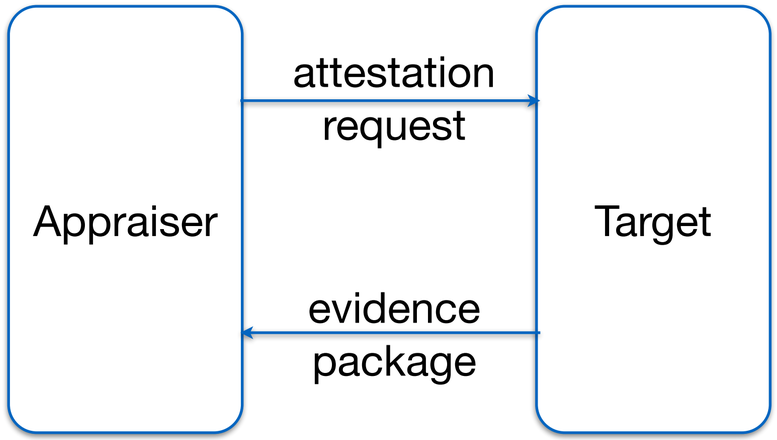
Flexible Mechanisms for Remote Attestation

Perry Alexander, Adam Petz, Anna Fritz, Grant Jurgensen
Information and Telecommunication Technology Center
The University of Kansas
{palexand,ampetz,arfritzz,gajurgensen}@ku.edu

Semantic Remote Attestation

- Appraiser requests evidence
  - specifies needed information
  - provides a fresh nonce
- Target gathers evidence
  - measures application
  - gathers evidence of trust
- Target generates evidence package
  - measurement results
  - the appraiser's nonce
  - cryptographic signatures
- Appraiser assesses evidence
  - good application behavior
  - infrastructure trustworthiness
  - good nonce

