

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 KU Beamer Class for L^AT_EX 2_ε

What is Trust?

“An entity can be trusted if it always behaves in the expected manner for the intended purpose”¹

¹ *The Ten Page Introduction to Trusted Computing* by Andrew Martin

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

Required Capabilities for Establishing Trust

- ▶ *Strong Identification* — An unambiguous, immutable identifier associated with the platform.
- ▶ *Reporting Configuration* — An unambiguous identification mechanism for software and hardware running on the platform.
- ▶ *Reporting Behavior* — A mechanism for observing and reporting execution behavior.

- ▶ $\#X$ — Hash of X
 - ▶ $\#X$ is unique for each X
 - ▶ Guessing X from $\#X$ is impossible
- ▶ $\{X\}_Y$ — Encrypt X with Y
 - ▶ X cannot be obtained from $\{X\}_Y$ without Y
 - ▶ Guessing X from $\{X\}_Y$ is impossible
 - ▶ Guessing Y is impossible
- ▶ $[X]_{Y^{-1}}$ — Sign X with Y^{-1}
 - ▶ $[X]_{Y^{-1}}$ is unique for every X and Y^{-1} pair
 - ▶ Guessing $[X]_{Y^{-1}}$ from X is impossible
- ▶ $M \mid \#X$ — Extend M with $\#X$
 - ▶ Concatenate M with $\#X$ and hash the result
 - ▶ Ideal $M \mid \#X$ unique for M and X

- ▶ $(X, \{X^{-1}\}_{Y^{-1}})$ — Wrap X with Y^{-1}
 - ▶ Can use X for encryption and signature checking
 - ▶ Cannot use X^{-1} for decryption or signing without Y^{-1}
- ▶ $(D, \{C\})$ — Seal D to configuration C
 - ▶ D is not available if system is not in configuration C
 - ▶ Usually accompanied by encryption
- ▶ $(\{SK\}_K, \{D\}_{SK})$ — Envelope D with K
 - ▶ Encrypt large data D with session key SK
 - ▶ Encrypt SK with K
 - ▶ D behaves as if encrypted with K
- ▶ $[[A, B]]_{Y^{-1}}$ — Certify binding of A and B with Y^{-1}
 - ▶ Y signs (A, B) with private key Y^{-1}
 - ▶ Certificate is checked using Y
 - ▶ Valid signature provides evidence A and B are bound together

Wrapping A Key

$$\text{wrap}(X, Y) = (X, \{X^{-1}\}_{Y^{-1}})$$

- ▶ X^{-1} is encrypted with Y^{-1} while X is clear
- ▶ $\{D\}_X$ and checking $[D]_{X^{-1}}$ may be done without Y
- ▶ Decrypting $\{D\}_X$ and generating $[D]_{X^{-1}}$ require Y

Chaining Keys

$$(X_0, \{X_0^{-1}\}_{X_1^{-1}}), (X_1, \{X_1^{-1}\}_{X_2^{-1}}) \dots (X_{n-1}, \{X_{n-1}^{-1}\}_{X_n^{-1}})$$

- ▶ Each key depends on the previous key
- ▶ If the root key is trustworthy the chain is trustworthy

Sealing to State

$(D, \{C\})$ — Seal D to configuration C

- ▶ D is protected by a key or other mechanism
- ▶ C describes an acceptable system state
- ▶ D cannot be accessed if system is not in state C
- ▶ Used to protect data even when system is mis-configured

Enveloping

$$\text{envelope}(K, D) = (\{SK\}_K, \{D\}_{SK})$$

- ▶ SK is a symmetric session key for bulk encryption
- ▶ D is encrypted with SK
- ▶ SK is encrypted with K
- ▶ D is protected as if encrypted with K

Certificates and Certification

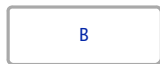
$$[[(A, B)]]_{Y^{-1}} = [(A, B)]_{Y^{-1}}$$

- ▶ (A, B) associates A with B
- ▶ Y certifies the association by signing with Y^{-1}
- ▶ Certificate is checked using Y
- ▶ If we trust Y , then we trust the binding of A to B

