KU INSTITUTE FOR INFORMATION SCIENCES
The University of Kansas

Trusted Systems Virtual Institute

Predictable and Scalable Remote Attestation

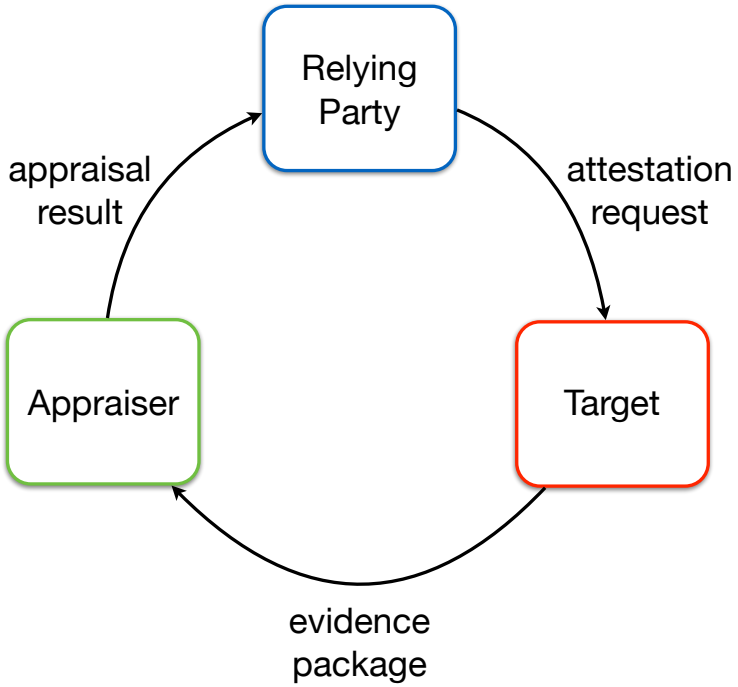
Perry Alexander, Adam Petz, Will Thomas, Logan Schmalz, Sarah Johnson
Institute for Information Sciences
The University of Kansas
{palexand,ampetz,30willthomas,loganschmalz,sarahjohnson}@ku.edu

Predictable and Scalable Remote Attestation

- Evidence and Time - A semantics of evidence over time that allows predictions about the effectiveness of attestation evidence in appraising systems
- Flexible Mechanisms at Scale - A semantics for appraisal architectures and its realization as a collection of reusable attestation components and tools for static analysis.
- Empirical Case Studies - Large scale empirical studies of defining, implementing, and running attestation architectures with applications in supply chain and zero trust.

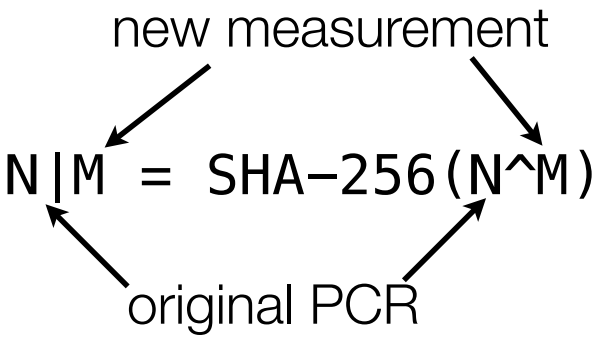
Semantic Remote Attestation

- Relying Party requests appraisal
  - specifies needed information
  - provides a fresh nonce
- Target gathers and generates evidence
  - measures OS & applications
  - generates cryptographic signatures
- Appraiser assesses evidence
  - good application behavior
  - infrastructure trustworthiness
  - good nonce



Extending a PCR

- PCRs contain measurements
  - SHA-256 hashes of images and data
  - may be more sophisticated
- PCRs are extended rather than set
  - SHA-256 of the PCR concatenated with a new measurement
  - captures the new value, original value, and order
- Records the state of a system and trajectory of states
  - used in attestation to evaluate system state
  - used to seal secrets to system state

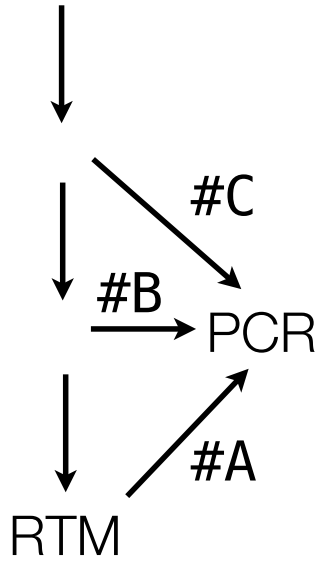


order matters!

$N|M \neq M|N$

Measurement to Record Boot Trajectory

- SENTER resets the TPM
  - PCRs initialize to **#FFFF**
  - SENTER resets PCRs to **#0000**
- RTM measures and starts A
  - measures A
  - extends PCR with #A
  - starts A
- A measures and starts B
  - measures B
  - extends PCR with #B
  - starts B
- B measures and starts C
  - measures C
  - extends PCR with #C
  - starts C
- C is running with PCR indicating startup status
  - record of what binaries started
  - record of startup order
  - *TPM can seal secrets to the PCR value*



