Annotated Local Interaction Systems

Jon Millen and John Ramsdell
The MITRE Corporation
Protocol Exchange
Autumn, 2008

What's an ALIS?

Annotated Local Interaction System

- Describes interactions between system modules
- Something like a protocol, but...
- Messages are correctly delivered and authenticated
- Module trust is based on measurements, not key possession

Motivation: Trust Research Platform research project

- Multiple Virtual Machines
- Hypervisor-controlled communication
- Enchanced chipset for measurement storage
- "Modules" include VMs and hypervisor

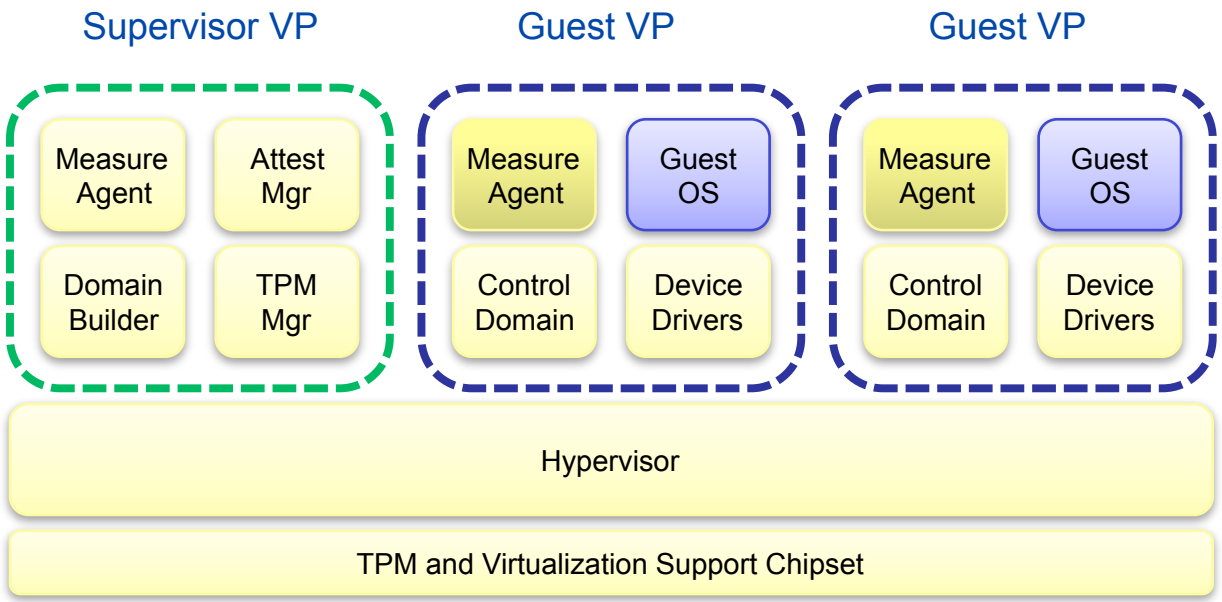
TRP Module Architecture

Virtual Platform (VP) paradigm for domain decomposition

- Communication controlled on VM-to-VM basis

One Supervisory VP, one or more guest VPs

VMs are separated for least privilege, varying trust levels, confidentiality



Example Trust Argument

Modules: Controller (CTL), Measure Agent (MA), OS

CTL objective: determine whether to trust OS, then launch it

Designer-style description of the process:

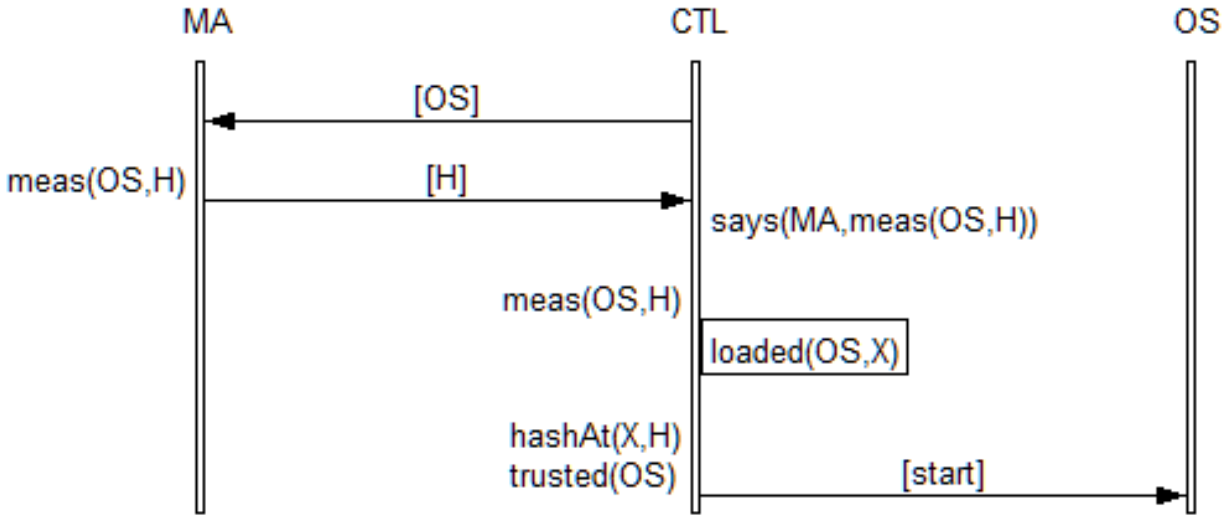
1. CTL obtains OS measurement (hash) from MA, result M
2. CTL obtains OS location from manifest
3. CTL measures OS, result H
4. if H = M, then OS is OK, so launch OS