Chaining Measurement — Gathering Evidence

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

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



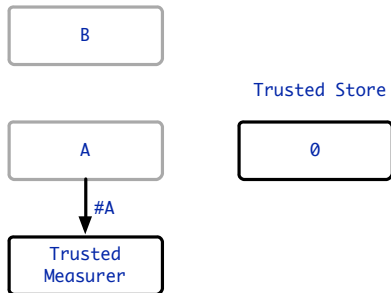
Trusted Store



Chaining Measurement — Gathering Evidence

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

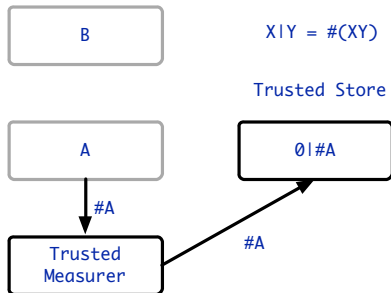
- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched



Chaining Measurement — Gathering Evidence

We would like to start A and B while gathering evidence for determining trust

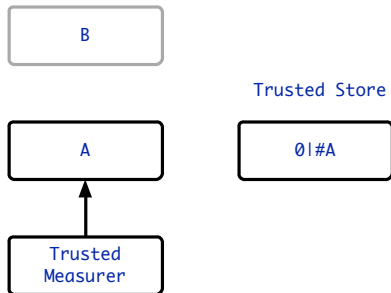
- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched
- ▶ Store the measurement of the new software



Chaining Measurement — Gathering Evidence

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

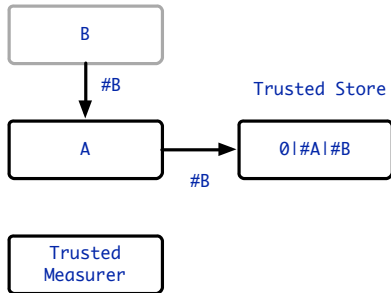
- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched
- ▶ Store the measurement of the new software
- ▶ Launch the new software



Chaining Measurement — Gathering Evidence

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

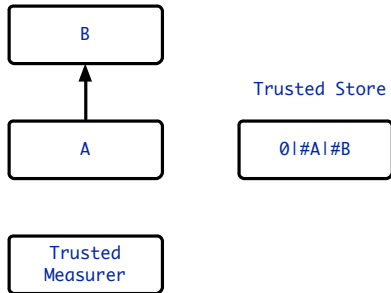
- ▶ Start with a root measurer and store that are trusted *a priori*
- ▶ Measure the new software to be launched
- ▶ Store the measurement of the new software
- ▶ Launch the new software
- ▶ Repeat for each system software component



Chaining Measurement — Gathering Evidence

We would like to start *A* and *B* while gathering evidence for determining 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 by definition 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 to check trust relationships.

A *chain of trust* from X_0 to X_n :

$$T^{X_0}[X_1] \wedge T^{X_1}[X_2] \wedge \dots \wedge T^{X_{n-1}}[X_n]$$

- ▶ If X_0 is trusted, then X_n is trusted
- ▶ X_0 is called a *root-of-trust*
- ▶ Establishing trust chains defines a framework for measurement
- ▶ Measurement provides evidence that trust chains are not violated
- ▶ Appraisal checks evidence to assess trust 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 like OS security rings
 - ▶ Resettable — PCR can be reset to known value after *SENTER*
 - ▶ Many others that we don't need yet

Non-Resettable

- ▶ History since reboot
- ▶ Reset only on reboot
- ▶ Good for recording trajectory

Resettable

- ▶ No history
 - ▶ Reset before use
 - ▶ Good for one-off user data
-
- ▶ Reset requires appropriate permissions based on locality
 - ▶ A PCR is resettable if defined in platform spec

Locality = Access Control for PCRs

- ▶ Each PCR is assigned a locality
- ▶ Only processes with locality greater than or equal a PCR's locality may modify it
- ▶ Increases monotonically starting at SENTER invocation

Locality and What Runs There

| <i>Locality</i> | <i>Purpose</i> |
|-----------------|-------------------------------|
| 4 | Trusted Hardware/SINIT Policy |
| 3 | Other MLE Components |
| 2 | Operating System |
| 1 | Applications Static |
| 0 | RTM/Legacy |