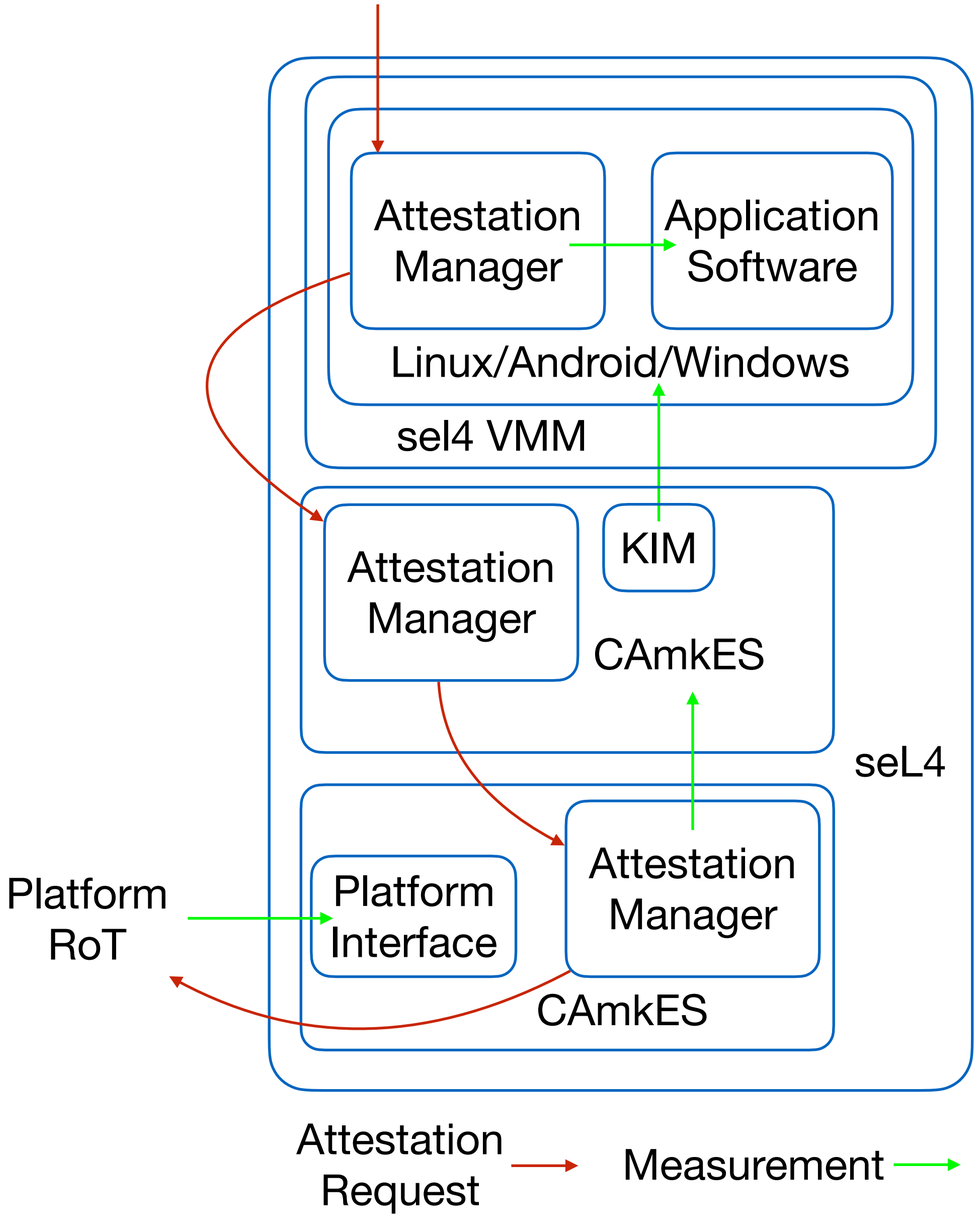


Research Goals

- Formal semantics of trust - Definition of trust sufficient for evaluating systems
- Verified remote attestation infrastructure - Verified components for assembling trusted systems
- Enterprise attestation and appraisal - Scaling trust to large, complex systems in principled ways
- Sufficiency and soundness of measurement - Formally defining what measurements reveal about a system

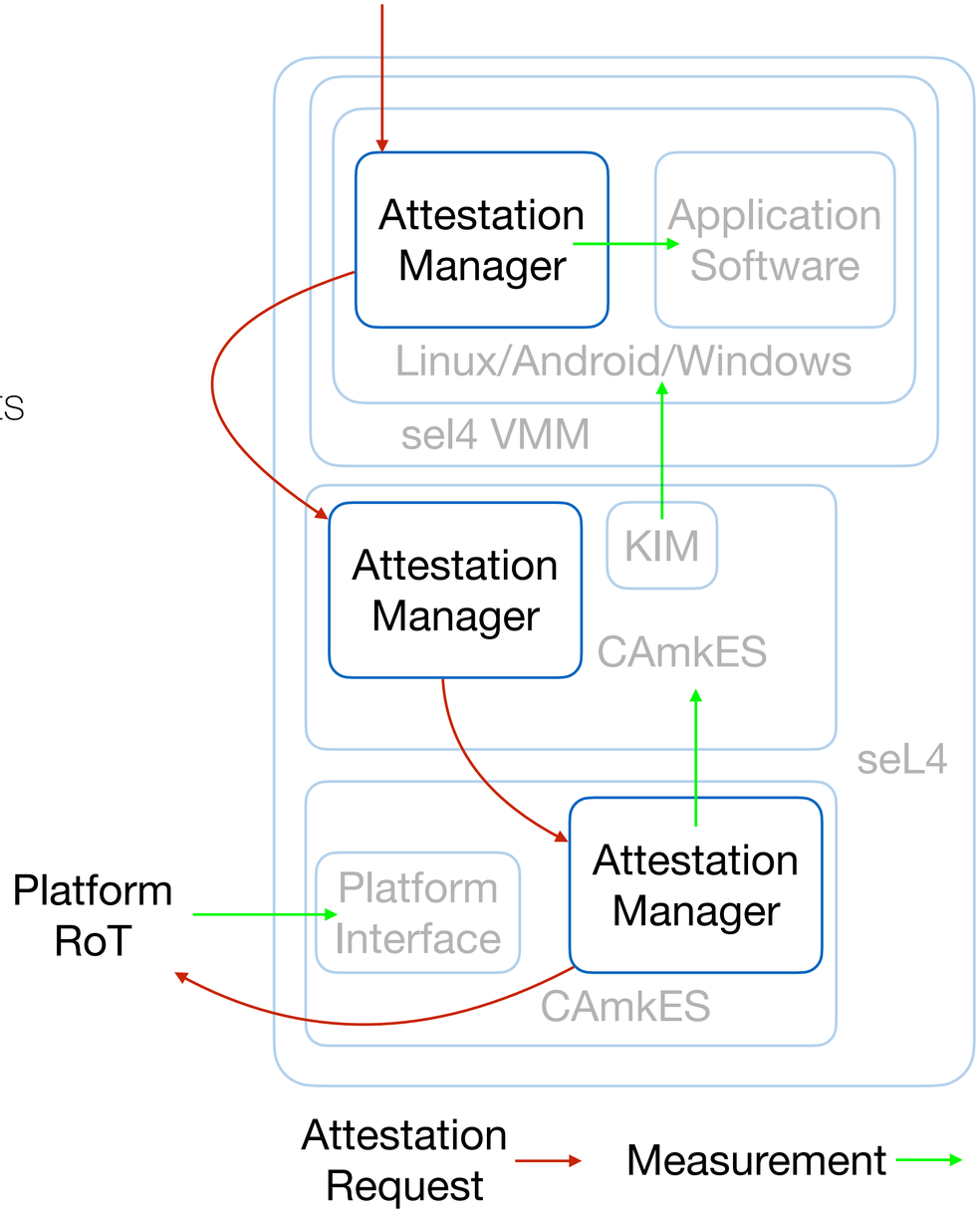
Remote Attestation Example

- Three attestation managers
  - UserAM - application software attestation
  - PlatformAM - kernel integrity measurement
  - seL4AM - hardware platform attestation
- seL4 implementation infrastructure
  - Linux VM running application software
  - CAmkES components running attestation infrastructure
  - platform roots-of-trust for late launch (pending)
- Attestation gathers evidence
  - attestation requests made top down
  - critical components measured bottom-up
  - evidence composed bottom up from roots-of-trust
- Is this a one-off attestation architecture?



