Remote Attestation for Cloud-Based Systems

Dr. Perry Alexander[1] Dr. Andrew Gill[1] Dr. Prasad Kulkarni[1] Adam Petz[1] Paul Kline[1] Justin Dawson[1] Jason Gevargizian[1] Mark Grebe[1] Edward Komp[1] Edward Bishop[2]

[1]Information and Telecommunication Technology Center
Electrical Engineering and Computer Science
The University of Kansas

[2]Southern Cross Engineering

May 6, 2015

When should you trust my system?

- You know its identity
  - strong, unambiguous identification
  - asymmetric key cryptography
  - secret key strongly bound to the platform
- You know it is built from good parts
  - strong identification of system configuration
  - boot-time hashes stored in protected memory
  - trusted configuration delivery mechanism
- You know it is behaving as expected
  - direct or trusted indirect observation of good behavior
  - contextual evidence gathered during system operation
  - trusted evidence delivery, storage and evaluation mechanism

Trust is grounded in knowing *identity* and *behavior*.

Clouds and Trust

Cloud structure complicates knowing identity and behavior

- Applications no longer run "under the desk"
  - platform ownership is gone
  - difficult to directly observe behavior
  - no access to hardware
- Anonymous and changing operating environment
  - hardware is virtualized and invisible
  - migration moves applications and systems
  - identity and measurement cannot be rooted in hardware
- Unknown actors in the same operating environment
  - many virtual platforms on the same physical platform
  - many applications accessing the same resources
  - significant trust in the cloud to separate virtual platforms

Virtual Blinking Lights

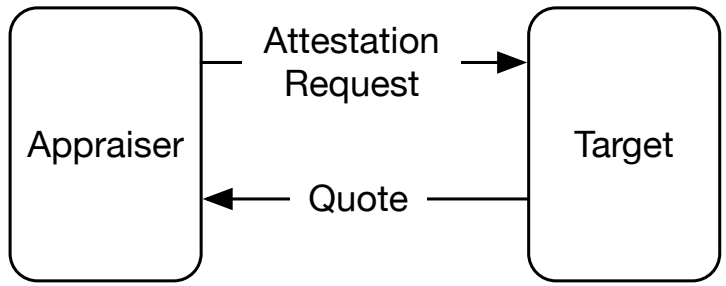
Provide new capabilities that establish and maintain trustworthy cloud-based application deployment

- Establish trust in cloud applications
  - trust in user-space applications
  - trust in cloud infrastructure
  - trust in application cohorts
- Provide a common trust infrastructure
  - standard application architecture
  - flexible communication mechanisms
  - application-specific measurement
  - formally verifiable trust protocols
  - roots-of-trust for storage and reporting
- Integration with existing standards and practices
  - Integration with RedHat Linux, Xen, and OpenStack
  - Uses Trusted Computing Group's TPM and vTPM guidelines
  - Developed in concert with NSA R2X and R2D

Semantic Remote Attestation

A basic four step process for establishing trust:

- Appraiser requests a quote
  - specifies needed information
  - provides a nonce
- Target gathers evidence
  - measures application
  - gathers evidence of trust
- Target generates quote
  - measurements and evidence
  - original nonce
  - cryptographic signature
- Appraiser assesses quote
  - good application behavior
  - infrastructure trustworthiness

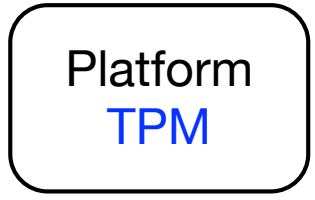


Trusted Platform Module

- Provides and Protects Roots of Trust
  - Storage Root Key (SRK) - root of trust for storage
  - Endorsement Key (EK) - root of trust for reporting
- Quote generation
  - high integrity quotes - ($\{\|RS\|\}_{AIK^-}$, SML, $\{\|n, PCRComp\|\}_{AIK^-}$)
  - high integrity evidence - ($\langle E, n\rangle$, $\{\||\langle E, n\rangle|, PCR\|\}_{AIK^-}$
- Sealing data to state
  - $\{D, PCR\}_{K^+}$ will not decrypt unless PCR = current PCR
  - data is safe even in the presence of malicious machine
- Binding data to TPMs and machines
  - ($\{K^-\}_{SRK^+}$,K) - $\{D\}_{K^+}$ cannot be decrypted unless SRK^- is installed
  - ($\{J^-\}_{K^+}$,J) - $\{D\}_{J^+}$ cannot be decrypted unless K^- and SRK^- are installed

