

# 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](mailto:palexand@ku.edu)

Formatted with the Beamer Class for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

## 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

# Necessary Capabilities for Trust

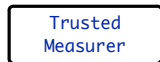
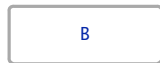
- ▶ *Strong Identification* — An unambiguous, immutable identifier associated with the platform. The identifier is a protected encryption key in the TXT implementation.
- ▶ *Reporting Configuration* — An unambiguous identification mechanism for software and hardware running on the platform. The mechanism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

- ▶  $\#X$  — Hash function such as MD-5
  - ▶  $\#X$  is unique for each  $X$
  - ▶ Guessing  $X$  from  $\#X$  is impossible
- ▶  $\{X\}_Y$  — Encrypting  $X$  with  $Y$ 
  - ▶  $X$  cannot be obtained from  $\{X\}_Y$  without  $Y$
  - ▶  $Y$  cannot be guessed

# Measurement — Chaining Evidence

We would like to start *A* and *B* while gathering evidence of trust

- ▶ Start with a root measurer and store that are trusted *a priori*



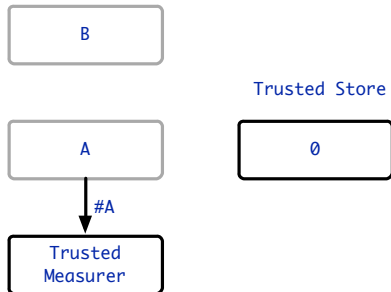
Trusted Store



# Measurement — Chaining Evidence

We would like to start *A* and *B* while gathering evidence of trust

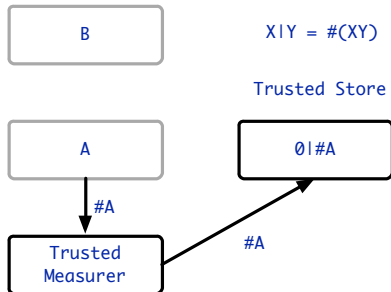
- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched



# Measurement — Chaining Evidence

We would like to start  $A$  and  $B$  while gathering evidence of trust

- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched
- ▶ Store the measurement of the new software

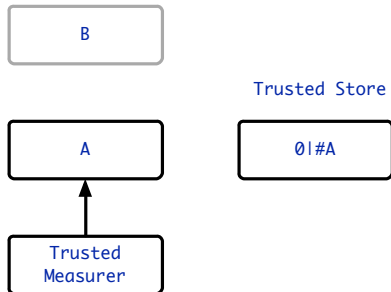




# Measurement — Chaining Evidence

We would like to start  $A$  and  $B$  while gathering evidence of trust

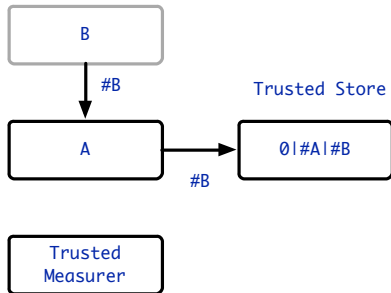
- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched
- ▶ Store the measurement of the new software
- ▶ Launch the new software



# Measurement — Chaining Evidence

We would like to start  $A$  and  $B$  while gathering evidence of trust

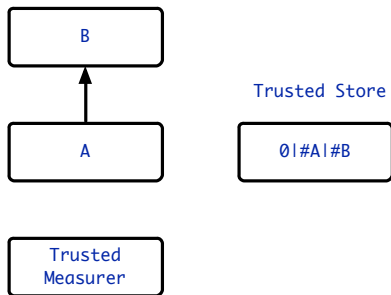
- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched
- ▶ Store the measurement of the new software
- ▶ Launch the new software
- ▶ Repeat for each system software component



# Measurement — Chaining Evidence

We would like to start *A* and *B* while gathering evidence of trust

- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched
- ▶ Store the measurement of the new software
- ▶ Launch the new software
- ▶ Repeat for each system software component



