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_ε

Trust

"An entity can be trusted if it always behaves in the expected manner for the intended purpose [1]"

Properties

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

- *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
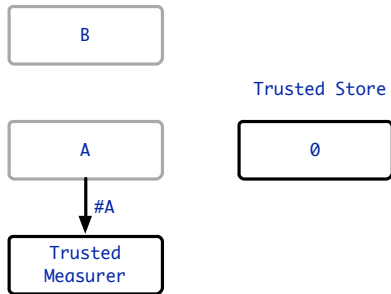
- Start with a measurer and store that is trusted
- Measure software to be launched
- Store the measurement
- Launch the new software
- Repeat until system boot

B

Trusted Store

A

0
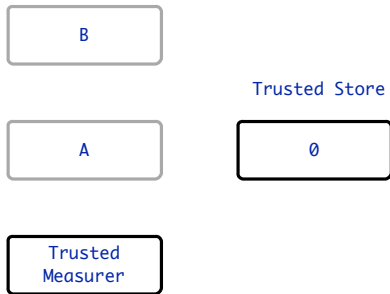
Trusted Measurer

Chaining Measurements

- Start with a measurer and store that is trusted
- Measure software to be launched
- Store the measurement
- Launch the new software
- Repeat until system boot

Chaining Measurements

- Start with a measurer and store that is trusted
- Measure software to be launched
- Store the measurement
- Launch the new software
- Repeat until system boot

- ▶ Start with a measurer and store that is trusted
- ▶ Measure software to be launched
- ▶ Store the measurement
- ▶ Launch the new software
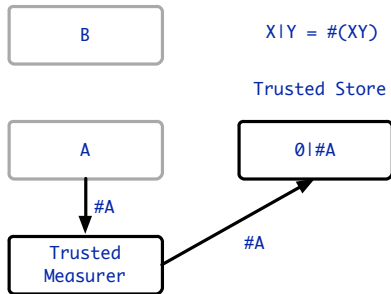- ▶ Repeat until system boot


```
B
```

Trusted Store

```
A
```

```
0|#A
```

```
Trusted
Measurer
```

KU INFORMATION
& TELECOMMUNICATION
TECHNOLOGY CENTER
The University of Kansas

- ▶ Start with a measurer and store that is trusted
- ▶ Measure software to be launched
- ▶ Store the measurement
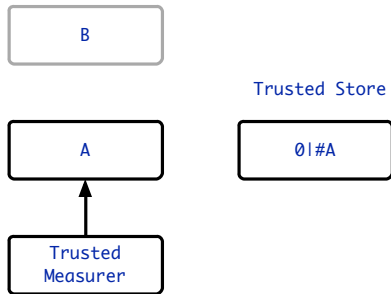- ▶ Launch the new software
- ▶ Repeat until system boot

- Start with a measurer and store that is trusted
- Measure software to be launched
- Store the measurement
- Launch the new software
- Repeat until system boot


```
      ┌─────────────┐
      │      B      │
      └─────────────┘
             ▲
             │
      ┌─────────────┐
      │      A      │
      └─────────────┘

      ┌─────────────┐
      │   Trusted   │
      │   Measurer  │
      └─────────────┘
```
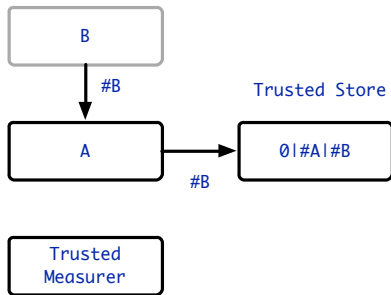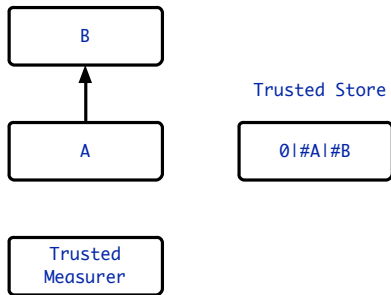
Trusted Store

```
┌─────────────┐
│  0|#A|#B    │
└─────────────┘
```

Assume we know $0|\#A|\#B$ is correct

- $\#A$ and $\#B$ must be correct
  - A and B are the correct binaries
  - A and B include hash and launch functions
- Measurement occurred in the right order
  - $\#(XY) \neq \#(YX)$
  - Trusted store started with 0

A chain exists from Trusted Measurer and Trusted Store to B

- Trusted Measurer and Trusted Store are trusted *a priori*
- A is trusted to be A because its measurement is:
  - Correct
  - Taken by a trusted party (Trusted Measurer)
  - Stored by a trusted party (Trusted Store)
- B is trusted to be B because its measurement is:
  - Correct
  - Taken by a trusted party (A)
  - Stored by a trusted party (Trusted Store)
  - If A's ability to measure B were compromised, $\#A$ would be wrong
- and so on and so on...