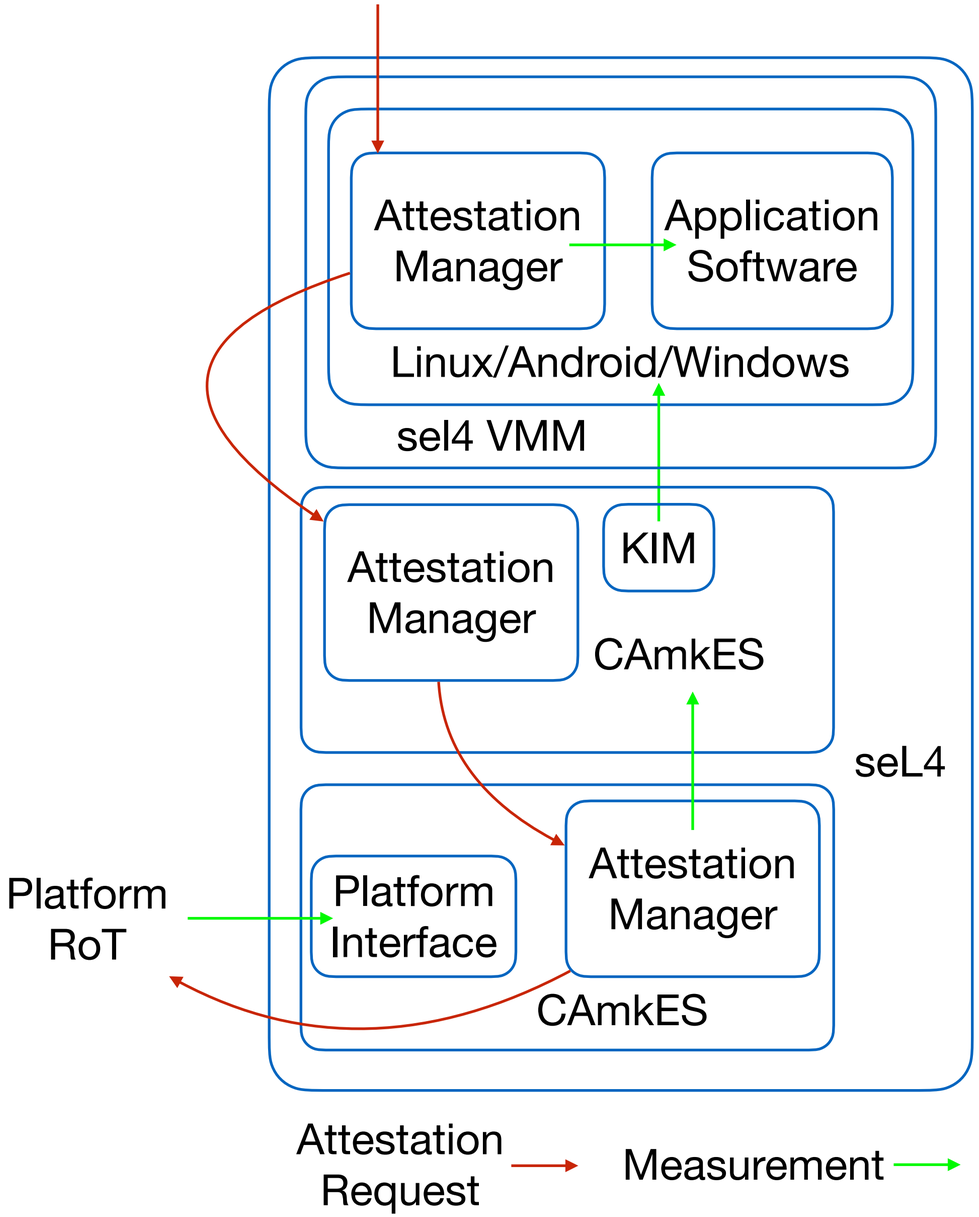
Layered Attestation

- UserAM receives attestation request
  - sends layered request to PlatformAM
  - receives PlatformAM evidence
  - performs application measurements
  - bundles PlatformAM evidence, nonce, and local measurements
- PlatformAM receives attestation request
  - sends layered request to seL4AM
  - receives seL4AM evidence
  - performs kernel integrity measurements
  - bundles seL4AN evidence, nonce, and KIM measurements
- seL4AM receives attestation request
  - retrieves boot evidence
  - bundles and returns boot evidence
- Reusable attestation architecture
  - builds evidence and trust bottom up from roots-of-trust
  - principled, reusable attestation template
  - captured by attestation protocol and system architecture



Flexible Mechanisms

- Attestation architecture building blocks
  - Common Attestation Manager
  - Attestation Service Providers
  - Copland attestation protocol language
- Patterns for attestation
  - common attestation structures like Layered Attestation
  - evidence bundling mechanisms
- Tools and Semantics for assessment
  - when is a protocol “good”?
  - when is one protocol better than another?
  - what does a protocol accomplish?



