL the correct way to retrieve this information?
    - We are using OVAL to task and transport quotes, but processing is still external to OVAL
- What other TPM data to collect?
  - Quote collects measurements
  - Other probes could collect configuration and capabilities of a TPM
    - TPM enabled or disabled
    - TPM provisioned or in factory default
    - Manufacturer, build set, manufacturer custom strings
    - Characteristics of the chip (IO speed, number of registers, etc.)
    - TPM device driver version and support level

# Questions?

# quotereport_test

## Object

```
<xsd:extension base="oval-def:ObjectType">
  <xsd:sequence>
    <xsd:element name="mask" type="oval-def:EntityObjectIntType" minOccurs="1" maxOccurs="unbounded"/>
    <xsd:element name="quotetype" type="tpm-def:EntityObjectQuotetypeType" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="aikblob" type="oval-def:EntityObjectBinaryType" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="nonce" type="oval-def:EntityObjectBinaryType" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:extension>
```

## State

```
<xsd:extension base="oval-def:StateType">
 <xsd:sequence>
  <xsd:element name="mask" type="oval-def:EntityStateIntType" minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="aikblob" type="oval-def:EntityStateBinaryType" minOccurs="0" maxOccurs="1"/>
  <xsd:element name="nonce" type="oval-def:EntityStateBinaryType" minOccurs="0" maxOccurs="1"/>
  <xsd:element name="quotetype" type="tpm-def:EntityStateQuotetypeType" minOccurs="0" maxOccurs="1"/>
  <xsd:element name="signature" type="oval-def:EntityStateBinaryType" minOccurs="0" maxOccurs="1"/>
  <xsd:element name="pcr" type="oval-def:EntityStateRecordType" minOccurs="0" maxOccurs="1"/>
  <xsd:element name="pcrcomposite" type="oval-def:EntityStateBinaryType" minOccurs="0" maxOccurs="1"/>
  <xsd:element name="locality" type="oval-def:EntityStateIntType" minOccurs="0" maxOccurs="1"/>
 </xsd:sequence>
</xsd:extension>
```

# tpminfo_test

## State

```
<xsd:extension base="oval-def:StateType">
  <xsd:sequence>
    <xsd:element name="version" type="oval-def:EntityStateVersionType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="revision" type="oval-def:EntityStateVersionType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="errata_number" type="oval-def:EntityStateIntType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="manufacturer" type="oval-def:EntityStateBinaryType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="manufacturer_info" type="oval-def:EntityStateBinaryType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="command_support_level" type="oval-def:EntityStateBinaryType" minOccurs="0"
                 maxOccurs="1"/>
    <xsd:element name="pcr_count" type="oval-def:EntityStateIntType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="pcr_attributes" type="oval-def:EntityStateBinaryType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="buffer_size" type="oval-def:EntityStateIntType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:extension>
```