Locality Rules of Thumb

- ▶ Only SENTER runs in locality 4
- ▶ Only SINIT runs in locality 3
- ▶ OS and core system run in locality 2
- ▶ Applications run in locality 1
- ▶ Locality 0 is rarely used

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)

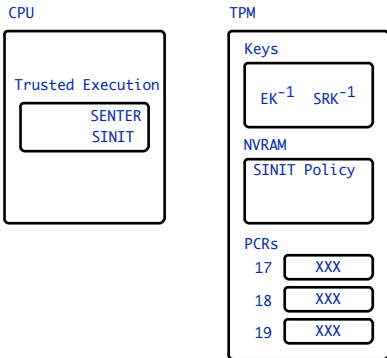
A *Root of Trust for Storage* is trusted to protect stored data

- ▶ 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.

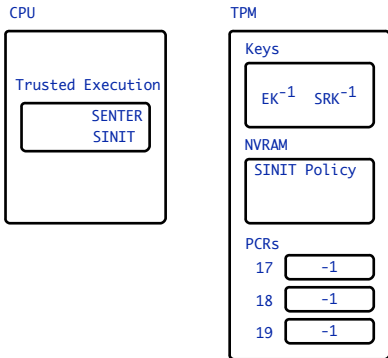
- Power-on reset



One Step from Roots of Trust

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

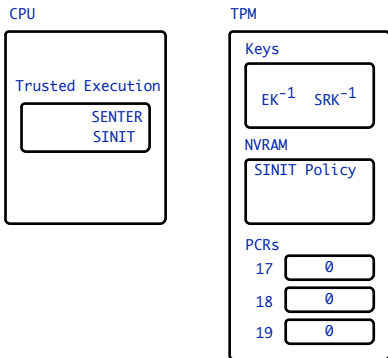
- ▶ Power-on reset
- ▶ Resettable PCRs set to -1



One Step from Roots of Trust

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

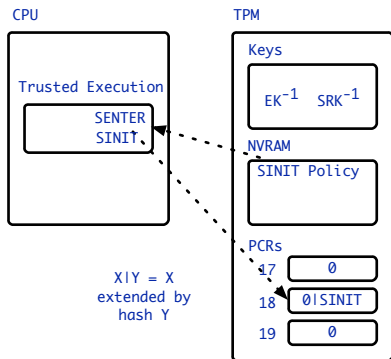
- ▶ Power-on reset
- ▶ Resettable PCRs set to -1
- ▶ SENTER called, resets resettable PCRs to 0



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:

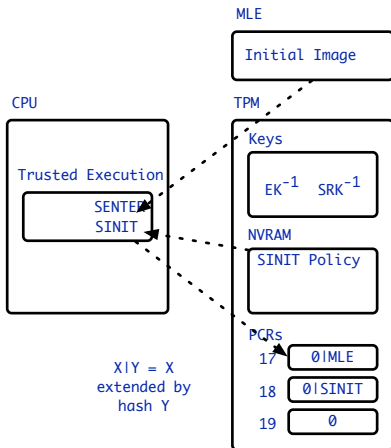
- ▶ SENTER was called — Resetting PCR 18 starts measurements at 0 rather than -1
- ▶ SINIT was measured by SENTER — Only SENTER can extend PCR 18
- ▶ SINIT uses the correct policy — PCR 18 is extended with SINIT measurement policy
- ▶ SENTER ran before SINIT was measured — $A \mid B \neq B \mid A$

Measurement \neq Trust

Measurements must be appraised to determine trust.

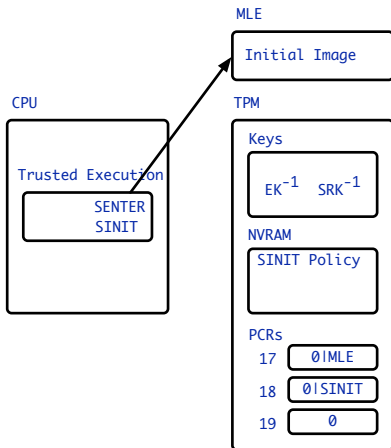
Two Steps from Roots of Trust

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



Two Steps from Roots of Trust

- ▶ SINIT measures the Measured Launch Environment (MLE) using measured SINIT 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

Boot is generic until the MLE starts

MLE is the beginnings of the operating system.