Attestation Manager

- Negotiation
  - establish a security context
  - find a mutually approved attestation protocol
- Copland Interpeter
  - executes a Copland protocol
  - verified compiler and Copland VM
- Communication
  - establish communication among AMs
  - API for executing **@P** commands
- Nonce Management
  - generating new, unique nonces
  - remembering nonces for appraisal
- Appraisal
  - general purpose appraisal function
  - “re-runs” attestation with golden values

- Negotiation
- Copland Interpreters
- Communications
- Nonce Management
- Appraisal

```
do{n<-nonce();
   e1<-@P:n[…];
   m<-nonce();
   e2<-@Q:m[…];
   a1<-(app n e1);
   a2<-(app m e2);
   return a1,a2
}
```

Verified Attestation Component

- Attestation Monad
  - state monad with exceptions
  - provides state for Copland interpreter
- Copland Interpeter
  - invoked on nonces and protocols
  - verified in previous work (NFM'21)
- Communication
  - assumes platform provides secure channel
  - abstract interface to specific communication implementations
- Nonce Management
  - assumes a platform source of randomness
  - tracks nonce generation for verification
- Appraisal
  - verified all gathered evidence is appraised
  - verified correctness up to specific measurement values

- Negotiation
- Copland Interpreters
- Communications
- Nonce Management
- Appraisal

```
do{n<-nonce();
   e1<-@P:n[…];
   m<-nonce();
   e2<-@Q:m[…];
   a1<-(app n e1);
   a2<-(app m e2);
   return a1,a2
}
```

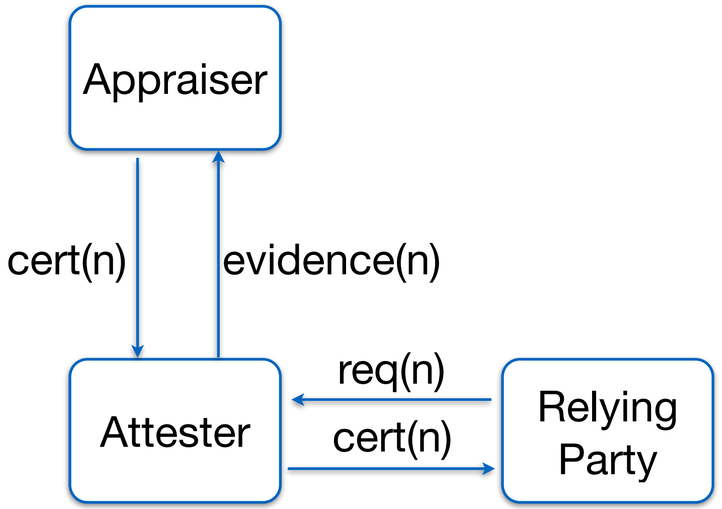
Attestation Patterns

- Attestation Protocol templates for common shapes
  - Layered
  - Certificate-Style
  - Cached
  - Background Check
- Implemented using communicating Attestation Manager instances
  - attestation service providers for measurement and other services
  - requires “plumbing” for communication, scheduling, and access control
- Principled composition
  - assembling attestation ecosystems
  - scaling to the enterprise
  - assessing impacts on adversaries

Certificate-Style

- Appraisal as a service
  - attester generates evidence
  - appraiser evaluates evidence
  - a certificate indicates appraisal results to relying party
- Relying party requests an appraisal
  - sends a request and a fresh nonce to attester
  - signs request for authenticity
- Attester gathers evidence and meta-evidence
  - executes measurers to gather system information
  - signs evidence with nonce to ensure integrity
- Appraiser evaluates evidence
  - checks evidence values and signature
  - generates a certificate with Relying Party's nonce
- Certificate returned to Relying Party
  - check the nonce, signature and appraisal result
  - include result in trust decisions

