Trust

What it is and how to get it

Dr. Perry Alexander

Information and Telecommunication Technology Center
Electrical Engineering and Computer Science
The University of Kansas
palexand@ku.edu

Formatted with the Beamer Class for $\LaTeX\,2_\varepsilon$

Defining Trust

Trust

"An entity can be trusted if it always behaves in the exptected manner for the intended purpose [**?**]"

Defining Trust

Properties

- Unambiguous identification
- Unimpeded operation
- First-hand observation of good behavior *or* indirect experience of good behavior by a trusted third party

Necessary Capabilities for Trust

- *Strong Identification* — An unambiguous, immutable identifier associated with the platform. The identifier is a protected encryption key in the TXT implementation.
- *Reporting Configuration* — An unambiguous identification mechanism for software and hardware running on the platform. The mechanism is hashing in the TXT implementation

Trust is a Preorder

$T^{\mathrm{x}}[\mathrm{y}]$ is an homogeneous relation over actors that is true when *x trusts y*. $T^{\mathrm{x}}[\mathrm{y}]$ is a preorer:

- Reflexive - $\forall x \cdot T^{\mathrm{x}}[\mathrm{x}]$
- Transitive - $\forall x, y, z \cdot T^{\mathrm{y}}[\mathrm{x}] \wedge T^{\mathrm{z}}[\mathrm{y}] \Rightarrow T^{\mathrm{z}}[\mathrm{x}]$

The transitive property defines *chains of trust*.

Trusted Platform Module

The *Trusted Platform Module (TPM)* is a cryptographic coprocessor for trust.

- Endorsement Key (EK) — factory generated asymmetric key that uniquely identifies the TPM
- Attestation Instance Key (AIK) — `TPM_CreateIdentity` generated asymmetric key alias for the EK
- Storage Root Key (SRK) — `TPM_TakeOwnership` generated asymmetric key that encrypts data associated with the TPM
- Platform Configuration Registers (PCRs) — protected registers for storing and extending hashes
- NVRAM — Non-volatile storage associated with the TPM

Endorsement Key

- Asymmetric key generated at TPM fabrication
- EK^{-1} is protected by the TPM
- *EK* by convention is managed by a Certificate Authority
  - Binds *EK* with a platform
  - Classic trusted third party
- Only used for encryption
- Attestation Instance Keys (AIK) are aliases for the EK
  - Used for signing
  - Authorized by the EK

Storage Root Key

- Asymmetric key generated by `TPM_TakeOwnership`
- SRK^{-1} is protected by the TPM
- SRK is available for encryption
- Used as the root for chaining keys by *wrapping*
  - A wrapped key is an asymmetric key pair with it's private key sealed
  - Safe to share the entire key
  - Only usable in the presence of the wrapping key with expected PCRs

Platform Configuration Registers

- Operations on PCRs
  - Extension — Hash a new value juxtaposed with the existing PCR value
  - Reset — Set to 0
  - Set — Set to a known value
- Operations using PCRs
  - Sealing data — PCR state dependent encryption
  - Wrapping keys — PCR state dependent encryption of a private key
  - Quote — Reporting PCR values to a third party
- Properties
  - Locality — Access control
  - Resettable — Can a PCR be reset
  - Many others that we don't need yet

Roots of Trust

A *root of trust* provides a basis for transitively building trust. Roots of trust are trusted implicitly.

There are three important Roots of Trust:

- Root of Trust for Measurement (RTM)
- Root of Trust for Reporting (RTR)
- Root of Trust for Storage (RTS)

Root of Trust for Measurement

A *Root of Trust for Measurement* is trusted to take the base system measurement.