Boot the MLE

- ▶ SENTER starts the MLE
 - ▶ SENTER starts the initial image
 - ▶ Initial image starts the system

initial image

Operating System

kernel
file system
drivers
run-time trust
applications

TPM

Keys

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

NVRAM

SINIT Policy

PCRs

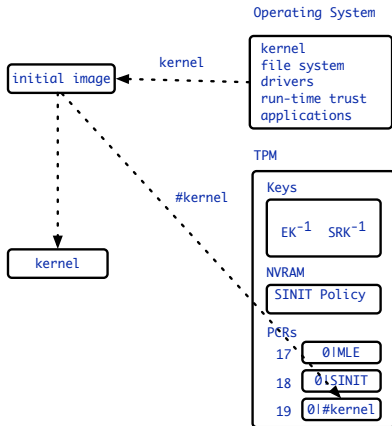
17 0IMLE

18 0ISINIT

19 0

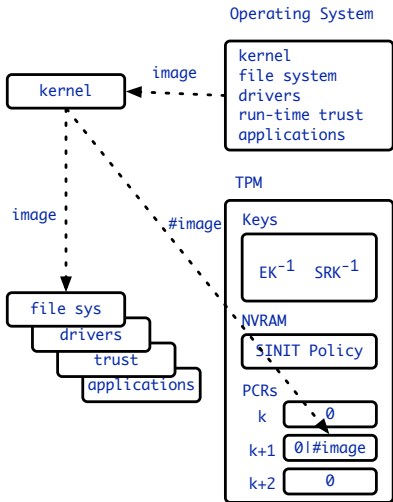
Boot the MLE

- ▶ SENTER starts the MLE
 - ▶ SENTER starts the initial image
 - ▶ Initial image starts the system
- ▶ Initial image initializes the kernel
 - ▶ Measures the kernel into the TPM
 - ▶ Starts the kernel



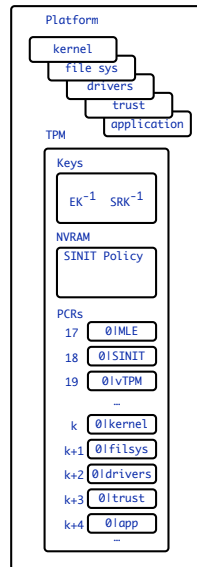
Boot the MLE

- ▶ SENTER starts the MLE
 - ▶ SENTER starts the initial image
 - ▶ Initial image starts the system
- ▶ Initial image initialized the kernel
 - ▶ Measures the kernel into the TPM
 - ▶ Starts the kernel
- ▶ Kernel boots the system
 - ▶ Measures remaining images into the TPM
 - ▶ Starts remaining images
 - ▶ Measures application into the TPM
 - ▶ Starts the application



Boot the MLE

- ▶ **SENDER starts the MLE**
 - ▶ SENDER starts the initial image
 - ▶ Initial image starts the system
- ▶ **Initial image initialized the kernel**
 - ▶ Measures the kernel into the TPM
 - ▶ Starts the kernel
- ▶ **Kernel boots the system**
 - ▶ Measures remaining images into the TPM
 - ▶ Starts remaining images
 - ▶ Measures application into the TPM
 - ▶ Starts the application



Built from Good Parts

We can construct a proof that the platform is constructed correctly from PCR contents

- ▶ SINIT measured the right initial image - PCR 18 measurement and we trust SENTER
- ▶ The right initial image started - PCR 17 measurement and we trust SENTER, SINIT and SINIT Policy is measured
- ▶ The right kernel started - PCR 19 measurement and we trust SENTER, SINIT, and initial image is measured
- ▶ The right system components started - PCRs and the kernel is measured
- ▶ The right application started - TPM PCRs and the kernel is measured

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
 - ▶ SENTER/SINIT is the Trusted Measurer
 - ▶ SRK and TPM is the Trusted Storage Root (Unused so far)
 - ▶ EK and TPM is the Trusted Reporter (Coming next)
- ▶ 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 instead of EK^{-1}
- ▶ Generated by the TPM with command TPM_Quote