PCR:=0|#A|#B|#C

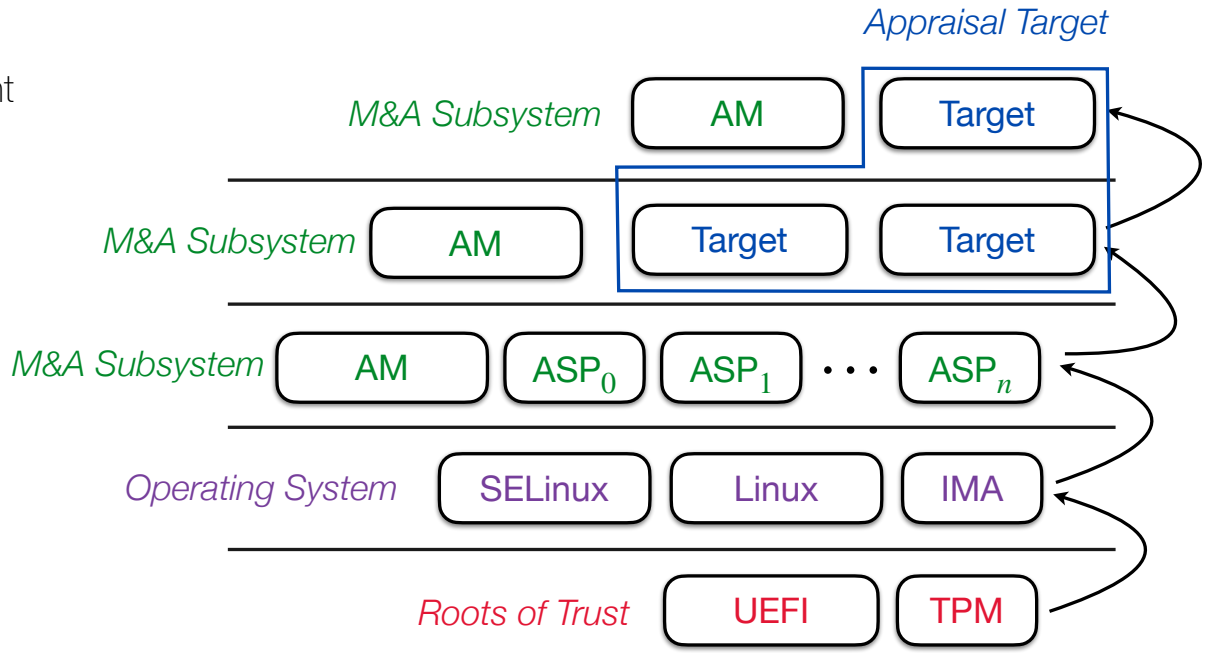
PCR:=0|#A|#B

PCR:=0|#A

PCR:=0

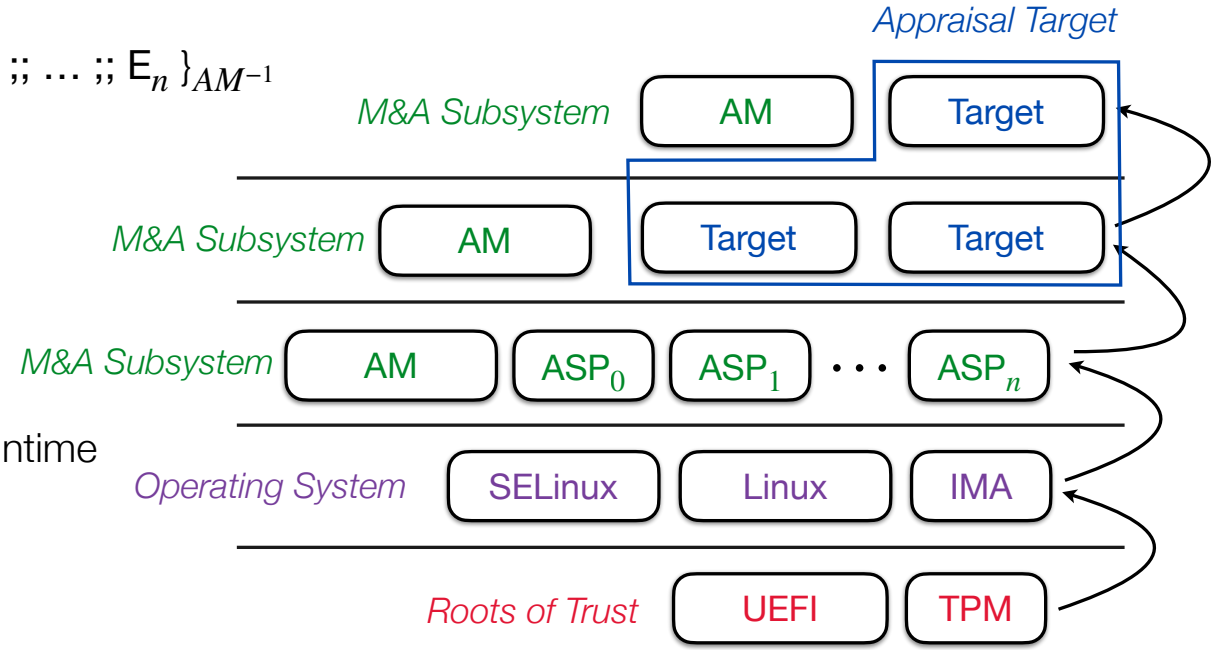
Layered Runtime Attestation

- Target
  - system to be appraised at runtime
  - potentially with component targets
  - cross domain system for this experiment
- M&A Subsystem
  - MAESTRO attestation manager (AM)
  - unique attestation manager key (AM^{-1})
  - attestation service providers (ASPs)
  - Copland attestation protocol
- Operating System
  - RedHat Linux
  - SELinux
  - IMA
- Roots of Trust
  - storage and reporting (TPM)
  - measurement (UEFI)



Layered Runtime Attestation

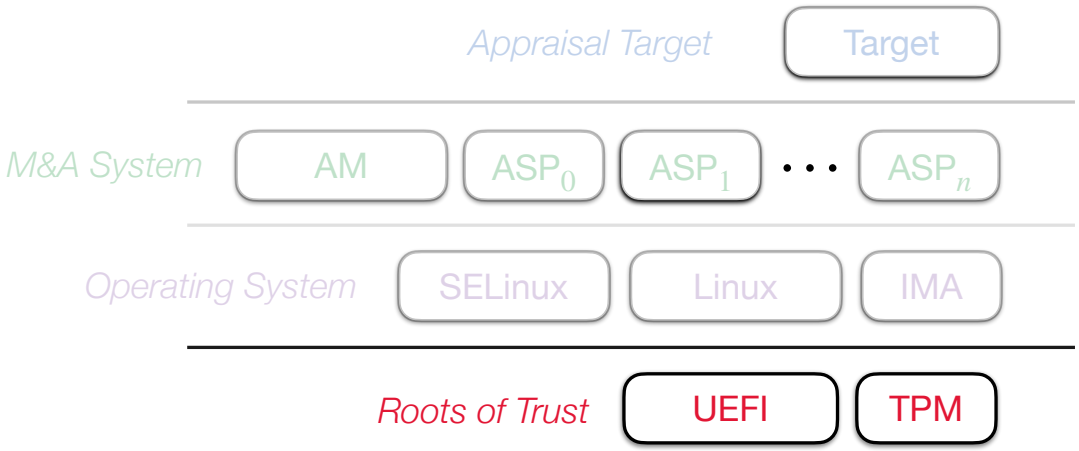
- $\{ E_0 ;; \{ E_{10} ;; \{ E_{20} \}_{AM^{-1}} ;; \ldots ;; E_{1n} \}_{AM^{-1}} ;; \ldots ;; E_n \}_{AM^{-1}}$
  - E_k - evidence from ASP execution
  - ;; - bundling operator from protocol indicating order
  - $\{\ldots\}_{AM^{-1}}$ - Attestation Manager signature
- Layering results in nested evidence
  - ASP invokes an attestation manager - @AM[…]
  - AM produces a signed evidence package
  - MAESTRO enables arbitrary nesting
- AM^{-1} signing key indicates transition to runtime
  - signing memorializes good boot
  - keys must be unique
  - only an AM can access its key
  - access allowed only on good AM startup



proper bundling
∧ satisfies appraisal policy
∧ valid signatures
} ⇒ trustworthy target