# Appraisal — What Do We Know?

Measurement  $\neq$  trust — Measurements must be appraised

- ▶ Determine if  $0 \mid \#A \mid \#B$  is correct
  - ▶ Calculate a *golden hash* from  $A$  and  $B$
  - ▶ Compare golden hash with  $0 \mid \#A \mid \#B$  from trusted store
  - ▶ Correct  $0 \mid \#A \mid \#B$  implies trusted boot
- ▶ Correct  $0 \mid \#A \mid \#B$  implies  $A$  and  $B$  must be correct
  - ▶ Correct  $0 \mid \#A \mid \#B$  implies  $\#A$  and  $\#B$  are the correct hashes
  - ▶ Correct  $\#A$  and  $\#B$  implies  $A$  and  $B$  are the correct binaries
  - ▶  $A$  includes hash and launch functions
- ▶ Correct  $0 \mid \#A \mid \#B$  implies measurement occurred in the right order
  - ▶  $\#(XY) \neq \#(YX)$
  - ▶ Trusted store started with 0

# Appraisal — But Why Trust B?

A chain exists from the 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 -  $\forall x, y, z \cdot T^x[y] \wedge T^y[z] \Rightarrow T^x[z]$

Measured Boot gathers evidence of 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 its 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)

# 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)

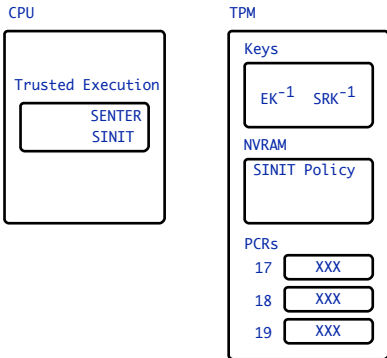
A *Root of Trust for Storage* is trusted to protect stored data

- ▶ 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.

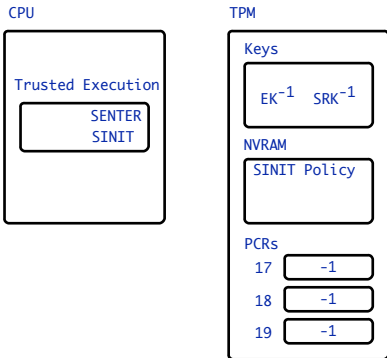
- Power-on reset



# 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



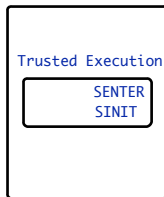


# One Step from Roots of Trust

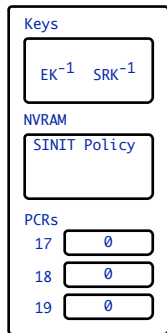
Roots of trust are used to build a trusted system from boot.

- ▶ Power-on reset
- ▶ Resettable PCRs set to -1
- ▶ SENTER called, resets resettable PCRs to 0

CPU



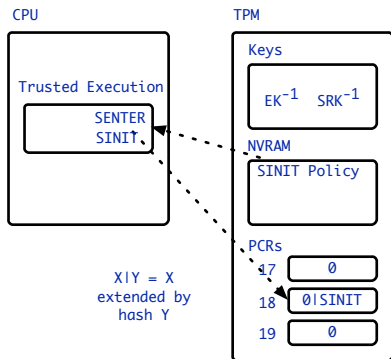
TPM



# 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, 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:

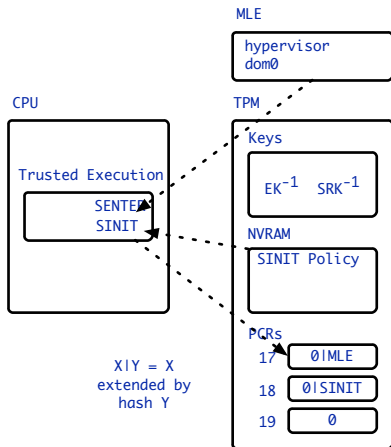
- ▶ SENTER was called — Resetting PCR 18 starts measurements at 0 rather than -1
- ▶ SINIT was measured by SENTER — Only SENTER can extend PCR 18
- ▶ SINIT uses the correct policy — PCR 18 is extended with SINIT measurement policy
- ▶ SENTER ran before SINIT was measured —  $A \mid B \neq B \mid A$

**Measurement  $\neq$  Trust**

Measurements must be appraised to determine trust.