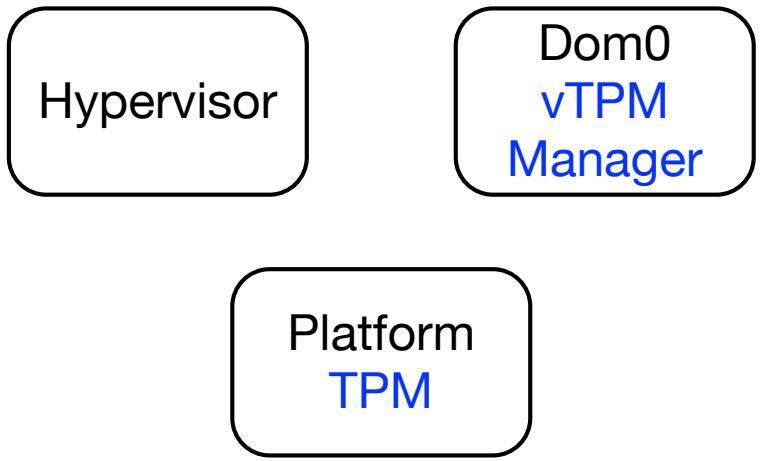
The Cloud Challenge

Chasing the bottom turtle



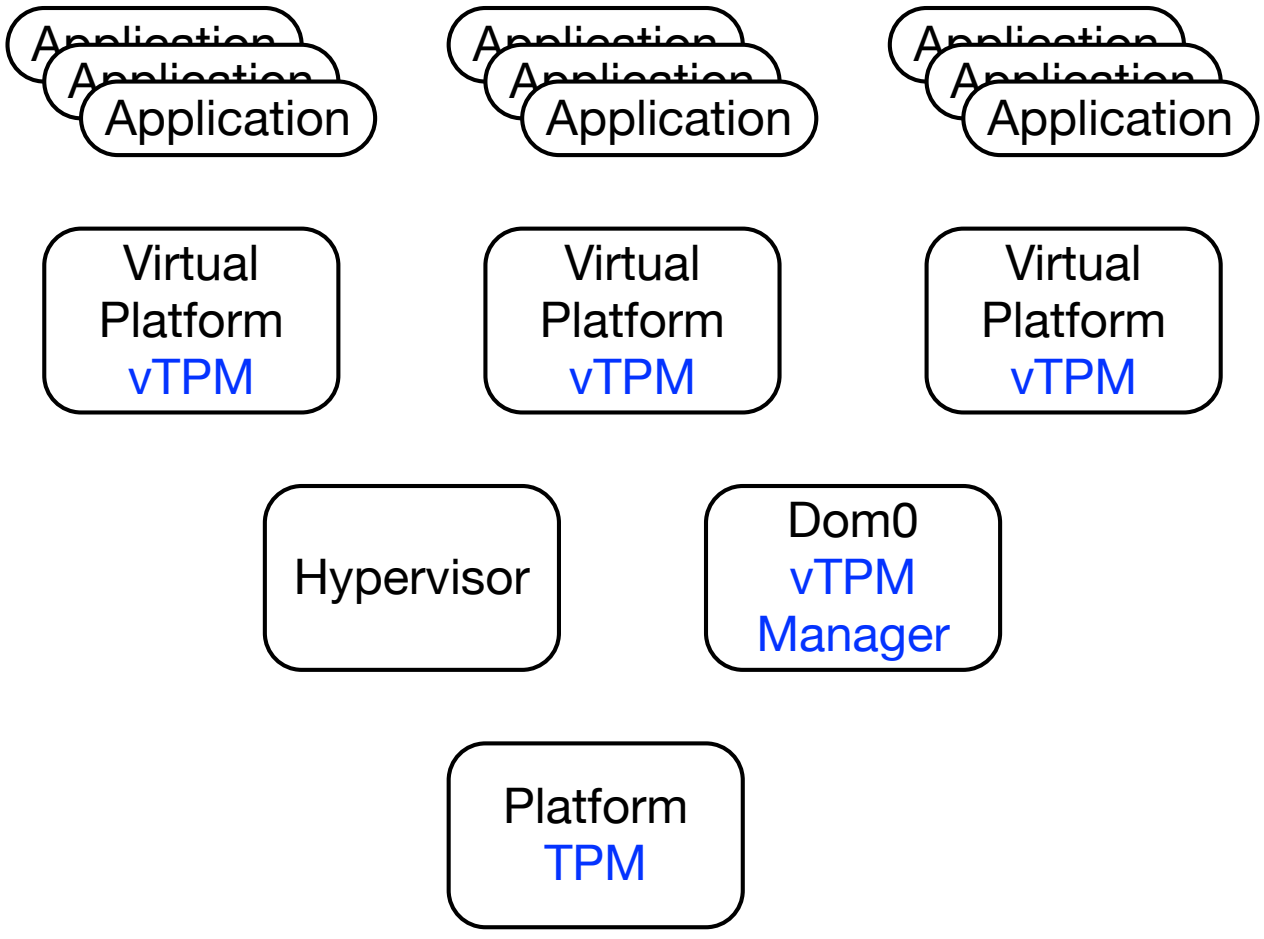
The Cloud Challenge

Chasing the bottom turtle



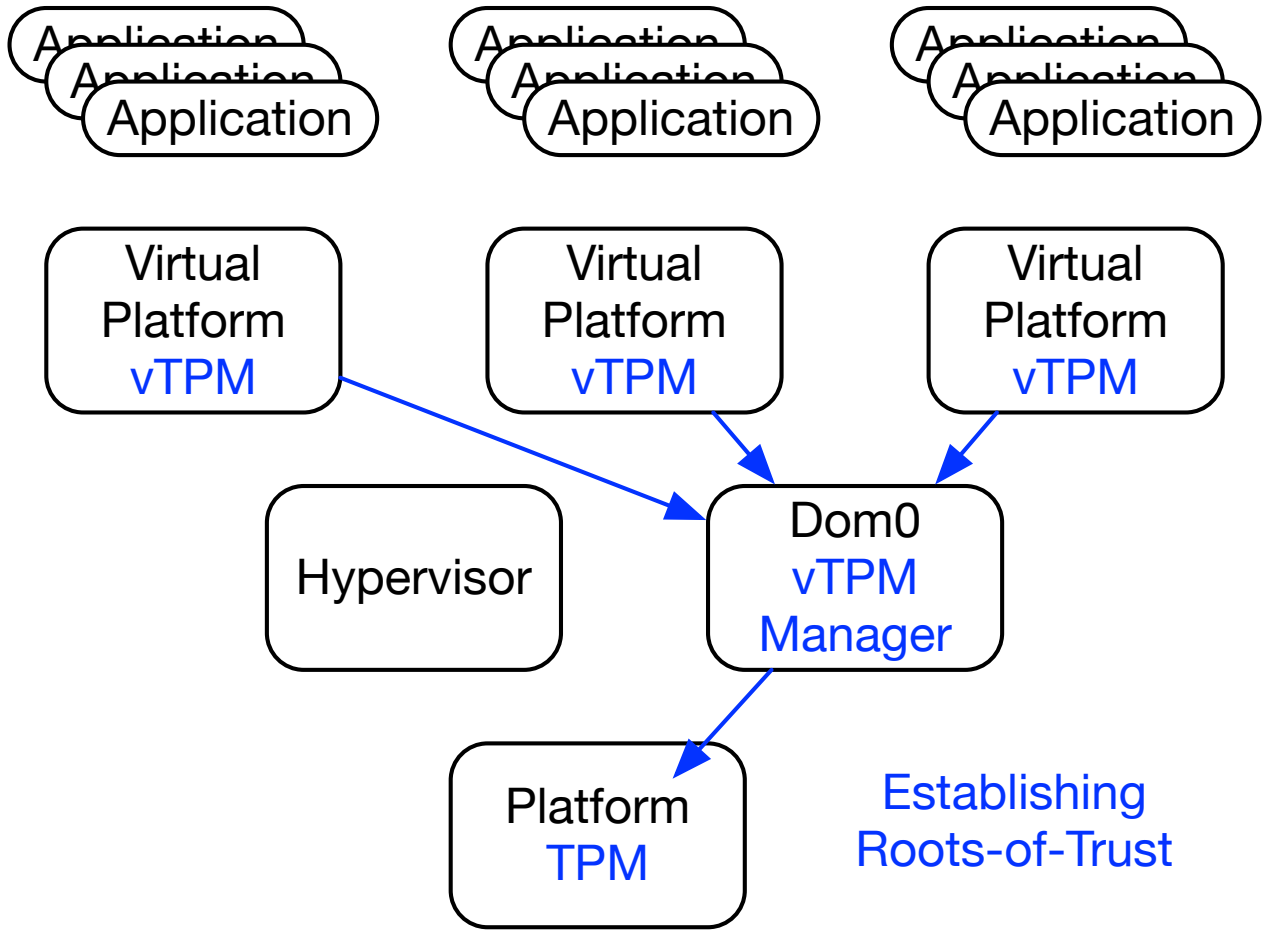
The Cloud Challenge

Chasing the bottom turtle



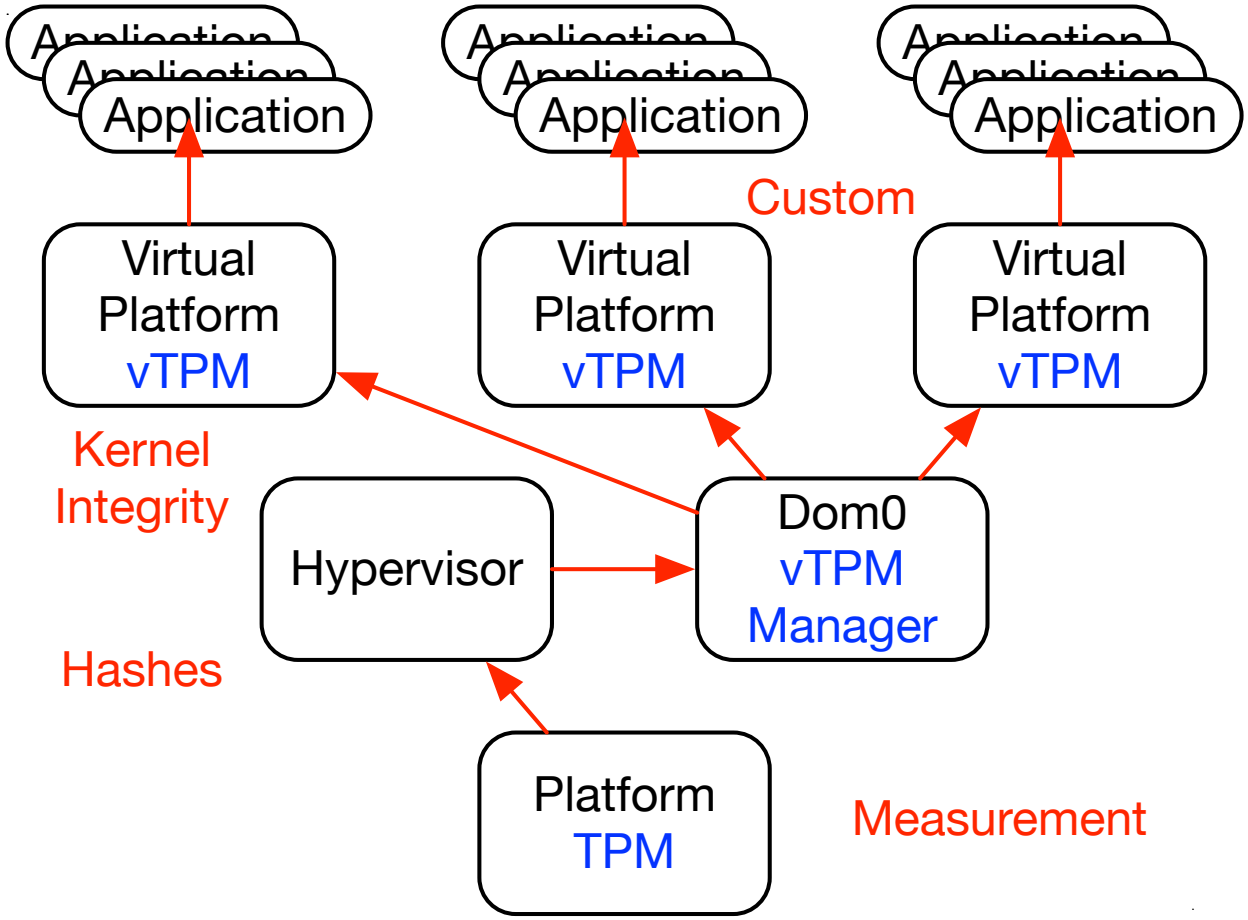
The Cloud Challenge

Chasing the bottom turtle



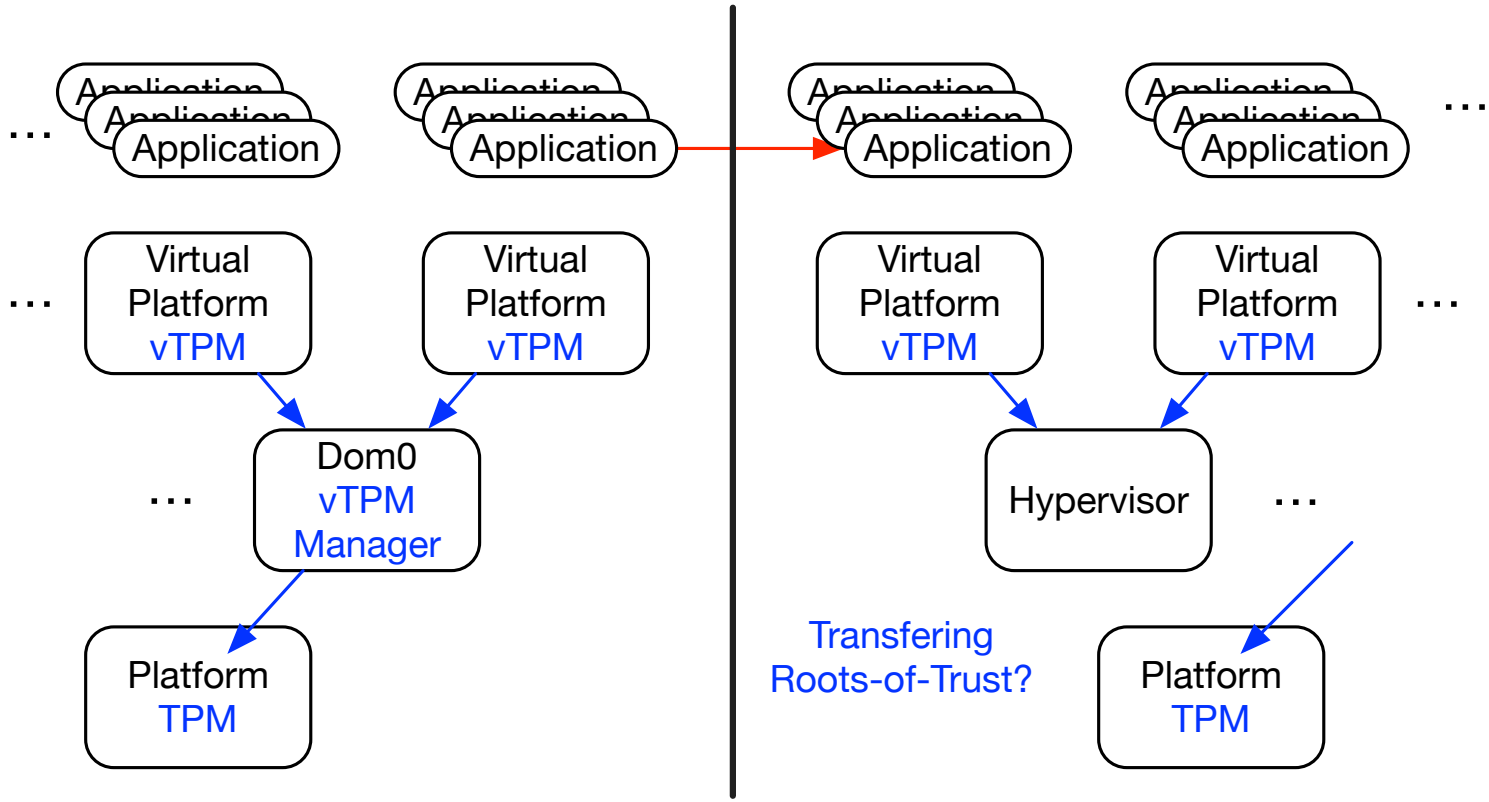