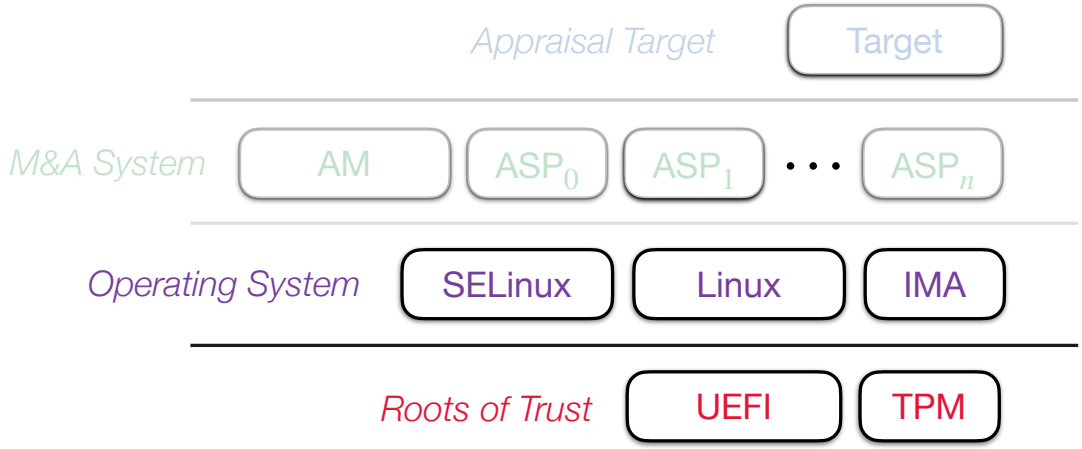
Roots of Trust Base

- TPM
  - Root of trust for Storage and Reporting
  - trusted *a priori*
  - evidence signing
  - generates, stores and seals AM's signing key
  - binds signing key to an AM
- UEFI
  - Root of trust for Measurement
  - trusted *a priori*
  - bootloader measurement and initiation



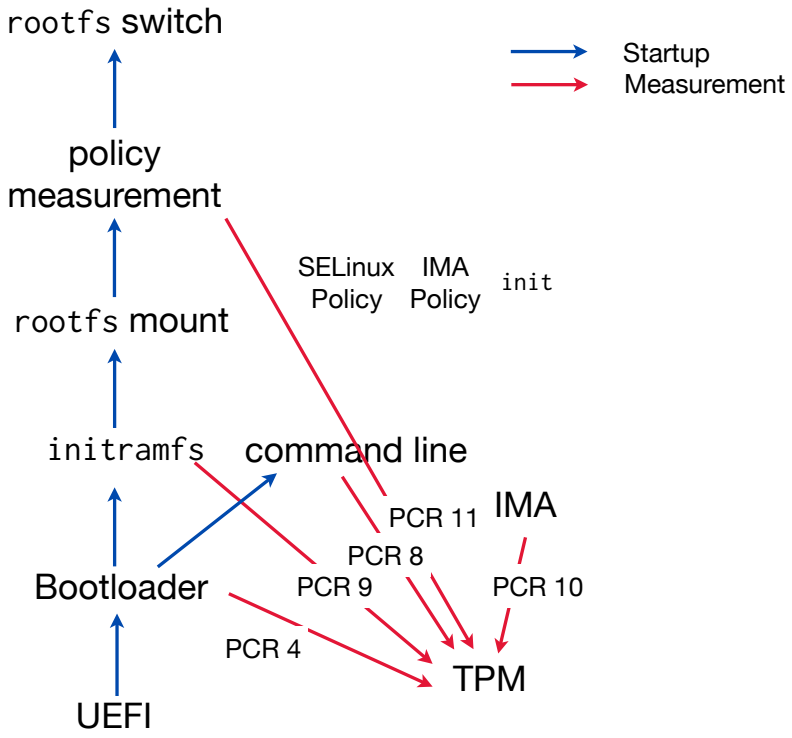
Operating System Layer

- Measure and start Linux
- Measure policy and start SELinux
- Measure policy and start IMA



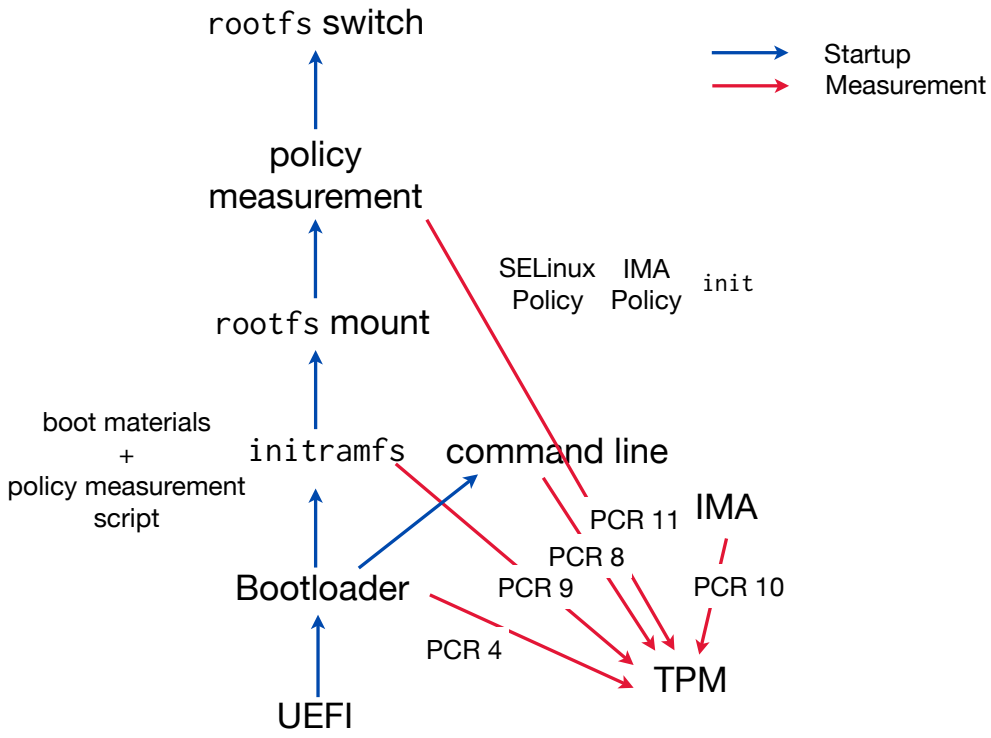
Trusted OS Infrastructure

- Firmware measures and starts boot loader
  - firmware hashes and starts boot loader (PCR 4)
- `initramfs` contents
  - traditional boot materials
  - custom measurement script for SELinux and IMA policies and `init` system
  - IMA will use SELinux types requiring early policy measurement and SELinux start
- Boot `initramfs`
  - bootloader hashes command line to start `initramfs` (PCR 8)
  - bootloader hashes and starts `initramfs` (PCR 9)
- Switch to `rootfs`
  - mount `rootfs`
  - hash IMA and SELinux policies (PCR 11)
  - hash `init` binary
  - execute `init` binary on `rootfs`
  - kernel running with measured IMA and SELinux policies