- A hash function called on an initial code base from a protected execution environment
- Starts the measurement process during boot
- In the Intel TXT process the RTM is `SENTER` implemented on the processor

Root of Trust for Reporting

A *Root of Trust for Reporting* is trusted to guarantee the integrity of the base system report or quote

- A protected key used for authenticating reports
- In the Intel TXT processes this is the TPM's Endorsement Key (EK)
- Created and bound to its platform by the TPM foundry
- EK^{-1} is stored in the TPM and cannot be accessed by any entity other than the TPM
- EK is available for encrypting data for the TPM
- EK^{-1} is used for decrypting data inside the TPM
- Linking EK to its platform is done by a trusted Certificate Authority (CA)

Root of Trust for Storage

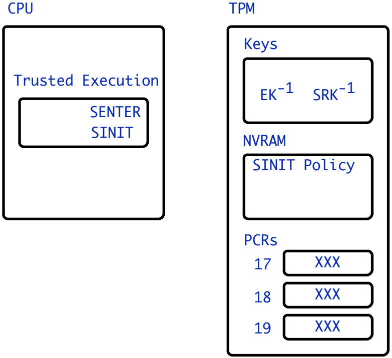
A *Root of Trust for Storage* is trusted to protect stored data

- A key stored in a protected location
- In the Intel TXT boot process this is the TPM's Storage Root Key (SRK)
- Created by `TPM_TakeOwnership`
- SRK^{-1} is stored in the TPM and cannot be accessed by any entity other than the TPM
- SRK is available for encrypting data for the TPM
- SRK is used for protecting other keys

One Step from Roots of Trust

Roots of trust are used to build a trusted system from boot.

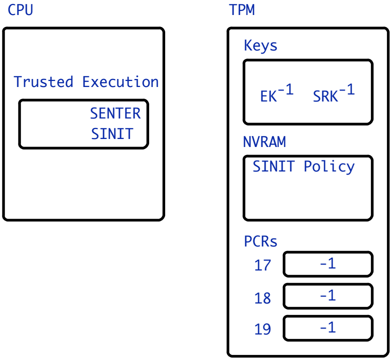
- Power-on reset, resettable PCRs set to -1
- `SENTER` called
- `SENTER` resets resettable PCRs to 0
- `SENTER` measures `SINIT` policy into PCR 18



One Step from Roots of Trust

Roots of trust are used to build a trusted system from boot.

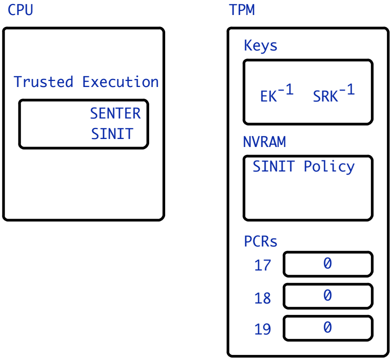
- Power-on reset, resettable PCRs set to -1
- `SENTER` called
- `SENTER` resets resettable PCRs to 0
- `SENTER` measures `SINIT` policy into PCR 18



One Step from Roots of Trust

Roots of trust are used to build a trusted system from boot.

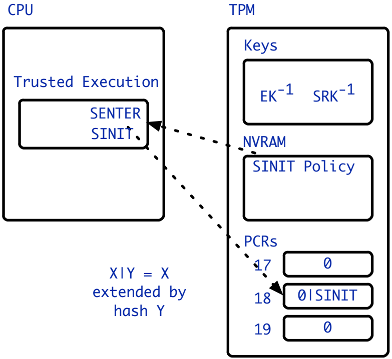
- Power-on reset, resettable PCRs set to -1
- `SENTER` called
- `SENTER` resets resettable PCRs to 0
- `SENTER` measures `SINIT` policy into PCR 18



One Step from Roots of Trust

Roots of trust are used to build a trusted system from boot.

- Power-on reset, resettable PCRs set to -1
- `SENTER` called
- `SENTER` resets resettable PCRs to 0
- `SENTER` measures `SINIT` policy into PCR 18



What We Know From Good PCR 18

A good value in PCR 18 tells us:

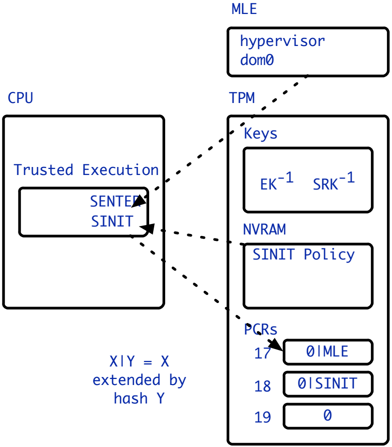
- `SENTER` was called — Resetting PCR 18 starts measurements at 0 rather than -1
- `SINIT` was measured by `SENTER` — Only `SENTER` can extend PCR 18
- `SINIT` uses the correct policy — PCR 18 is extended with `SINIT` measurement policy
- `SENTER` ran before `SINIT` was measured — $A \mid B \neq B \mid A$

Measurement $\neq$ Trust

Measurements must be appraised to determine trust.