The Cloud Challenge

Chasing the bottom turtle



The Cloud Challenge

Chasing the bottom turtle



New Enabling Technologies

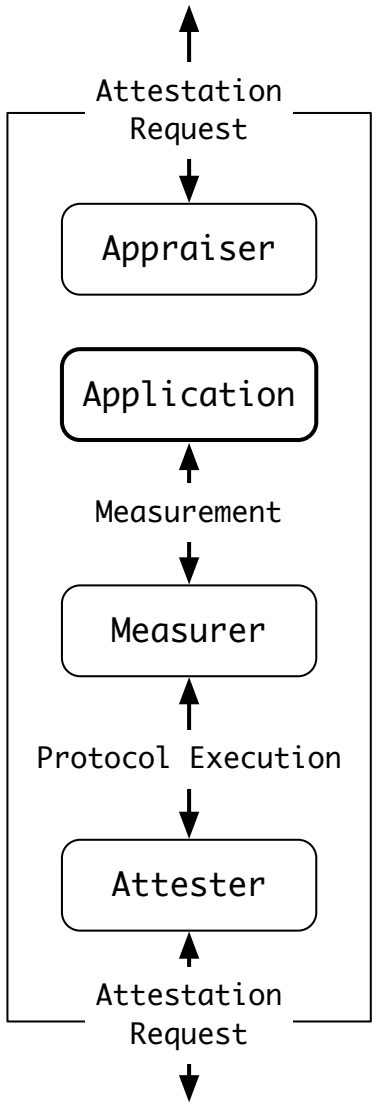
ArmoredSoftware will provide new technologies for system trust

- Trustworthy protocol execution
  - executable and analyzable protocol representation
  - privacy policy compliant attestation protocol negotiation
  - verifiable remote attestation protocol execution
- Application specific measurement
  - scriptable general purpose measurement engine
  - compile-time assistance for measurer synthesis
  - specialized measurement bundled with applications
- Lightweight trust infrastructure
  - strong identity establishment and maintenance
  - abstract communications capability
  - migration-sensitive vTPM infrastructure