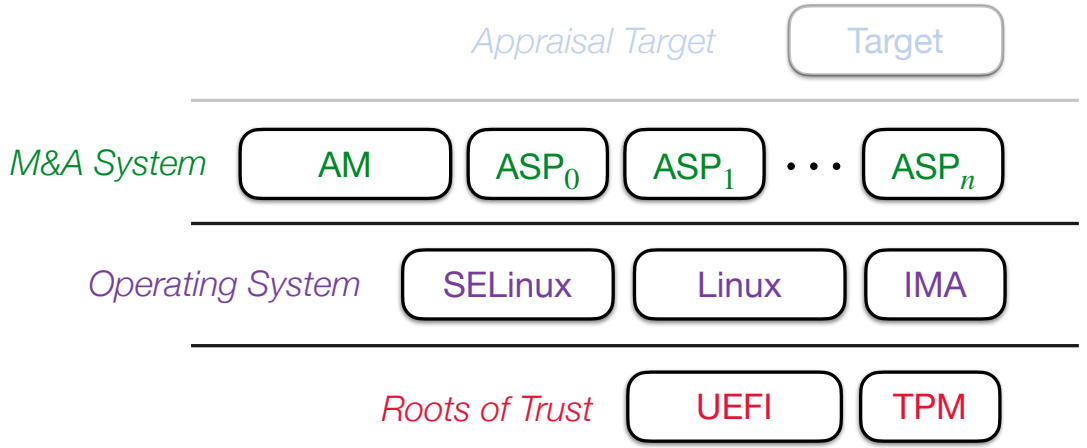
TPM State

- Good PCR 4
  - good bootloader
  - should measure `initramfs`
  - should use command line specification to start
- Good PCR 8 & 9
  - good command line starts `initramfs`
  - good `initramfs`
  - good boot materials
  - *good policy measurement script*
  - *good measurement script invocation*
- PCR 10 (ignored here)
  - memorializes IMA trace
  - not useful for sealing
- Good PCR 11
  - policy measurement ran
  - good initial SELinux and IMA policies
  - good `init` indicates start with good policies



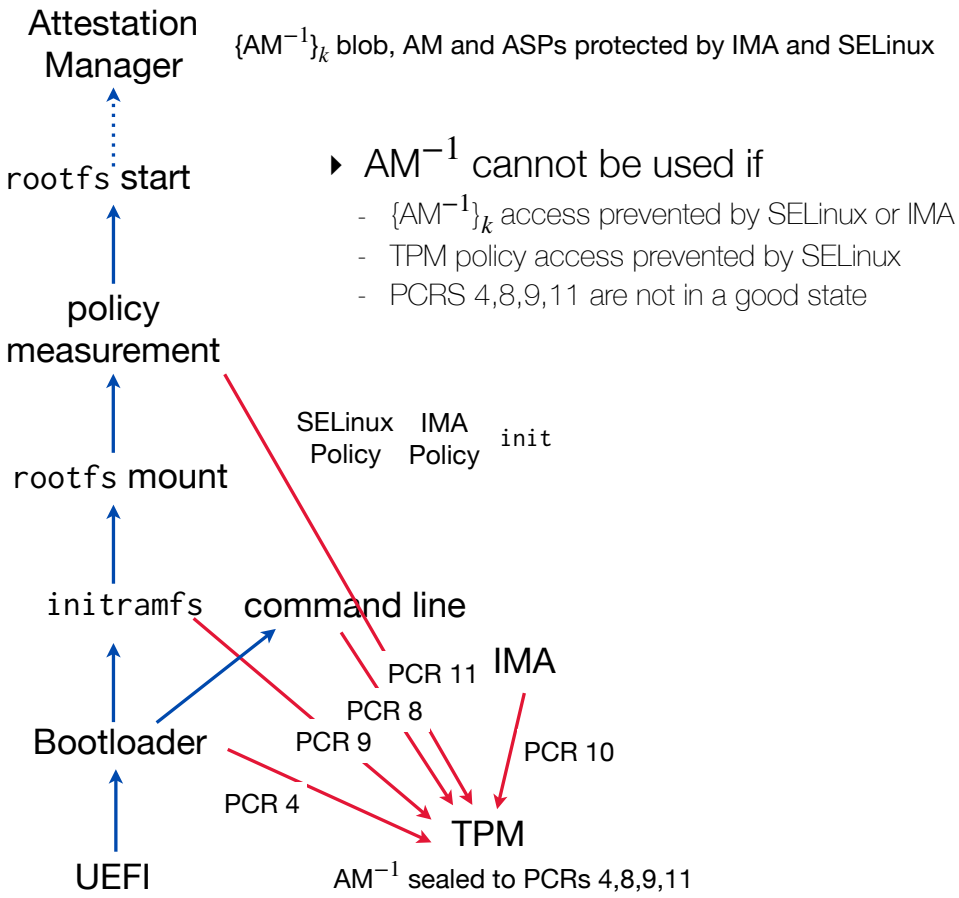
Runtime Attestation Layer

- Measure and start AM
- Establish ASP libraries
- Ensure AM^{-1} availability
- Begin Copland protocol execution



AM^{-1} Protection and Use

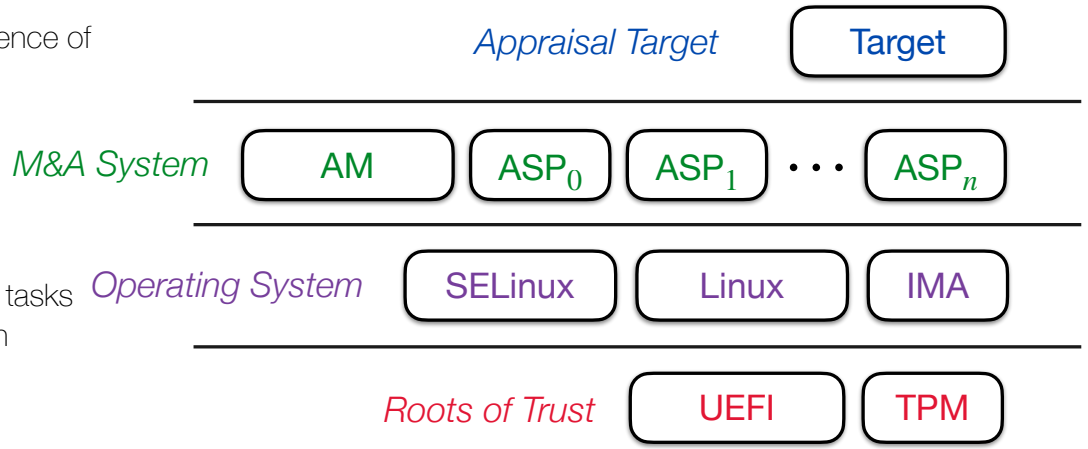
- Starting and Protecting AM
  - IMA policy prevents bad AM binary starting
  - IMA policy prevents bad ASPs from running
  - SELinux provides runtime access control
  - AM is formally verified to properly execute Copland protocols
- Generating and Protecting AM^{-1}
  - TPM generates AM^{-1} from $\{AM^{-1}\}_k$ blob on demand
  - SELinux enforces $\{AM^{-1}\}_k$ access control
  - IMA Extended Verification Mode (EVM) protects $\{AM^{-1}\}_k$ permissions
  - Authorized TPM policy must be loaded to enable key
  - Authorized TPM Policy seals AM^{-1} to PCRs 4,8,9,11
  - SELinux enforces access control over TPM Policy
- Using AM^{-1}
  - key is a strongly bound identifier for the AM
  - AM signature binds evidence to the associated AM
  - AM signature memorializes boot
  - effectively extends trust to user-space attestation mechanisms