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

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

Trusted Boot builds evidence supporting these chains

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

- 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

- 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

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)

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

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
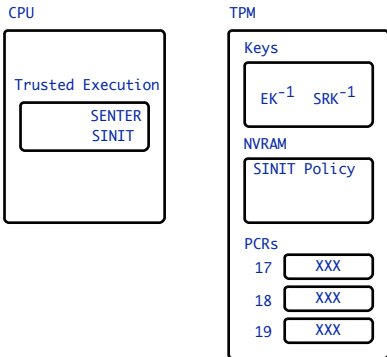- Linking EK to its platform is done by a trusted Certificate Authority (CA)

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

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

CPU

Trusted Execution

SENTER
SINIT

TPM

Keys

EK^{-1} SRK^{-1}

NVRAM

SINIT Policy

PCRs

17   XXX

18   XXX

19   XXX

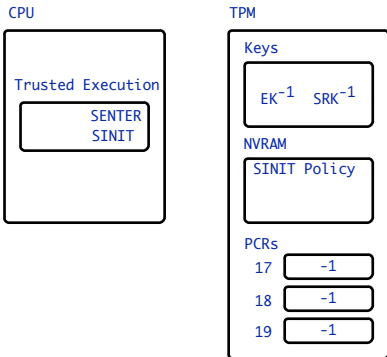Roots of trust are used to build a trusted system from boot.

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

CPU

Trusted Execution
    SENTER
    SINIT

TPM

Keys

EK^{-1} SRK^{-1}

NVRAM

SINIT Policy

PCRs
17   -1
18   -1
19   -1

KU INFORMATION
& TELECOMMUNICATION
TECHNOLOGY CENTER
The University of Kansas

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

CPU

Trusted Execution
  SENTER
  SINIT

TPM

Keys

EK^{-1} SRK^{-1}

NVRAM

SINIT Policy

PCRs
17 0
18 0
19 0

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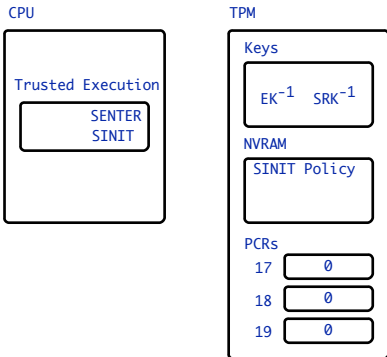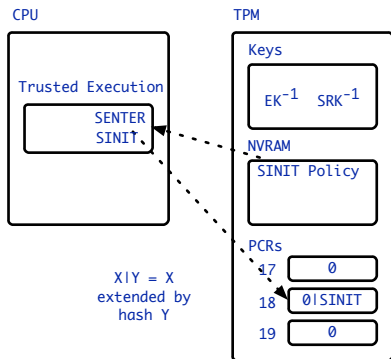
Roots of trust are used to build a trusted system from boot.

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

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.

KU INFORMATION
& TELECOMMUNICATION
TECHNOLOGY CENTER
The University of Kansas

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

MLE

| hypervisor dom0 |

CPU

Trusted Execution

SENTER
SINIT

TPM

Keys

EK^{-1} SRK^{-1}

NVRAM

SINIT Policy

PCRs

17 0|MLE
18 0|SINIT
19 0

X|Y = X extended by hash Y

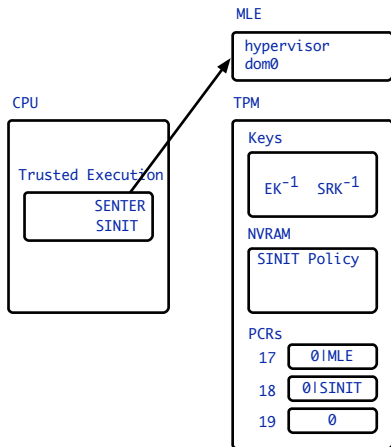KU INFORMATION & TELECOMMUNICATION TECHNOLOGY CENTER
The University of Kansas