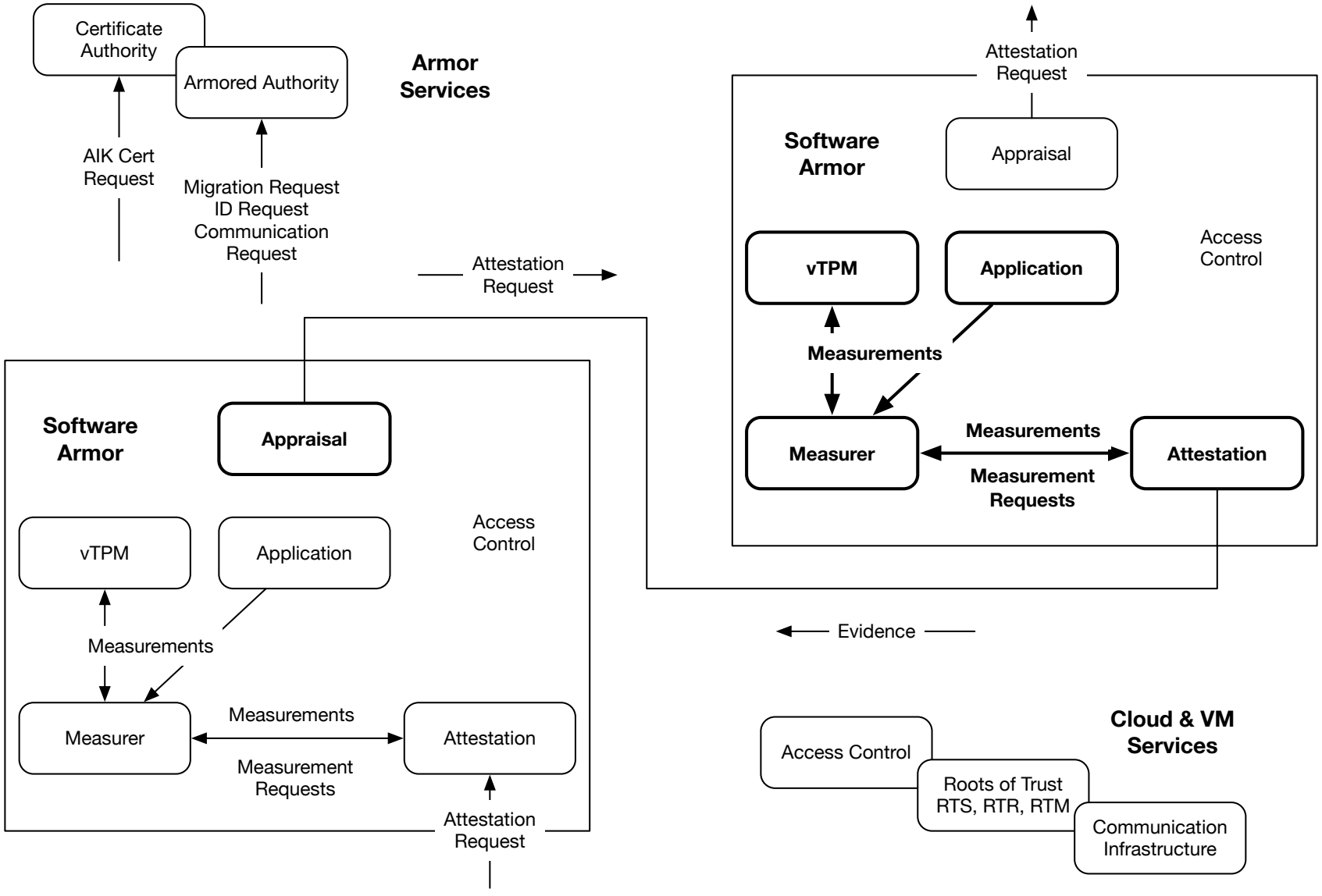
Armored Application Architecture

M&A targeted to an application

- Appraiser makes attestation requests
- Attester responds to attestation requests
- Measurer gathers evidence from application
- Influenced by the *Trusted Research Platform* and *Principles of Remote Attestation*



System-Level Architecture

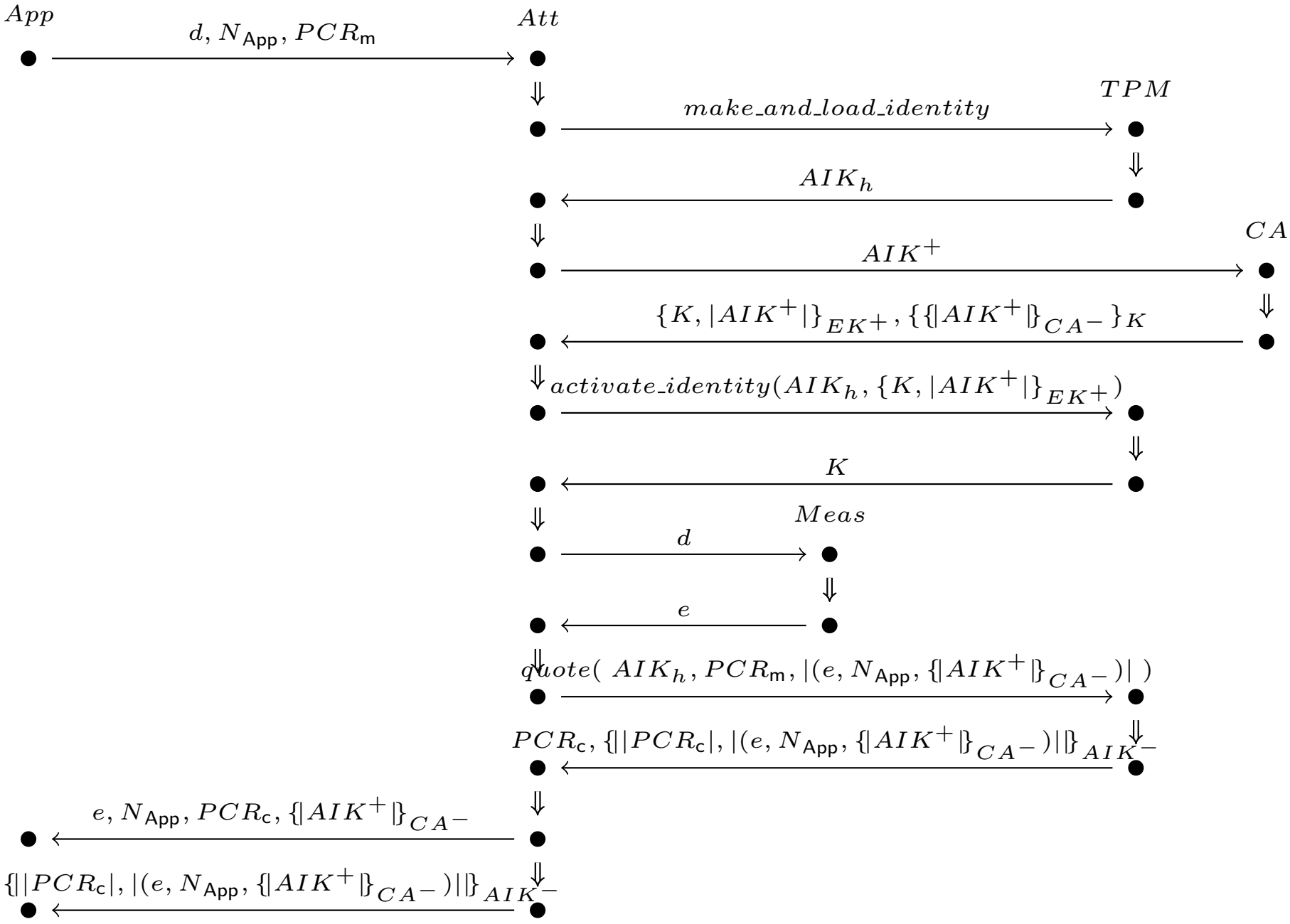


Trustworthy Protocol Execution

Negotiation and execution

- Representation and execution
  - protocols as first-class language structures
  - generates evidence of trusted execution
  - respects privacy policies
  - formal semantics
- Attestation Protocol Negotiation
  - appraiser and attester agree on an attestation protocol
  - appraiser needs information for assessment
  - attester protects target assets
- Attestation Protocol Execution
  - invokes measurement routines to gather evidence of behavior
  - packages evidence to ensure evidence integrity and confidentiality
  - generates meta-evidence to ensure process integrity