- AM^{-1} cannot be used if
  - $\{AM^{-1}\}_k$ access prevented by SELinux or IMA
  - TPM policy access prevented by SELinux
  - PCRS 4,8,9,11 are not in a good state

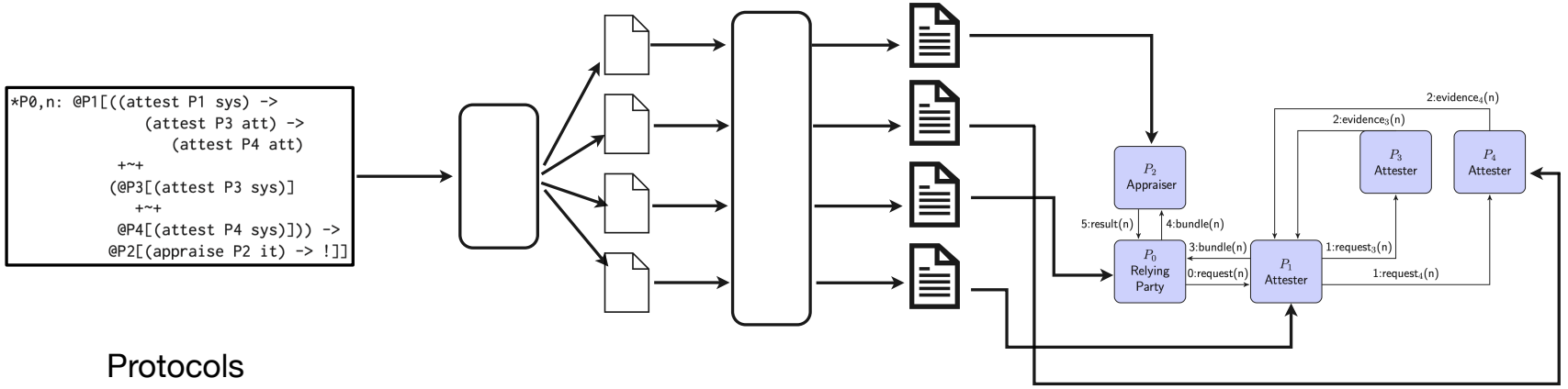
General Purpose Runtime Attestation

- Boot to AM is generic
  - any good signature over evidence $\forall e \,.\, \{e\}_{AM^{-1}}$ is evidence of trusted AM
  - AM is configurable and formally verified
  - small and memory safe
- M&A Subsystem
  - runs arbitrary Copland attestation protocols
  - attestation service providers (ASPs) perform attestation tasks
  - Copland attestation protocols sequence ASP execution
  - AM signing itself is an ASP
- Appraisal Targets
  - customize ASPs and protocol for specific applications
  - no requirement to customize target
- Evidence $\{ \mathsf{E} \}_{AM^{-1}}$
  - check signature to assure evidence integrity and good boot
  - check evidence to establish trust in target
  - formal semantics for protocol and evidence



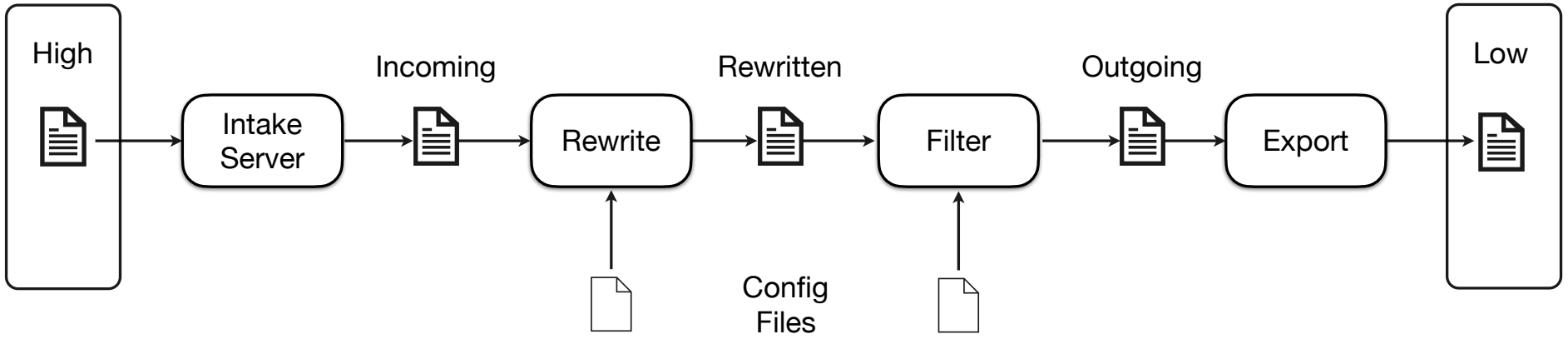
proper bundling
∧ satisfies appraisal policy
∧ valid signature
} ⇒ trustworthy target