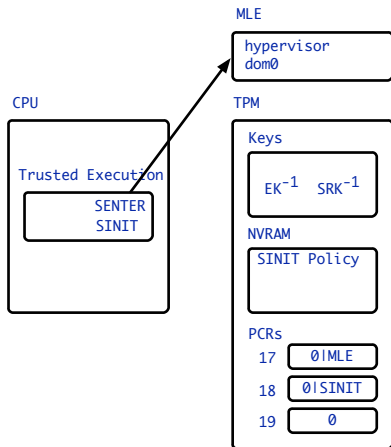
# Two Steps from Roots of Trust

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



# 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 —  $0 \mid SINIT \neq -1 \mid SINIT$
- ▶ MLE is good — Measured by good SINIT into PCR
- ▶ Initial OS is good — Measured by good MLE into PCR

# Boot the MLE

- ▶ SENTER starts the MLE
  - ▶ SENTER starts the hypervisor
  - ▶ SENTER passes dom0 to hypervisor
  - ▶ hypervisor starts dom0



Armored VP

vTPM  
appraiser  
attester  
measurer  
application

TPM

Keys

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

NVRAM

SINIT Policy

PCRs

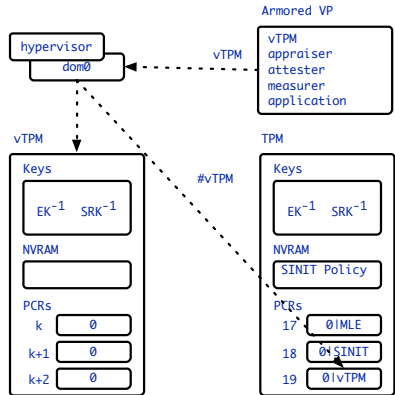
17 0IMLE

18 0ISINIT

19 0

# Boot the MLE

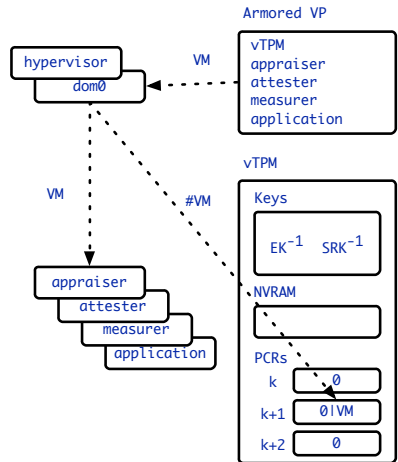
- ▶ SENTER starts the MLE
  - ▶ SENTER starts the hypervisor
  - ▶ SENTER passes dom0 to hypervisor
  - ▶ hypervisor starts dom0
- ▶ dom0 constructs the Armored VP
  - ▶ Measures the vTPM into the TPM
  - ▶ Starts the vTPM



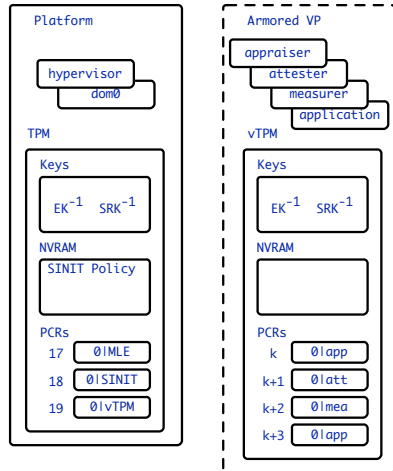


# Boot the MLE

- ▶ 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



- ▶ 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



# What we know from good PCRs

- ▶ The right hypervisor and dom0 started - PCR 17 measurement and we trust SENTER, SINIT and SINIT Policy
- ▶ The right vTPM started - PCR 19 measurement and we trust SENTER, SINIT, and dom0<sup>1</sup>
- ▶ The right ArmoredSoftware components started - vTPM PCRs and we trust dom0 and the vTPM
- ▶ The right application started - vTPM PCRs and we trust dom0 and the vTPM

---

<sup>1</sup>More work for vTPM startup remains

# Chaining Trust (Reprise)

- ▶ 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

A *quote* is a signed data package generated by a TPM used to establish trust

- ▶  $q = \{|\langle n, pcr \rangle|\}_{AIK^{-1}}$ 
  - ▶  $n$  - A nonce or other data
  - ▶  $pcr$  - A PCR composite generated from TPM PCRs
  - ▶  $AIK^{-1}$  - An alias for  $EK^{-1}$  used for signing
- ▶  $AIK$  is a wrapped TPM key usable only in the TPM that generated it
  - ▶  $wrap(AIK, \{pcr\}) = \langle AIK, seal(AIK^{-1}, \{pcr\}) \rangle$
  - ▶  $seal(AIK^{-1}, \{pcr\}) = \{AIK^{-1}\}_{SRK}$  and decrypts only when  $pcr$  matches the TPMs PCRs at decryption time
- ▶ Generated by the TPM with command `TPM_Quote`