Privacy CA Attestation



EDSL for Trusted Protocols

First-class protocol structures

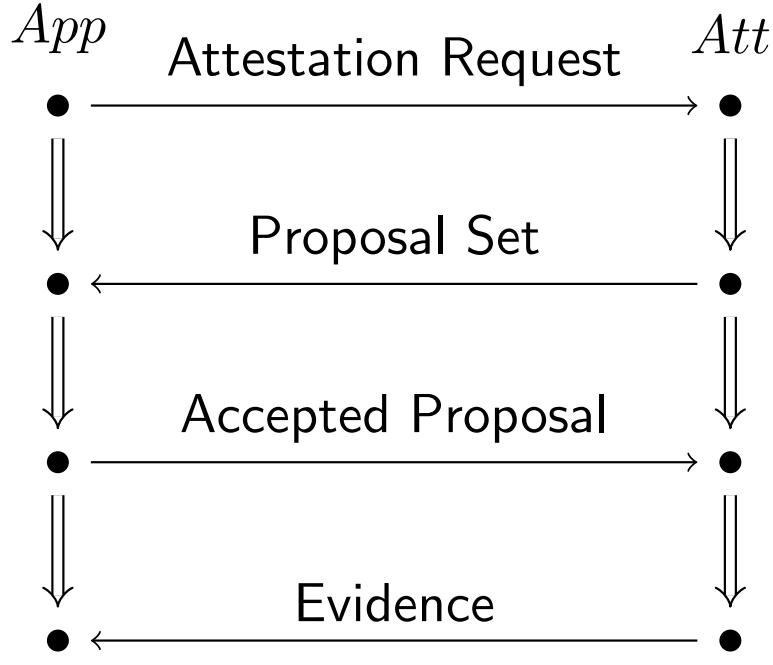
- First-class structure for protocols
  - encapsulates a protocol-centered computation
  - semantics provide a basis for static analysis
  - based loosely on the `Reader` monad
- Abstract communication primitives
  - extended RPC-style capability
  - requests remote execution
  - defines `send` and `receive` operations
  - abstracts away communication details

```
do {
     f(x);
     y <- f(x);
     z <- send a y;
     y <- receive a
}
```

Negotiating a Protocol

Respecting privacy

- Typical negotiation
  - request sent to Attester
  - Attester generates proposal
  - Appraiser selects protocol
  - Attester executes protocol
- Three kinds of requests
  - execute protocol 22
  - provide {`OS_config`, `http_stat`, `firewall_stat`}
  - execute protocol `do { ... }`
- Three negotiation criteria
  - ability to satisfy the request
  - satisfaction of appraiser and attester privacy policies
  - previously obtained evidence



Negotiation Protocol

Request and Select

- Requests an attestation
- Receives proposals
- Selects from proposals

```
do { send t r;
     q <- receive t;
     e <- case {p:q | (policy? p)} of
            ∅ : None
            p : send t (choose p)
          end;
     case e of
       Some v : (appraise v)
       None : None
     end }
```

Negotiation is a protocol that can itself be selected or negotiated

Negotiation Results

- Evidence and Protocol pairs
- Satisfies privacy policy of attester
- Provide some or all of requested information

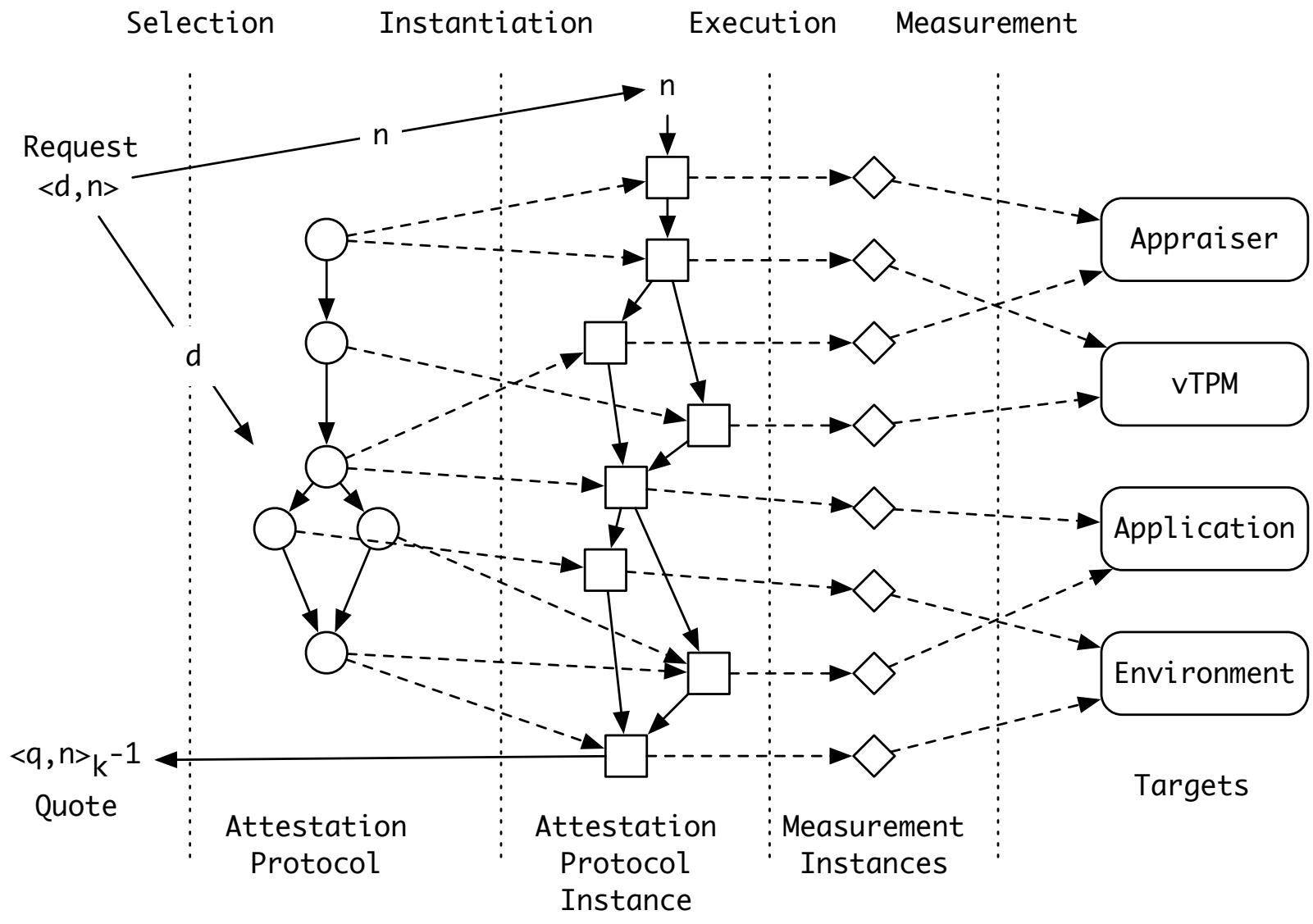
```
((ID,SIGHASH,SIGSRC),
 do { id <- getVCID;
      sig <- getSigFileEvidence;
      src <- getSigFileSrc;
      e <- createEvidence(id,sig,src);
      returnEvidence(e) })
```