MAESTRO Attestation Infrastructure



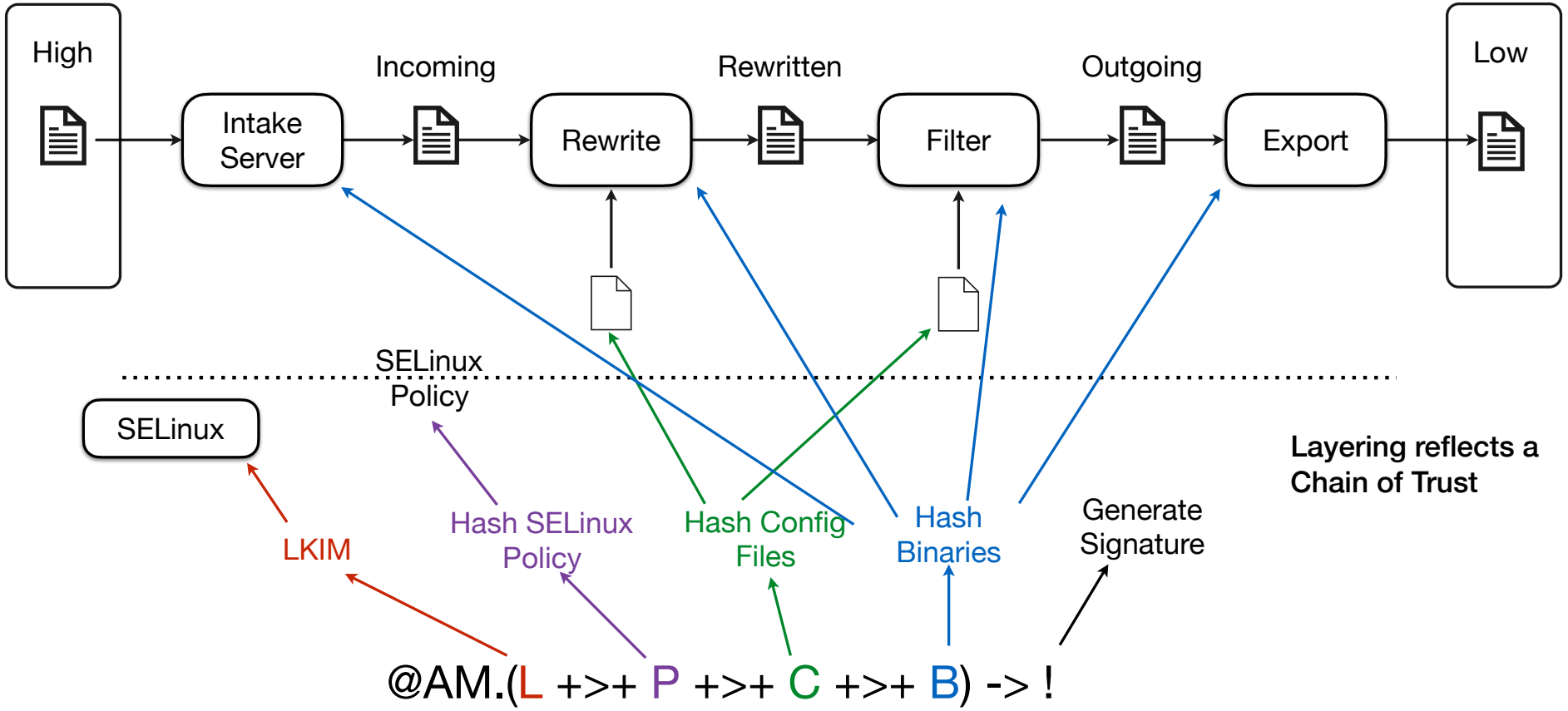
- Long running attestations
  - to our knowledge no one has studied long-running experiments on complex attestations
  - evaluating various flexible mechanisms
- Modeling attacks
  - sneaking by the attestation/appraisal system
  - directly attacking the attestation/appraisal system
- Attestation Test Bed
  - controlled evaluation environment
  - mixed architecture - ARM, Intel, IoT, Xen, KVM

Cross Domain System



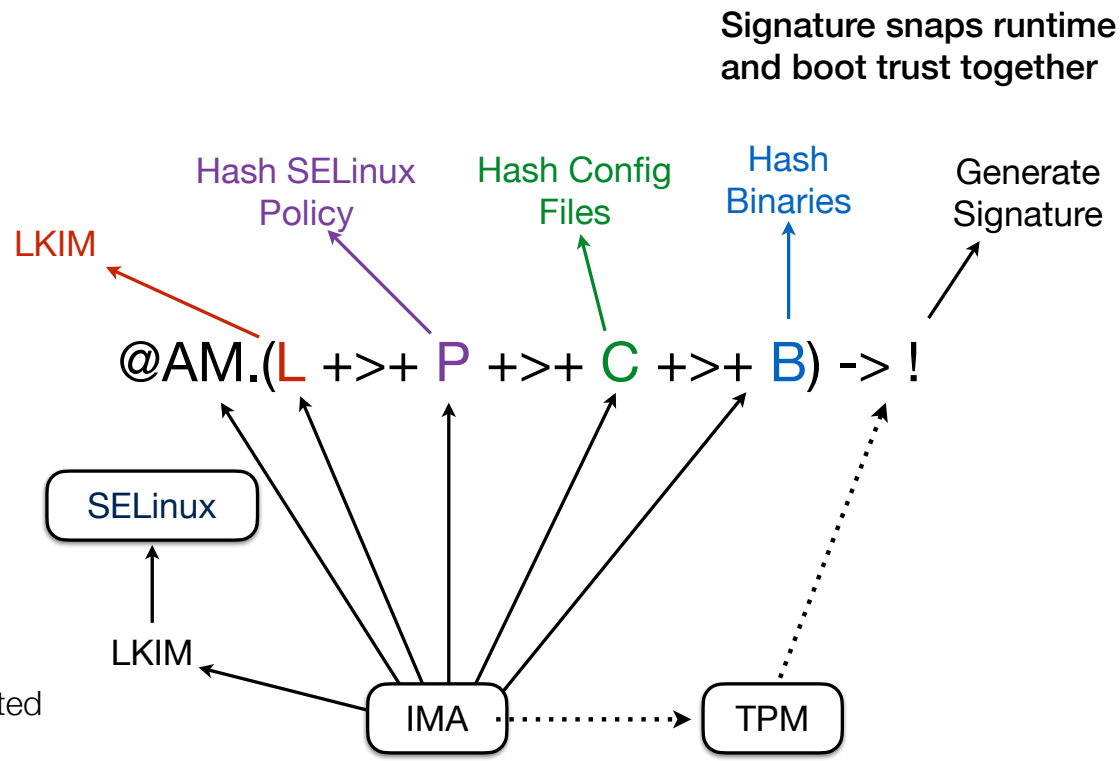
- Moving messages between security domains
  - intake receives a message from the high-side writes to incoming buffer
  - rewriter reads from the incoming buffer, applies rewrite rules, and writes to rewritten buffer
  - filter reads from the rewritten buffer, applies address filtering rules, and writes to outgoing buffer
  - export reads from outgoing buffer and outputs to low-side client
- Configuration
  - rewrite and filter processes have configuration files
  - SELinux policy enforces flow through the system
- Messages reaching the low-side client must be:
  - received from the high-side client
  - rewritten by a properly configured rewriter
  - filtered by a properly configured filter