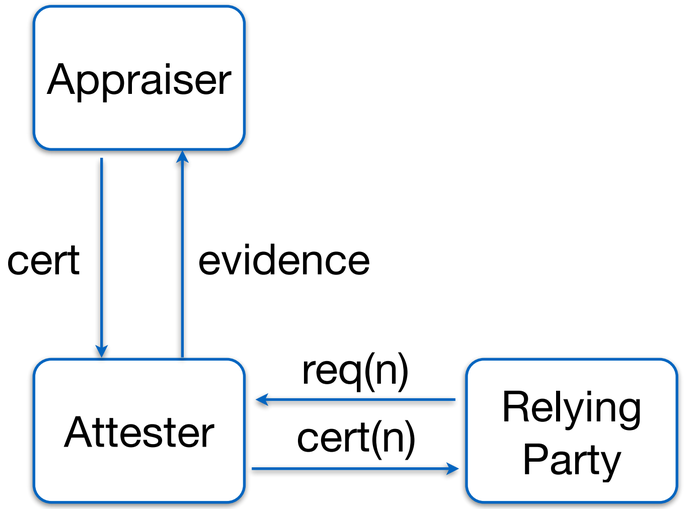
```
*P0,n: @P1[(attest P1 sys) ->
          @P2[(appraise P2 sys) ->
             (certificate P2 sys) ]]
```



Cached Certificate-Style

- Appraisal as a service (again)
  - attester generates and appraiser evaluates evidence
  - certificate is cached for future use
- Attester gathers evidence and meta-evidence
  - executes measurers to gather system information
  - signs evidence with nonce to ensure integrity
- Appraiser evaluates evidence
  - checks evidence values and signature
  - generates a certificate
- Attester caches certificate for future use
  - controls when and how attestation is performed
  - reuses attestation results for efficiency
- Relying party requests an appraisal
  - sends a request and a fresh nonce to attester
  - signs request for authenticity
- Certificate returned to Relying Party
  - check the nonce, signature and appraisal result
  - include result in trust decisions

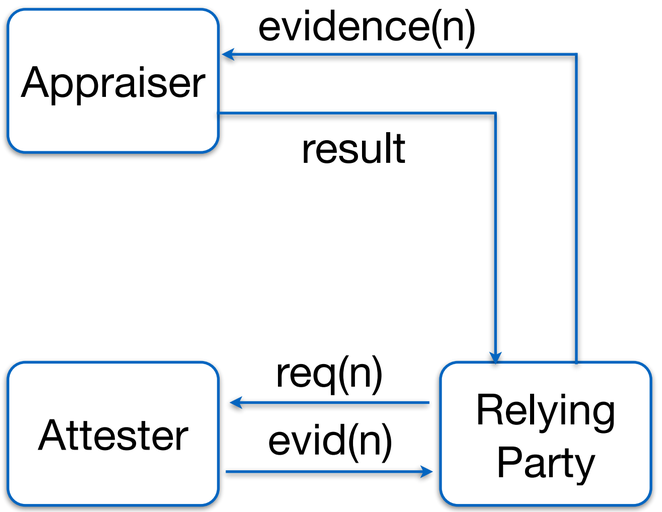
```
*P1:(attest P1 sys) ->
    @P2[(appraise P2 sys) -> (certificate P2 sys)] ->
       (store P1 cache)
*P0,n:@P1[((retrieve P1 cache) -<+ _) -> ![]
```



Background Check

- Appraisal as a service (again)
  - attester generates evidence
  - relying party requests appraisal
- Relying party requests an appraisal
  - sends a request and a fresh nonce to attester
  - signs request for authenticity
- Attester gathers evidence and meta-evidence
  - executes measurers to gather system information
  - signs evidence with nonce to ensure integrity
  - returns evidence to relying party with nonce
- Appraiser evaluates evidence
  - checks evidence values and signature
  - may generate a certificate if required
- Result returned to Relying Party
  - owns generated evidence
  - often Relying Party is also Appraiser

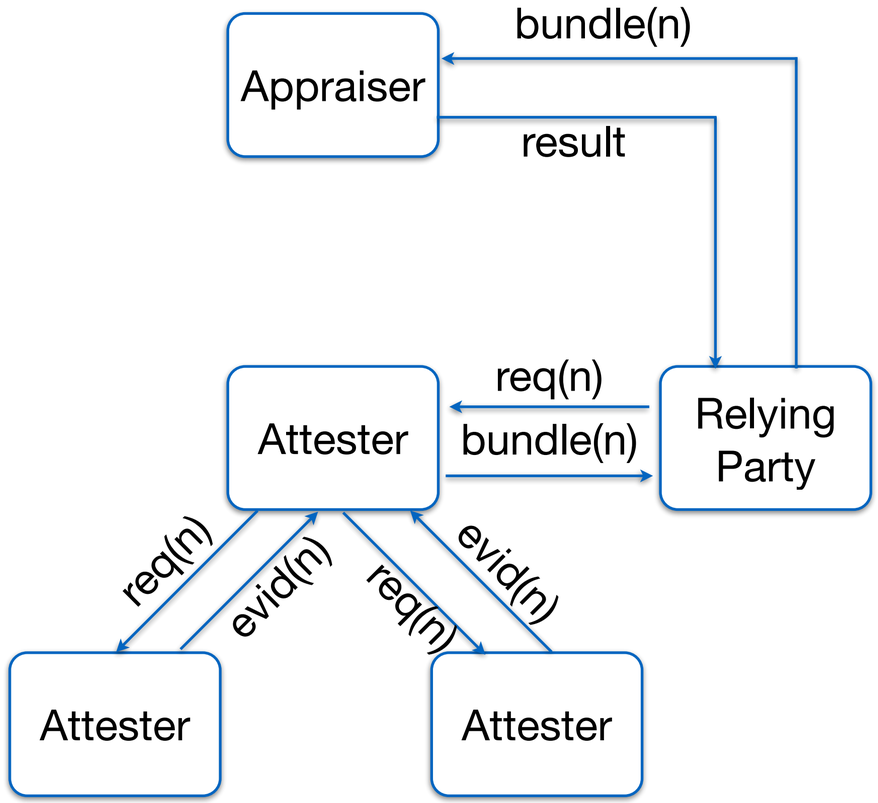
```
*p0,n: @P1[(attest P1 sys)] -> @P2[(appraise P2 sys)]
```