An *AIK* is A wrapped TPM key bound to an EK^{-1} usable only in the TPM that generated it in the right state

- ▶ $\{(AIK^{-1}, \{pcr\})\}_{EK^{-1}}$ - the AIK^{-1} encrypted with EK^{-1} and sealed to *pcr* values.
- ▶ $\{(AIK^{-1}, \{pcr\})\}_{EK^{-1}}$ decrypts and installs only when
 - ▶ *pcr* matches the TPM's PCRs at decryption time
 - ▶ EK^{-1} is the TPM's endorsement key
- ▶ Protected by a combination of encryption and state
- ▶ Generated by a TPM

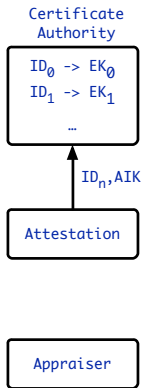
Given q of the form:

$$q = [\langle n, pcr \rangle]_{AIK-1}$$

1. Signature check using AIK verifies authenticity
 - ▶ Signature was generated by a TPM with AIK installed
 - ▶ Appraiser must know AIK
2. pcr check verifies built from good parts in the right order
 - ▶ Compare PCR composite to known good PCR composite
 - ▶ Composite generated from desired golden PCR values
3. Nonce check guarantees freshness
 - ▶ Nonce is random and known to the appraiser
 - ▶ Sent to the target during appraisal

Assume a trusted Certificate Authority (CA) that maintains links from ID to EK with well-known public key CA

- ▶ ID_n requests AIK certification from CA



Assume a trusted Certificate Authority (CA) that maintains links from ID to EK with well-known public key CA

- ▶ ID_n requests AIK certification from CA
- ▶ CA signs AIK with CA^{-1}

Certificate
Authority

$ID_0 \rightarrow EK_0$
 $ID_1 \rightarrow EK_1$
...

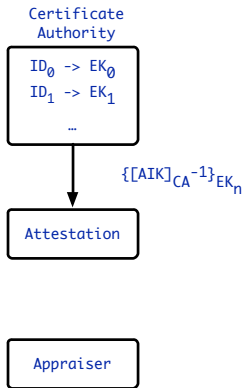
$[AIK]_{CA^{-1}}$

Attestation

Appraiser

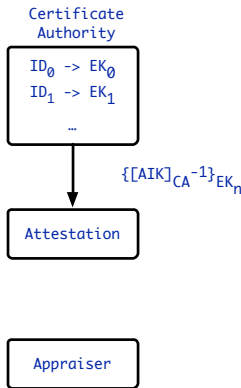
Assume a trusted Certificate Authority (CA) that maintains links from ID to EK with well-known public key CA

- ▶ ID_n requests AIK certification from CA
- ▶ CA signs AIK with CA^{-1}
- ▶ CA encrypts $[AIK]_{CA^{-1}}$ with ID_n 's EK_n



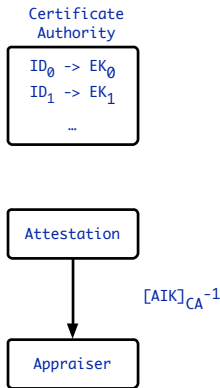
Assume a trusted Certificate Authority (CA) that maintains links from ID to EK with well-known public key CA

- ▶ ID_n requests AIK certification from CA
- ▶ CA signs AIK with CA^{-1}
- ▶ CA encrypts $[AIK]_{CA^{-1}}$ with ID_n 's EK_n
- ▶ CA sends $\{[AIK]_{CA^{-1}}\}_{EK_n}$ to ID_n



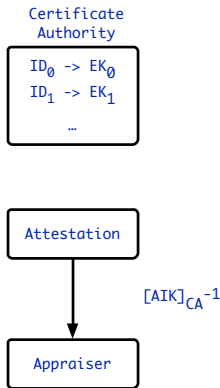
Assume a trusted Certificate Authority (CA) that maintains links from ID to EK with well-known public key CA

- ▶ ID_n requests AIK certification from CA
- ▶ CA signs AIK with CA^{-1}
- ▶ CA encrypts $[AIK]_{CA^{-1}}$ with ID_n 's EK_n
- ▶ CA sends $\{[AIK]_{CA^{-1}}\}_{EK_n}$ to ID_n
- ▶ ID_n decrypts encrypted AIK with EK_n^{-1}