ASPs and Protocol



Protecting Attestation at Runtime

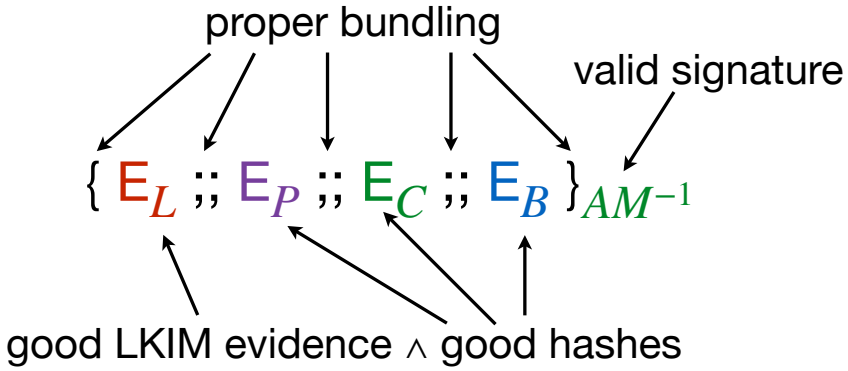
- Runtime IMA Measurements
  - Policy specifies hashes for ASPs
  - Policy specifies a hash for AM
  - IMA writes log to TPM PCR 10 (currently unused)
- AM^{-1} Signature
  - key is TPM resident
  - SELinux controls access to key blob
  - IMA EVM controls key blob permissions
- Linux
  - measured during boot using Invary LKIM
  - remeasured at runtime using Invary LKIM
  - SELinux policy dumped and hashed
  - good signature memorializes boot
  - the AM's key is not available if boot policy is violated



Appraising Attestation Results

- Trustworthy target if
  - proper bundling
  - evidence satisfies appraisal policy
  - valid signature
- Proper bundling
  - indicates measurement ordering
  - generated by verified AM
- Satisfies appraisal policy
  - E_L - LKIM policy appraisal
  - E_{P-B} - Hashes checked against golden values
  - AM^{-1} - Signature checked with public AM key
- Provisioning requirements
  - gather good hashes
  - generate and distribute AM key pair
  - define LKIM appraisal policy

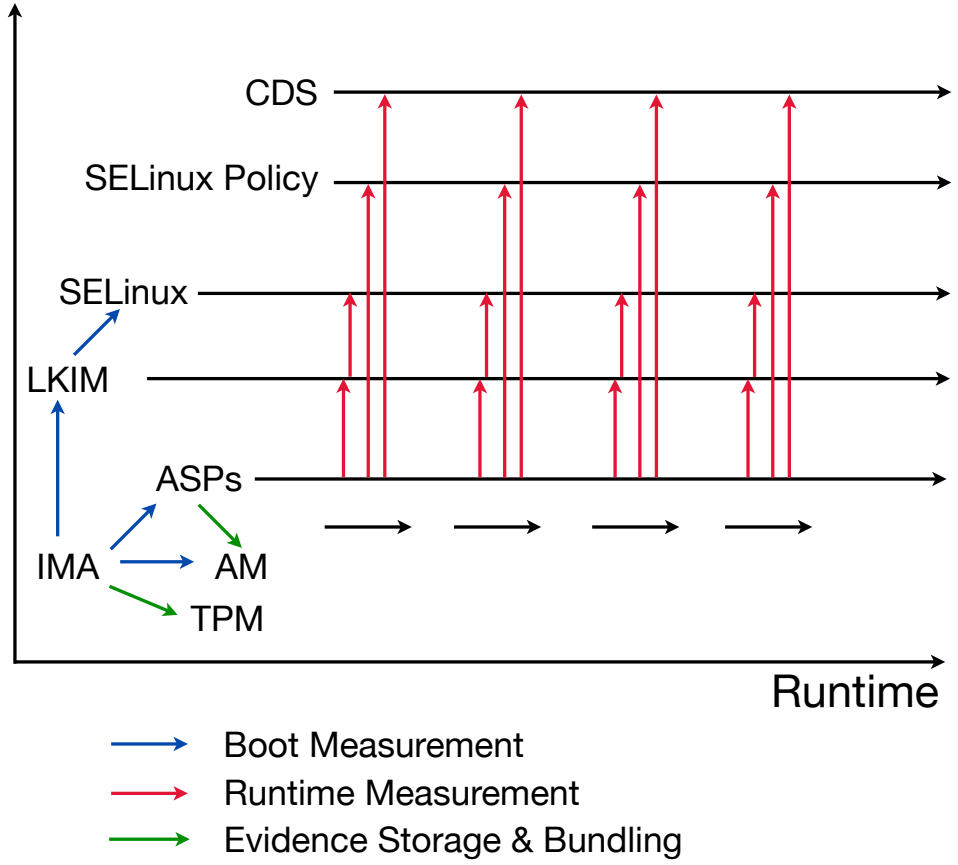
@AM.(L +>+ P +>+ C +>+ B) -> !



proper bundling
∧ satisfies appraisal policy
∧ valid signature
} ⇒ trustworthy target

Layered Runtime Attestation

- Boot to an initial measured state
  - establish running AM with bound key
  - IMA hashes and checks AM on invocation
  - AM^{-1} is available on good PCRs, good AM and encrypted blob
- Remeasure at runtime
  - AM executes Copland attestation protocols
  - ASPs gather information after IMA check by IMA
  - Protocol execution bundles evidence
  - AM signs gathered evidence with AM^{-1}
- Appraisal and Remeasurement
  - AM communicates with relying party
  - Appraisal may occur in AM, Relying Party, or third party appraiser
  - Remeasurement may occur in AM or Relying party
- PCRs and AM^{-1} are the trust link
  - boot measured into PCRs
  - signing key sealed by PCRs and protected by SELinux and IMA
  - signature carries trust meta-evidence
- Layering builds trust bottom up
  - dependencies measured first
  - bundled evidence reflects measurement order
  - verified in earlier work



Adversary Goals and Attack Mechanisms

The adversary's primary goal is convincing a relying party to trust something it should not

The adversary's secondary goal is convincing a relying party not to trust something it should

- Attacks on attestation target
  - change target without impacting policy compliance
  - change target and repair before measurement (TOCTAU)
- Attacks on evidence and meta-evidence
  - post measurement changes directly to evidence
  - generate signatures using incorrect components
  - cache alterations and poisoning
  - evidence package replay and spoofing
- Attacks on attestation infrastructure
  - compromise AM identity and steal AM's signing key
  - compromise AM execution and ASP ordering
  - alter ASPs to report incorrect, but compliant evidence
  - attack crypto and attestation protocol infrastructure
  - incorrectly report appraisal results
- Attacks on system infrastructure
  - compromises to hardware
  - changing boot images and boot order
  - TPM, IMA, and SELinux policy modifications