Layered Background Check

- Composing Layered and Background Check
  - background check style appraisal
  - layered style builds evidence bottom up
- Relying party requests an appraisal
  - sends a request and a fresh nonce to attester
  - signs request for authenticity
- Attester makes requests of separate attesters
  - sends a request and nonce to multiple attesters
  - manages ordering of attestation requests
  - layered attesters gather evidence
- Attester assembles evidence package
  - indicates evidence ordering
  - composes multiple attestation results
  - returns evidence to relying party
- Appraiser evaluates evidence
  - checks evidence values and signature
  - may generate a certificate if required
  - result returned to Relying Party

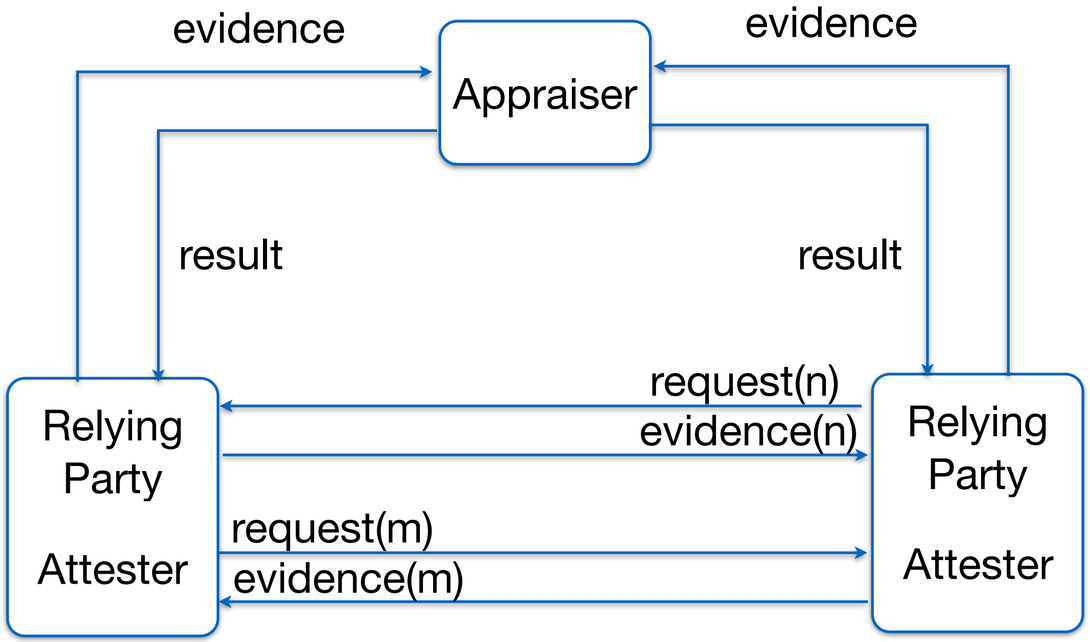
```
*p0,n: @P1[((attest P1 sys) ->
             (attest P3 att) ->
               (attest P4 att)
            +~+
           (@P3[(attest P3 sys)]
              +~+
            @P4[(attest P4 sys)])) ->
          @P2[(appraise P2 it) -> !]]
```