Assume a trusted Certificate Authority (CA) that maintains links from ID to EK with well-known public key CA

- ▶ ID_n requests AIK certification from CA
- ▶ CA signs AIK with CA^{-1}
- ▶ CA encrypts $[AIK]_{CA^{-1}}$ with ID_n 's EK_n
- ▶ CA sends $\{[AIK]_{CA^{-1}}\}_{EK_n}$ to ID_n
- ▶ ID_n decrypts encrypted AIK with EK_n^{-1}
- ▶ ID_n sends $[AIK]_{CA^{-1}}$ to appraiser



Why Believe AIK Belongs to ID_n ?

Cryptographic evidence ensures AIK is an alias for the right EK

- ▶ Only the CA can generate $[AIK]_{CA^{-1}}$
- ▶ CA is trusted to know $ID_n \rightarrow EK_n$
- ▶ CA is trusted to generate $\{[AIK]_{CA^{-1}}\}_{EK_n}$
- ▶ Only ID_n can decrypt $\{[AIK]_{CA^{-1}}\}_{EK_n}$
- ▶ Appraiser can check $[AIK]_{CA^{-1}}$ to ensure use of trusted CA
- ▶ If Appraiser can use AIK then it was decrypted by ID_n

AIK is now a certified alias for EK used for signing

Protocol notation specifies communication:

Sender \rightarrow *Receiver* : *Message*

Key Certification Protocol

$ID_n \rightarrow CA : AIK$

$CA \rightarrow ID_n : \{[AIK]_{CA^{-1}}\}_{EK_n}$ (1)

$ID_n \rightarrow App : [AIK]_{CA^{-1}}, [\langle n, pcr \rangle]_{AIK^{-1}}$

Boot and appraise remote system for covert data acquisition

- ▶ Target reset triggers target boot initialization
- ▶ Target initial boot goes through BIOS and device startup
- ▶ Target secondary boot establishes comm link and “phones home”
- ▶ Appraiser evaluates target
- ▶ Appraiser responds to target with OS image
- ▶ Target boots OS image
- ▶ Target begins operation
- ▶ Appraiser evaluates target
- ▶ Target begins data acquisition and transmission

General assumptions:

- ▶ Nonces, keys and hashes cannot be guessed
- ▶ Perfect cryptography
- ▶ All messages are carried by the adversary

System specific assumptions:

- ▶ Secure communication subsystem is trustworthy
- ▶ Target knows what to communicate with
- ▶ TPM present on the target
- ▶ AIK is established and certified prior to system launch

Designing A Trusted System

What assets should be protected and how?

- ▶ What assets must be handled confidentially?
- ▶ What assets must be handled with integrity?
- ▶ What assets and behaviors must be appraised?
- ▶ What behaviors must be prevented?

Some initial observations:

- ▶ Initial boot software - integrity, must be appraised
- ▶ Communication addresses - confidentiality
- ▶ Hardware and tamper protections - integrity, must be appraised
- ▶ Operating system - integrity, must be appraised
- ▶ Data transmission key - confidentiality

Trust is:

- ▶ Strong identification
- ▶ Direct observation of good behavior
- ▶ Indirect observation of good behavior by a trusted third party

Late Launch is:

- ▶ Initial measurement taken and stored by roots of trust for measurement and storage
- ▶ Reporting performed by root of trust for reporting
- ▶ Trust chains transitively from roots of trust outward
- ▶ System is constructed of good parts

- ▶ Root of trust for measurement
 - ▶ SENTER for launch
 - ▶ Initial measurement taken by SINIT
 - ▶ Hardware-based TPM initialization
- ▶ Root of trust for storage
 - ▶ TPM storage root key (SRK)
 - ▶ Locality enforcement
 - ▶ TPM separation
- ▶ Root of trust for reporting
 - ▶ TPM endorsement key (EK)
 - ▶ TPM separation

Design Decisions

- ▶ What are measurement responsibilities?
- ▶ What is in the MLE?
- ▶ Where are measurements stored?
- ▶ How is locality assigned?