Reified Protocol

Generated negotiation protocol code (currently by hand):

```
P = CreateChannel (AChannel "attesterChan") Target
    $ Send ANRequest (AChannel "attesterChan")
    $ Receive (Var "counterOffer") (AChannel "attesterChan")
    $ CalculateFinalRequest (Var "finalReq")
                            ANRequest
                            (Var "counterOffer")
    $ Send (Var "finalReq") (AChannel "attesterChan")
    $ Receive (Var "finalConfirmation")
              (AChannel "attesterChan")
    $ Case (Var "finalConfirmation") [(Var "finalReq")]
           (HandleFinalChoice (Var "result") (Var "finalReq")
           (Result (Var "result")))
           (Stuck "finalConf and finalReq match error")
```

Performing Measurement and Attestation



Single Realm Attestation

Protocol for gathering virus checker evidence

```
do { id <- getVCID;
     sig <- getSigFileEvidence;
     src <- getSigFileSrc;
     e <- createEvidence(id,sig,src);
     returnEvidence(e) }
```

and generates evidence of the form:

$$\langle (id, sig, src), \{\!|\,|(id, sig, src)|, PCRComp_0\,|\!\}_{AIK_0^-} \rangle$$

Appraisal replays the protocol up to crypto operations with known good measurements

Multi-Realm Attestation

Nested attestation requests evidence from the signature server directly:

```
do { id <- getVCID;
     sig <- getSigFileEvidence;
     src <- getSigFileSrc;
     srcEvidence <- send src r;
     e <- createEvidence(id,sig,src,srcEvidence)
     returnEvidence(e)
}
```

and generates bundled evidence:

$$\text{let} \quad b \quad = \langle (e), \{\!|\,|e|, PCRComp_1\,|\!\}_{AIK_1^-} \rangle \text{ in}$$
$$\langle (id, sig, src, b), \{\!|\,|(id, sig, src, b)|, PCRComp_0\,|\!\}_{AIK_0^-} \rangle$$

Trusting Evidence

Why bundling is hard

- Trusting evidence
  - hashes and TPM quotes
  - measure and appraise the attestation infrastructure
  - gather evidence of good protocol execution
- Trusting bundled evidence
  - appraisers do not know the source of evidence *a priori*
  - no global name space for evidence sources
  - bundled appraisals vs bundled evidence
- Trusting the appraiser
  - negotiated protocols must satisfy privacy policies
  - trust may not be transitive for applications and infrastructure
  - global policy is not an answer

Current Status

Demos available

- Attestation and Appraisal development
  - CA-Based attestation protocol execution example
  - simple dynamic appraisal of attestation results
  - integrated negotiation protocol and attestation protocols
- Measurement development
  - HotSpot-based Java VM run time measurements
  - detect and report several runtime anomalies
  - standard mechanism for extending measurement capabilities
- Infrastructure development
  - vchan, TCP/IP and socket communication infrastructure
  - initial certificate authority implementation
  - language-based interface with TPM 1.2
  - integrated Berlios TPM emulator
  - JSON-based data exchange formats

Ongoing Work

Goals for 2015

- Establish roots-of-trust and trust argument
  - measured launch and remeasurement of ArmoredSoftware
  - establish trust in the Xen/OpenStack infrastructure
- Executable protocol representation and protocol semantics
  - evidence of proper execution
  - static trust analysis
  - protocol-centered appraisal
- More capable measurement
  - compiler directed measurement
  - continuous measurement—tripping and trending
- Publicly available libraries and infrastructure

References