Parallel Mutual Attestation

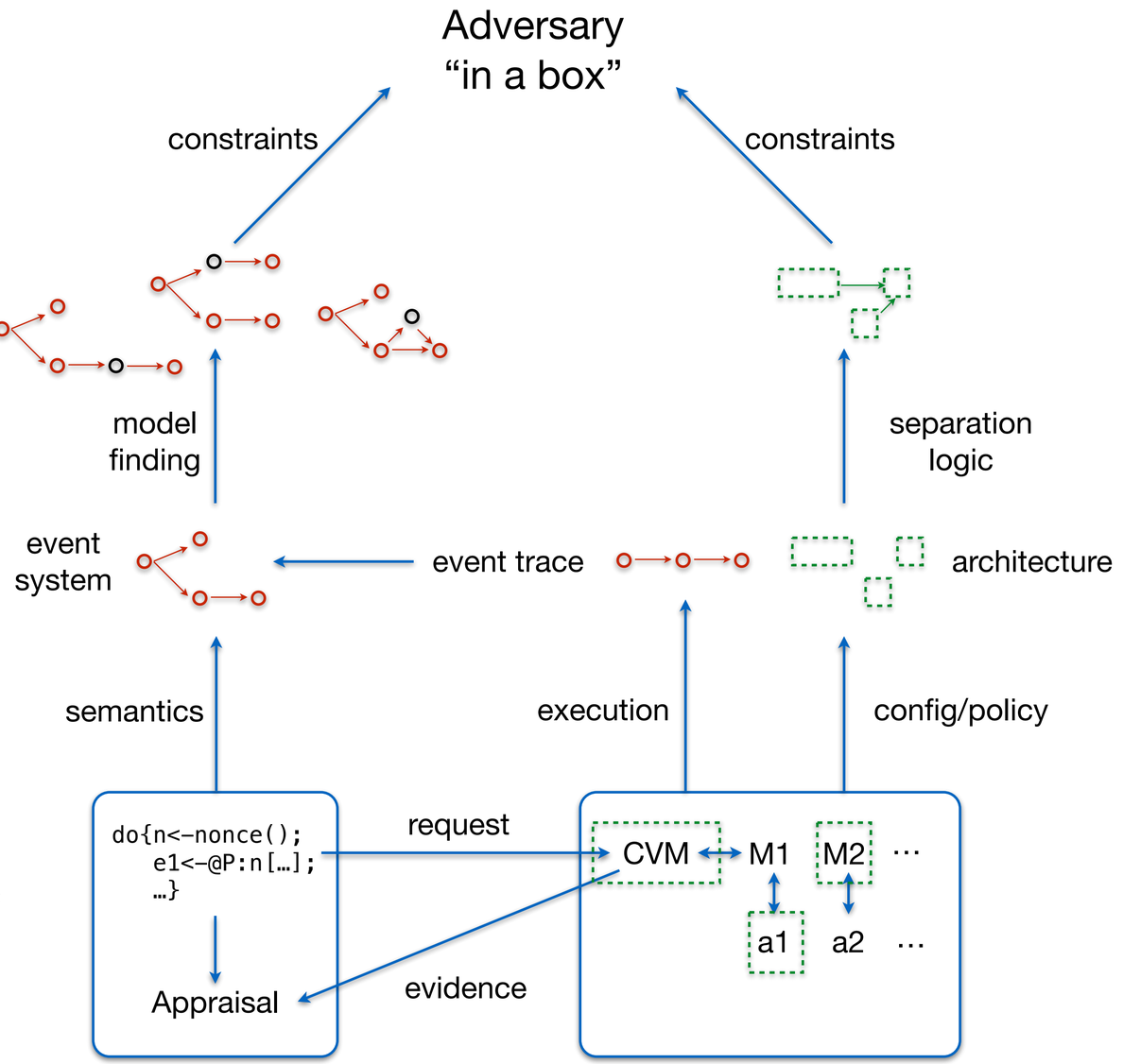
- Multi-Party Attestation
  - simultaneous attestation
  - single trusted appraiser
  - relying party = attester
- Both Relying Parties request attestation
  - send requests and nonces asynchronously
  - receive requests and nonces
- Both Attesters return evidence
  - attestation occurs asynchronously
  - no initial trust
- Both Relying Parties request Appraisal
  - shared, mutually trusted appraiser
  - returns appraisal result
- Same song, second verse
  - two background check attestations combined
  - could add caching or certificate generation

```
*P0,n 0 : @P1[(attest 01 P1 sys)] ->
              @P2[(appraise 01 P2 sys)]
*P1,m 1 : @p0[(attest 10 p0 sys)] ->
              @P2[(appraise 10 P2 sys)]
```