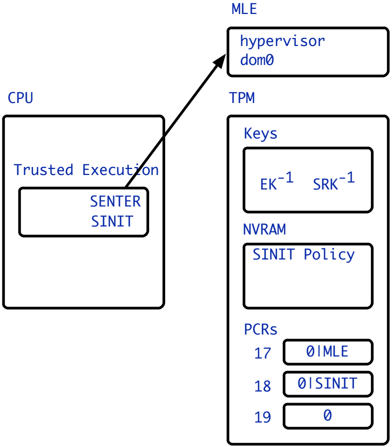
Two Steps from Roots of Trust

- `SINIT` measures the Measured Launch Environment (MLE) using measured policy
- `SINIT` returns control to `SENTER`
- `SENTER` invokes the MLE



Two Steps from Roots of Trust

- SINIT measures the Measured Launch Environment (MLE) using measured policy
- SINIT returns control to SENTER
- SENTER invokes the MLE

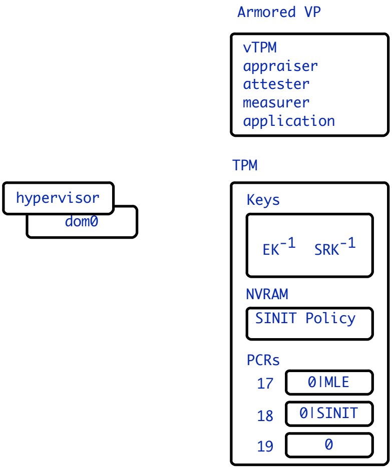


What We Know From Good PCRs

- SENTER was called — Resetting PCR 18 starts measurement sequence at 0 rather than -1
- SINIT policy was measured by SENTER — Only SENTER can extend PCR 18
- SINIT uses the correct policy — PCR 18 is extended with SINIT measurement policy
- SENTER ran before SINIT — $0 \mid \mathit{SINIT} \neq -1 \mid \mathit{SINIT}$
- SLE is good — Measured by good SINIT into PCR
- Initial OS is good — Measured by good SLE into PCR

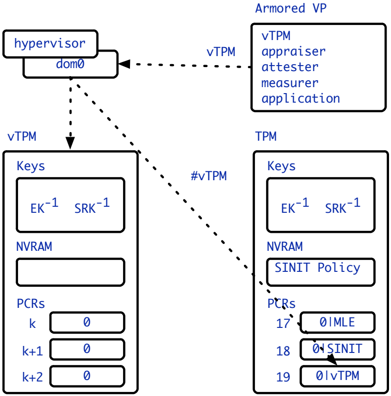
Boot the MLE

- `SENTER` starts the MLE
  - `SENTER` starts the hypervisor
  - `SENTER` passes dom0 to hypervisor
  - hypervisor starts dom0
- dom0 constructs the Armored VP
  - Measures the vTPM into the TPM
  - Starts the vTPM
  - Measures remaining Armored VMs into the vTPM
  - Starts remaining Armored VMs
  - Measures Armored application into the vTPM
  - Starts the Armored application



Boot the vTPM

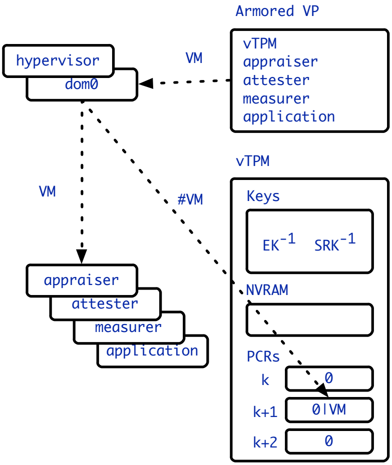
- SENTER starts the MLE
  - SENTER starts the hypervisor
  - SENTER passes dom0 to hypervisor
  - hypervisor starts dom0
- dom0 constructs the Armored VP
  - Measures the vTPM into the TPM
  - Starts the vTPM
  - Measures remaining Armored VMs into the vTPM
  - Starts remaining Armored VMs
  - Measures Armored application into the vTPM
  - Starts the Armored application