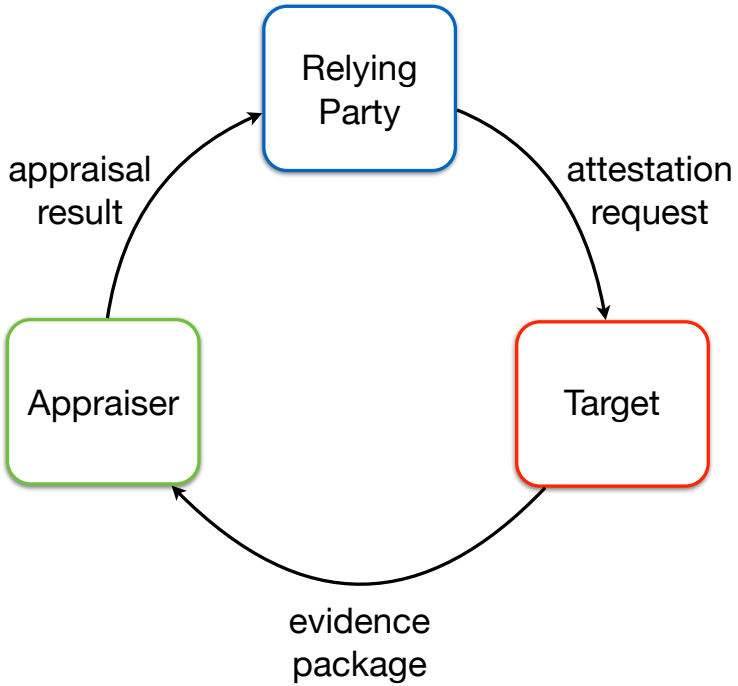
Testing Attestation and Appraisal

- Components targeted in testing
  - boot measurement infrastructure
  - runtime measurement infrastructure
  - CDS system configuration and components
- Attacks on configurations
  - altering component configuration
  - changing SELinux, IMA and TPM policy
- Attacks on executables
  - changing component runtime behavior
  - replacing or modifying executables
- Attacks across lifecycle
  - boot time attacks
  - runtime attacks
  - transitioning from boot trust to runtime trust

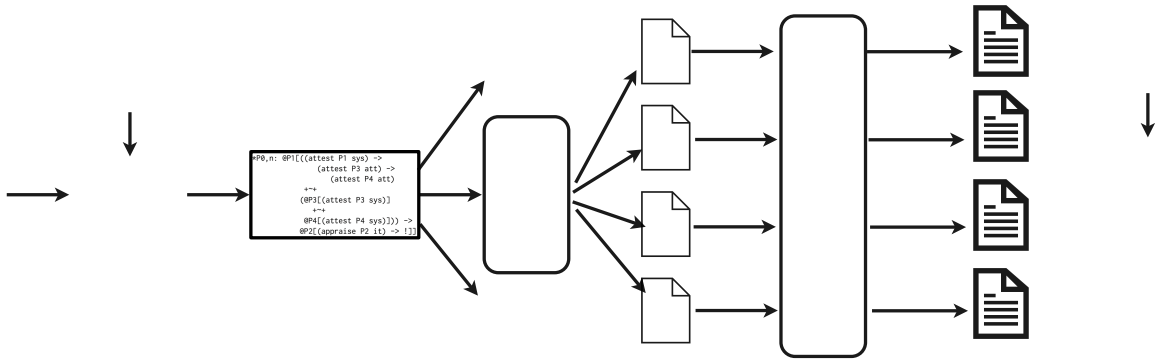
Attacks Considered		
Component	Configuration	Executable
Hardware	✘	✘
TPM	✔	✘
Bootloader	✔	✘
LKIM	✔	✘
Kernel	✔	✔
IMA	✔	✔
SELinux	✔	✔
AMs	✔	✔
ASPs	✔	✔
CDS Comp	✔	✔

What We Learned

- Boot transition to runtime is messy
  - boot trust must be reflected in runtime appraisal
  - yet there is no moment when runtime starts
  - integration with low level apparatus helps (IMA, SELinux, TPM)
- The AM's signing key is critical
  - a good AM key signature memorializes trusted boot
  - AM key compromise invalidates all attestation results
  - the AM key is long-lived and difficult to protect
- Design for attestation
  - short lived processes are more difficult to attack
  - processes run only when needed
  - dependencies first and layering is essential
  - separate infrastructure from application
- Mind the Gap (TOCTOU)
  - Time between measurement and use
  - IMA hash -> IMA startup
  - Binary runtime hash and use
- Assumptions and Interactions
  - boot to a bad state without SELinux
  - grab the AM^{-1} key blob that TPM will not unseal
  - reboot to a good state with good PCR values
  - use the AM^{-1} key blob from the bad boot



Next Up…



- Long-running attestation
  - re-measurement intervals
  - evidence caching and behavior
  - evidence behavior over time
- Larger layered targets
  - multi-machine attestations and appraisal
  - evidence bundling and abstraction
  - external appraisal services
- Evidence as program understanding
  - formal notions of measurement and abstraction
  - temporal evidence properties
  - composition evidence properties
- Protocols From Systems
  - move the user from protocol authoring to system modeling
  - generate protocols from system models
  - include adversary models
- Put Evidence Semantics to Work
  - linter to provide protocol writing guidance
  - type analysis to predict protocol behavior
  - understanding protocol orderings
- Separation issues in AM and ASPs
  - compartmentalization of ASP execution
  - separation within the AM
  - verus modeling for ASPs

Odds and Ends

- Perry Alexander - KU
  - palexand@ku.edu
- Adam Petz - KU
  - ampetz@ku.edu
- Will Thomas - KU
  - 30wthomas@ku.edu
- Logan Schmalz - KU
  - loganschmalz@ku.edu
- Sarah Johnson - KU
  - sarahjohnson@ku.edu
- Joshua Guttman - MITRE
  - guttman@mitre.org
- Paul Rowe - MITRE
  - prowe@mitre.org
- James Carter - NSA
- Stephen Smalley - NSA
- Daniel De Graaf - NSA

- KU/FBI CyberSecurity Conference, April 2025
- Research agreement between KU and Invary, LLC for commercialization
- Evaluation at KCNSC for potential deployment
- Meeting bi-weekly with MITRE and NSA Liaisons
- Presentations at CCS'25, HCSS'25, LSS'25
- Petz, A., W. Thomas, A. Fritz, T. Barclay, L. Schmalz, and Perry Alexander, "Verified Configuration and Deployment of Layered Attestation Managers," *Proceedings of the 22nd International Conference on Software Engineering and Formal Methods (SEFM'24)*, LNCS 15280, November 4-8, 2024, Aveiro, Portugal.
- Johnson, S. and P. Alexander, "Ordering Attestation Protocols," n preparation for *Network and Distributed System Security (NDSS'26) Symposium*, San Diego, California, February 23-27, 2026.
- Thomas, W., L. Schmalz, A. Petz, P. Alexander, J. Guttman, J. Carter, "Layered Attestation in Action: Attesting to a Cross-Domain Solution," in preparation for *Network and Distributed System Security (NDSS'26) Symposium*, San Diego, California, February 23-27, 2026.