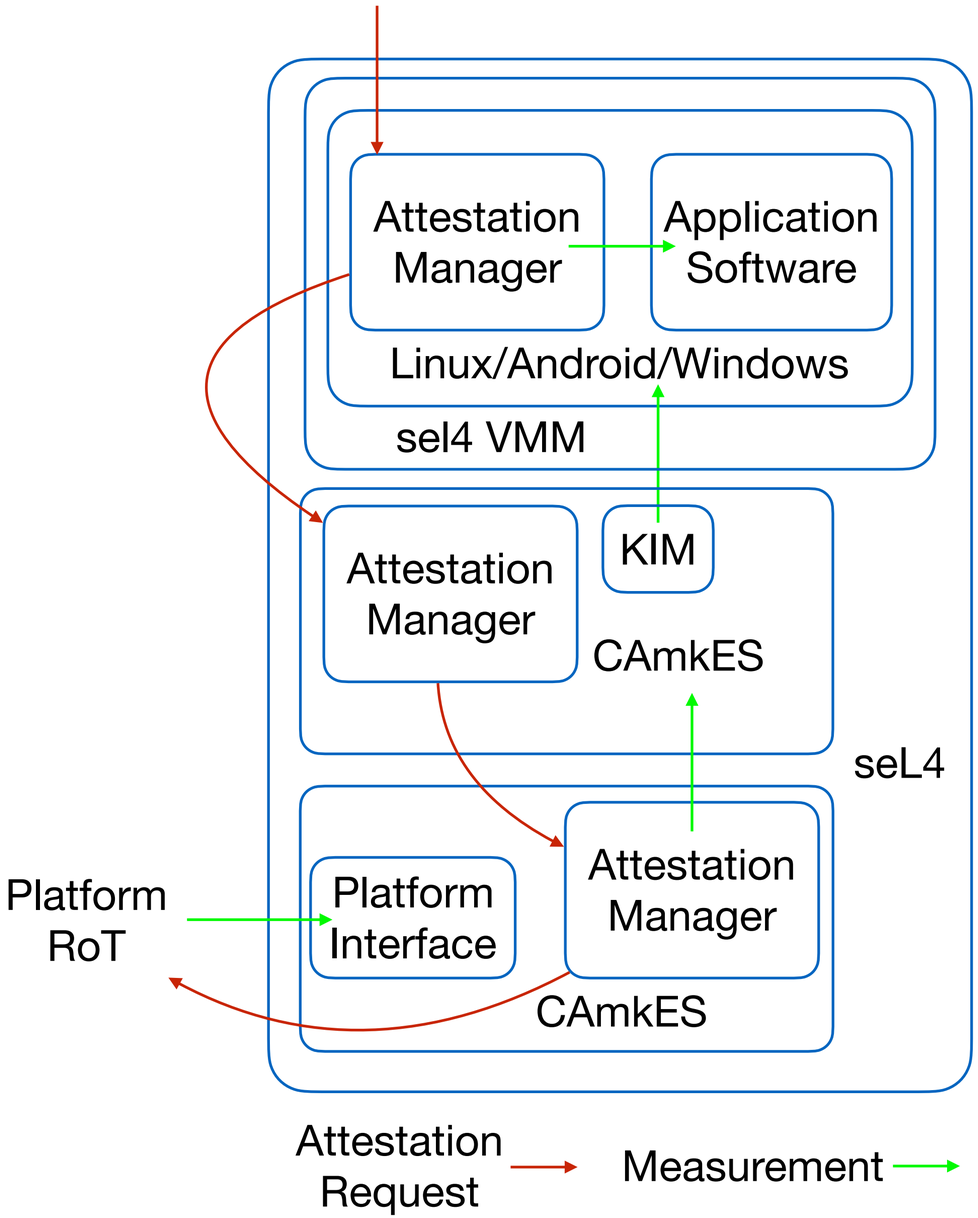
Protocol Analysis

- Assume a correct attestation platform
  - correctly executes Copland protocols
  - correctly appraises results
  - verified with respect to Copland semantics
- What can we say about protocols?
  - adversaries acting among protocol actions
  - adversaries accessing protected information
- Model Finding (MITRE's CHASE Tool)
  - discovers adversary models consistent with attestation protocols
  - allows evaluation of potential adversary behavior outside the attestation protocol
- Separation Analysis
  - CAmkES specifications define allowed communication
  - synthesize or analyze architectures to evaluate allowed interaction
- Adversary "in a box"
  - analysis specifies what an adversary might do in the presence of the protocol
  - "the box" constrains the adversary making them do things they don't want to
  - balance the level of constraint against the threat



Validation

- Re-targeting Experiments
  - moving attestation infrastructure from among problems
  - moving attestation infrastructure among architectures
- Testbed Development
  - attestation testbed planned for Fall 2021 deployment
  - will include heterogeneous systems from IoT devices to servers
- Public Domain Infrastructure
  - all tools and systems are public domain
  - available on Linux, MacOS, Windows (sort of)



Thank You!

- Colleagues
  - Peter Loscocco (NSA)
  - John Ramsdell (MITRE)
  - Paul Rowe (MITRE)
  - Ian Kretz (MITRE)
  - Sarah Helble (JHUAPL)
  - David Hardin (Collins Aerospace)
  - Konrad Slind (Collins Aerospace)
- Staff
  - Edward Komp
- Students
  - Adam Petz
  - TJ Barclay
  - Grant Jurgensen
  - Anna Fritz
  - Michael Neises
  - Sarah Scott
  - Anna Seib
  - Anna Burns