INFORMATION & TELECOMMUNICATION TECHNOLOGY CENTER
The University of Kansas

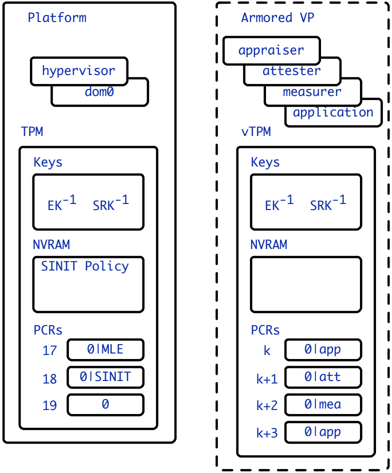
Boot the Armor Virtual Platform

- SENTER starts the MLE
    - SENTER starts the hypervisor
    - SENTER passes dom0 to hypervisor
    - hypervisor starts dom0
- dom0 constructs the Armored VP
    - Measures the vTPM into the TPM
    - Starts the vTPM
    - Measures remaining Armored VMs into the vTPM
    - Starts remaining Armored VMs
    - Measures Armored application into the vTPM
    - Starts the Armored application



Boot Result

- SENTER starts the MLE
  - SENTER starts the hypervisor
  - SENTER passes dom0 to hypervisor
  - hypervisor starts dom0
- dom0 constructs the Armored VP
  - Measures the vTPM into the TPM
  - Starts the vTPM
  - Measures remaining Armored VMs into the vTPM
  - Starts remaining Armored VMs
  - Measures Armored application into the vTPM
  - Starts the Armored application



Chaining Trust

- Trust is transitive
  - $T^x[y] \wedge T^y[z] \Rightarrow T^x[z]$
  - Construct chains of trust
  - Remember "directly observed or indirectly observed by a trusted third party"
- Roots of Trust define the "root" for trust
  - Use Roots of Trust to establish base for chain
  - RTM generates a trusted first measurement
  - RTS protects first measurement
  - RTR signs base quote for appraiser (eventually)
- Extend chains of trust by measuring before executing

Presentation Outline

- Review access control modeling objectives
  - modeling platform MAC
  - modeling local access control
- Overview access control policy definition
  - design and modeling assumptions
  - platform boot policy definition
  - local policy definitions
- Overview models
  - domain and system models
  - communication model
  - theorems and status
- Identify next steps
  - runtime and moving beyond the SVP line
  - adding M&A detail

Access Control Modeling Objectives

Reporting joint work with Geoffrey Brown, Indiana University (submitted) in which we verify two physical layer protocols.

- Biphase Mark Protocol (BMP)
- 8N1 Protocol

These protocols are used in data transmission for CDs, Ethernet, and Tokenring, etc. as well as UARTs.

- Correctness is reasonably difficult to prove due to many real-time constraints.
- Many previous formal modeling/verification efforts for these protocols.

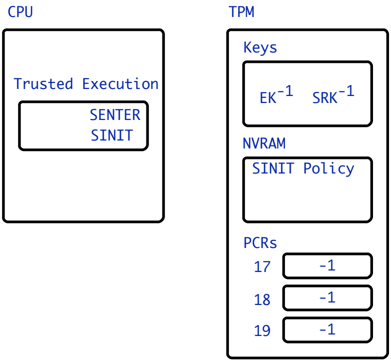
Columns and Blocks

Trying figures next to lists

Some normal text goes here just for introduction

- Appraisal
- Measurement
- Attestation
- vTPM

Why is this column getting higher?
Maybe it's not
Center alignment seems best.
I like this for two column test and graphics
Getting higher???



Big Picture

Armor Architecture

Simple Block

Introduction to $\LaTeX$

Beamer is a $\LaTeX$class for creating presentations that are held using a projector..."

This is a definition

Not really a proof.

1. This is a step

$\square$

Proofs

Not really a proof.

1. This is a step
2. This is another step

$\square$

Not really a proof.