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

MLE

| hypervisor |
| dom0 |

CPU

Trusted Execution

| SENTER |
| SINIT |

TPM

Keys

| EK^{-1} SRK^{-1} |

NVRAM

| SINIT Policy |

PCRs

17	0|MLE
18	0|SINIT
19	0

KU INFORMATION
& TELECOMMUNICATION
TECHNOLOGY CENTER
The University of Kansas

- 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 — $0 \mid SINIT \neq -1 \mid SINIT$
- 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

Armored VP

```
vTPM
appraiser
attester
measurer
application
```

TPM

```
Keys

  EK⁻¹  SRK⁻¹

NVRAM
  SINIT Policy

PCRs
 17   0|MLE
 18   0|SINIT
 19    0
```

hypervisor
  dom0

KU INFORMATION & TELECOMMUNICATION TECHNOLOGY CENTER — The University of Kansas

- `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

Armored VP

vTPM
appraiser
attester
measurer
application

hypervisor
dom0

vTPM

vTPM

Keys

EK^{-1} SRK^{-1}

NVRAM

PCRs

k   0
k+1   0
k+2   0

#vTPM

TPM

Keys

EK^{-1} SRK^{-1}

NVRAM

SINIT Policy

PCRs

17   0|MLE
18   0|SINIT
19   0|vTPM

INFORMATION
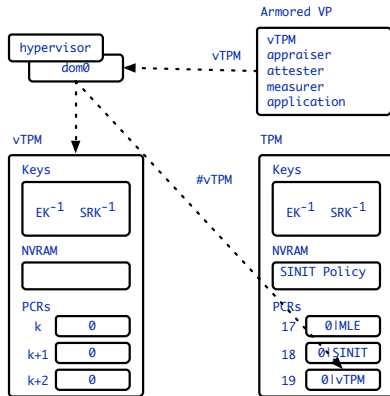& TELECOMMUNICATION
TECHNOLOGY CENTER
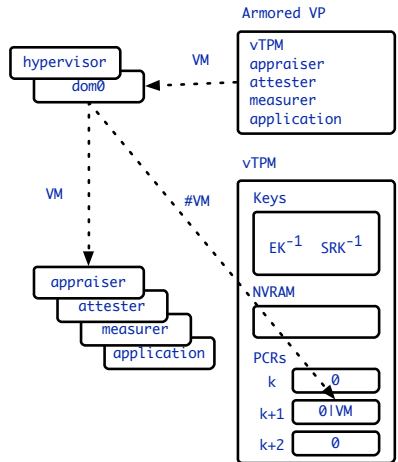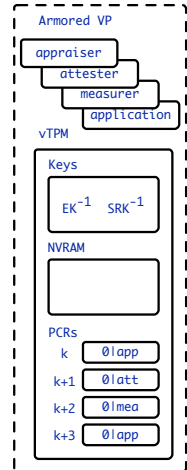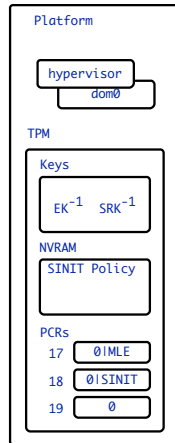The University of Kansas

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

- 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

Platform
- hypervisor
  - dom0

TPM

Keys
- EK^{-1} SRK^{-1}

NVRAM
- SINIT Policy

PCRs
- 17 0|MLE
- 18 0|SINIT
- 19 0

Armored VP
- appraiser
  - attester
    - measurer
      - application

vTPM

Keys
- EK^{-1} SRK^{-1}

NVRAM

PCRs
- k   0|app
- k+1 0|att
- k+2 0|mea
- k+3 0|app

- Trust is transitive
  - $T^x[y] \wedge T^y[z] \Rightarrow T^x[z]$
  - Construct evidence trust chains
  - 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 is the Trusted Measurer
  - RTS is the Trusted Store
  - RTR is the Trusted Reporter (coming soon...)
- Extend chains of trust by measuring before executing

[1] A. Martin et al. The ten page introduction to trusted computing. Technical Report CS-RR-08-11, Oxford University Computing Labratory, Oxford, UK, 2008.