Measurement Responsibilities

- ▶ SENTER measures SINIT policy
- ▶ SINIT measures the Measured Launch Environment (MLE)
- ▶ Initial boot measures hardware and Secondary boot software
- ▶ Secondary boot measures OS
- ▶ OS measures components as they start

Measurement Relation

SENER → SINIT → MLE → OS → app

- ▶ No measurement loops
- ▶ Everything is measured

Traditional Measurement Storage

| <i>PCR</i> | <i>Contents</i> |
|------------|---|
| 0–15 | Static RTM |
| 16 | Debug |
| 17 | Locality 4 measurements by SENTER |
| 18 | Locality 3 measurements by SINIT |
| 19 | Locality 2 measurements by MLE and OS |
| 20 | Locality 1 measurements by applications |
| 21–22 | T/OS controlled |
| 23 | Application specific measurements |

Need a PCR?

- ▶ Available non-resettable PCRs are 8-15
- ▶ Available resettable PCRs are 16,23

What is in the MLE?

- ▶ Anything measured by SINIT is in the MLE by definition
- ▶ MLE could be the initial boot image or the entire boot image
- ▶ How much granularity desired in measurements?

What is in the MLE?

- ▶ Initial boot image
- ▶ Nothing more

Where Are Measurements Stored?

- ▶ Initial measurement storage location is standard
- ▶ Typically 1 hash per measurement, but could be a sequence

PCRs 17 and 18

- ▶ SINIT Policy measured by SENTER into PCR 17
- ▶ MLE Measured by SINIT into PCR 18
- ▶ PCRs 17 & 18 are non-resettable

Where Are Measurements Stored?

- ▶ Later measurements must be designed
- ▶ One big hash, many hashes, one PCR, many PCRs

PCRs 19 and 20

- ▶ OS measured into PCR 19
- ▶ Application(s) measured into PCR 20

What are the Measurements?

- ▶ Measure the OS
 - ▶ OS measured as one hash
 - ▶ OS measured as a hash sequence by the OS startup
- ▶ Applications measured as one hash each
- ▶ Ordering application measurement?
 - ▶ Enforce application hash order by sequencing startup
 - ▶ Use multiple PCRs
 - ▶ Produce multiple good hashes in appraiser
- ▶ Measurement *must* be performed as components are started

Measurement Options and Appraisal

- ▶ **Measurement granularity**
 - ▶ One hash says good or bad for all system binaries and is simple to take
 - ▶ Many hashes extending a PCR says good or bad for all system binaries including order
- ▶ **One or Many PCRs**
 - ▶ One PCR says good or bad for all system binaries with order
 - ▶ Many PCRs says good or bad for individual system binaries without order
- ▶ **Resettable or Non-Resettable**
 - ▶ Resettable ignores history prior to reset
 - ▶ Non-resettable captures history from system startup

Where are Measurements Stored? (Redux)

- ▶ Poky Linux is measured as one hash into PCR 19
 - ▶ Need to know good or bad on startup
 - ▶ Using Poky Linux unmodified
- ▶ Application is measured as one hash into PCR 20
 - ▶ Need to know good or bad on startup
 - ▶ Only one application

PCRs 19 and 20

- ▶ OS measured into PCR 19 by secondary boot prior to start
- ▶ Application(s) measured into PCR 20 by OS prior to start

How is Locality Assigned?

- ▶ No reason to deviate from standard locality mapping
- ▶ No reason to reset PCRs

Locality Assignment

- ▶ PCR 17 - Locality 4, non-resettable
- ▶ PCR 18 - Locality 3, non-resettable
- ▶ PCR 19 - Locality 2, non-resettable
- ▶ PCR 20 - Locality 1, non-resettable

What The Appraiser Knows

- ▶ On initial measurement after startup
 - ▶ 0x00..0 | #SINITPolicy SENTER and SINIT ran
 - ▶ 0x00..0 | #MLE SINIT measured the MLE
 - ▶ MLE was correct on startup
- ▶ On measurement after application startup
 - ▶ 0x00..0 | #OS OS was correct on startup and was started by MLE
 - ▶ 0x00..0 | #app Application was correct on startup and was started by OS

Trusted \neq Secure

- ▶ The Appraiser knows what it is talking to
- ▶ Nothing is protected