1. This is a step
2. This is another step
3. This is a third step
4. This is a third step
5. This is a third step
6. This is a third step

$\square$

List with Overlays

- Item 1 followed by a pause

List with Overlays

- Item 1 followed by a pause
- Item 3 followed by a pause

List with Overlays

- Item 1 followed by a pause
- Item 2 followed by a pause
- Item 3 followed by a pause

Previous Efforts

- BMP has been verified in PVS twice and required
  - 37 invariants and 4000 individual proof directives (initially) in the one effort
  - 5 hours just to *check* the proofs in the other effort
  - A formal specification and verification of an independent real-time model in both efforts
- BMP has been verified in (the precursor to) ACL2 by J. Moore and required
  - A significant conceptual effort to fit the problem in the logic, arguably omitting some salient features of the model
  - The statement and proof of many antecedent results
  - J. Moore reports this as one of his “best ideas” in his career

Not Your Father's Theorem-Prover

The verifications are carried out in the SAL infinite-state bounded model-checker that combines SAT-solving and SMT decision procedures to *prove* safety properties about infinite-state models.

- Theorem-proving efforts took multiple engineer-months if not years to complete.
- Our initial effort in SAL consumed about *two engineer-days*. ...and we found a significant bug in a UART application note.

Parameterized Timing Constraints

SMT allows for *parameterized* proofs of correctness. The following are example constaints from the BMP verification:

```
TIME: TYPE = REAL;

TPERIOD: TIME = 16;
TSAMPLE: INTEGER = 23;

TSETTLE: {x: TIME |     0 <= x
                    AND (x + TPERIOD < TSAMPLE)
                    AND (x + TSAMPLE + 1 < 2 * TPERIOD)};

TSTABLE: TIME = TPERIOD - TSETTLE;

ERROR: {x: TIME |     (0 <= x)
                  AND (TPERIOD + TSETTLE < TSAMPLE*(1-x))
                  AND (TSAMPLE*(1+x) + (1+x) + TSETTLE < 2 * TPERIOD)};

RSAMPMAX: TIME = TSAMPLE * (1 + ERROR);
RSAMPMIN: TIME = TSAMPLE * (1 - ERROR);
RSCANMAX: TIME = 1 + ERROR;
RSCANMIN: TIME = 1 - ERROR;
```

KU INFORMATION & TELECOMMUNICATION TECHNOLOGY CENTER
The University of Kansas

SRI's SAL Toolset

- Parser
- Simulator
- Symbolic model-checker (BDDs)
- Witness symbolic model-checker
- Bounded model-checker
- Infinite-state bounded model-checker
- Future releases include:
  - Explicit-state model-checker
  - MDD-based symbolic model-checking

All of which are “state-of-the-art”

Please direct your attention to the whiteboard.

Timeout Automata[1] (Semantics)

An *explicit* real-time model.

- Vocabulary:
  - A set of state variables.
  - A *global clock*, $c \in \mathbb{R}^{0\leq}$.
  - A set of *timeout* variables T such that for $t \in T$, $t \in \mathbb{R}^{0\leq}$.
- Construct a transition system $\langle S, S^0, \rightarrow\rangle$:
  - States are mappings of all variables to values.
  - Transitions are either *time transitions* or *discrete transitions*.
    - Time transitions are enabled if the clock is less than all timeouts. Updates clock to least timeout.
    - Discrete transitions are enabled if the clock equals some timeout. Updates state variables and timeouts.

[1] B. Dutertre and M. Sorea. Timed systems in SAL. *SRI TR*, 2004.

Disjunctive Invariants

Even with *k*-induction, getting a sufficiently strong invariant is still hard! *Disjunctive invariants* help. A disjunctive invariant can be built iteratively from the counterexamples returned for the hypothesized invariant being verified.

```
t0:  THEOREM system |-
       G(   (    (phase = Settle)
             AND (rstate = tstate + 1)
             AND (rclk - tclk - TPERIOD > 0)
             AND (tclk + TPERIOD + TSTABLE - rclk > 0))
         OR
            (    (phase = Stable)
             AND (rstate = tstate + 1)
             AND (rclk - tclk - TSETTLE > 0)
             AND (tclk + TPERIOD - rclk > 0)
             AND (rdata = tdata))

                         .
                         .
                         .
```