Assume that the appraiser is given  $q$  of the form:

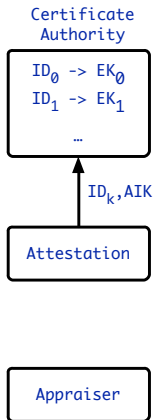
$$q = \{|\langle n, pcr \rangle|\}_{AIK^{-1}}$$

- ▶ Signature check using  $AIK$  — Signature was generated by a TPM with AIK installed
- ▶  $pcr$  check using regenerated composite from desired PCR values — TPM PCRs matched desired PCR values at quote generation time
- ▶  $n$  check by knowing nonce or data values — Nonce provides replay prevention. Other data serves other purposes.

The binding of AIK to the target is missing

We would like to start  $A$  and  $B$  while gathering evidence of trust

- Request AIK certification from a Certificate Authority



We would like to start  $A$  and  $B$  while gathering evidence of trust

- ▶ Request AIK certification from a Certificate Authority
- ▶ CA signs the AIK

Certificate  
Authority

$ID_0 \rightarrow EK_0$

$ID_1 \rightarrow EK_1$

...

$\{IAIK\}_{CA^{-1}}$

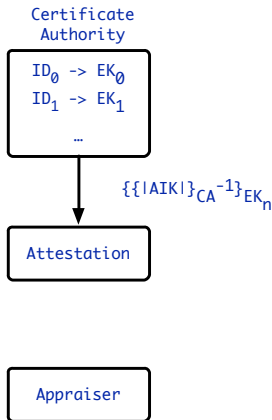
Attestation

Appraiser



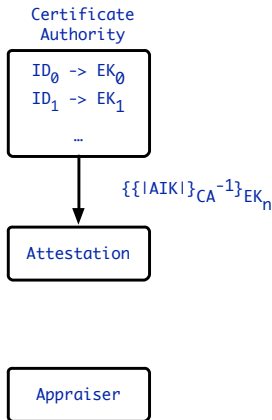
We would like to start *A* and *B* while gathering evidence of trust

- ▶ Request AIK certification from a Certificate Authority
- ▶ CA signs the AIK
- ▶ CA encrypts the signed AIK with the requester's EK



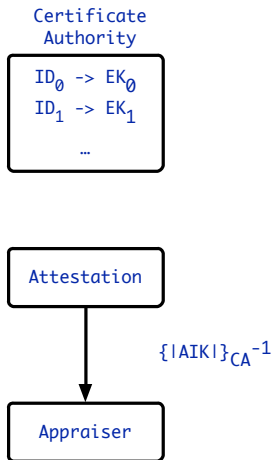
We would like to start *A* and *B* while gathering evidence of trust

- ▶ Request AIK certification from a Certificate Authority
- ▶ CA signs the AIK
- ▶ CA encrypts the signed AIK with the requester's EK
- ▶ CA sends encrypted AIK to requester



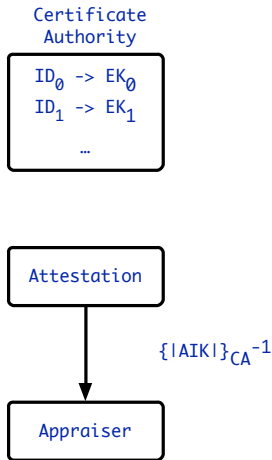
We would like to start  $A$  and  $B$  while gathering evidence of trust

- ▶ Request AIK certification from a Certificate Authority
- ▶ CA signs the AIK
- ▶ CA encrypts the signed AIK with the requester's EK
- ▶ CA sends encrypted AIK to requester
- ▶ Requester decrypts encrypted AIK



We would like to start  $A$  and  $B$  while gathering evidence of trust

- ▶ Request AIK certification from a Certificate Authority
- ▶ CA signs the AIK
- ▶ CA encrypts the signed AIK with the requester's EK
- ▶ CA sends encrypted AIK to requester
- ▶ Requester decrypts encrypted AIK
- ▶ Requester sends signed AIK to appraiser



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