The trust argument is implicit in this sequence of events

We wish to make trust arguments like this precise and explicit

ALIS Picture

Initial assumption
CTL: trusted(MA)



Protocol-like sequence diagram has formulas annotating nodes

Formulas may be assumed initially, computed, or proved

Prolog-like rules used for inferences:

trusted(OS) is the conclusion of a rule:

loaded(*M,X*) ∧ meas(*M,H*) ∧ hashAt(*X,H*) ⊃ trusted(*M*)

meas(OS,H) is the conclusion of a rule:

trusted(*P*) ∧ says(*P*,meas(*M,H*)) ⊃ meas(*M,H*)

Comparisons

With the annotated strand space protocol model:

- Messages are unstructured data (but extraction functions are allowed)
- Sending principal is known to receiver without crypto arguments
- There are non-message nodes (with call formulas)
- Rely formulas are always sender guarantees with "says"

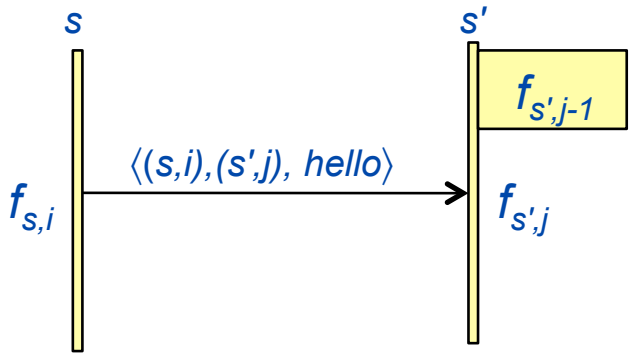
With UML sequence diagrams:

- Formal restrictions on messages, annotations

Abstract Model

An ALIS is a tuple (A,L,I,S) where

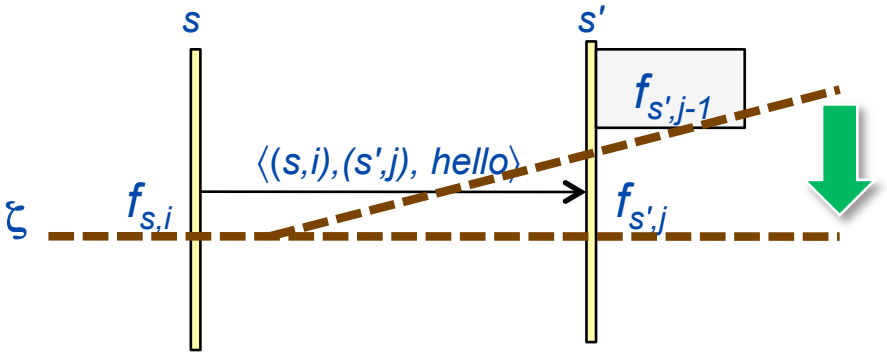
- A is a set of atoms, including principals $P \subset A$
- L is a logical language with "says" and "trusted"
- I is a set of interactions of the form $\langle (s,i), (s',j), m \rangle$ where
  - $s \in S, \quad s' \in S, \quad m \in A^*$
  - with at most one interaction at each node (s,i)
- S is a set of strands s with node formulas $f_{s,i}$ and owner-principal p_s.
  - Some strands are *initialization* roles, others *service* roles



State-Transition Behavior

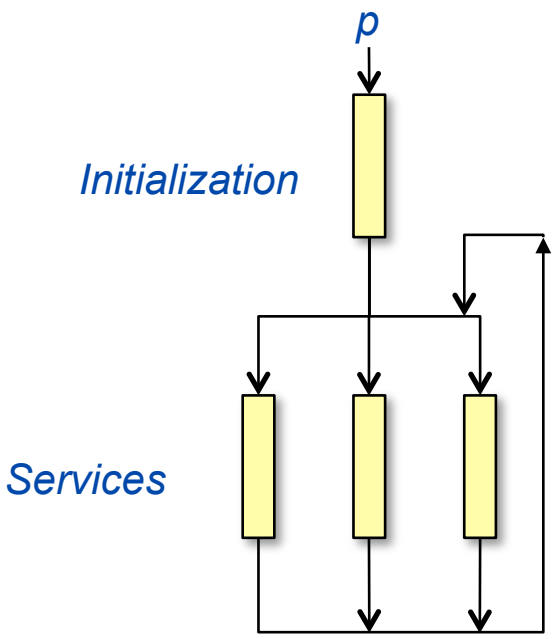
An ALIS has a state-transition structure:

- A state is a function ζ: $S \rightarrow \mathbf{N}$ indicating the current node of each strand
- The initial state is $\zeta_0 = \{(s,0) \mid s \in Z\}$ where $Z \subseteq S$
  - Z is the set of initialization strands, at most one per principal
- A transition replaces (s,i) with $(s,i+1)$ if $f_{s,i+1}$ is provable
  where "provable" means...
- If (s,n) is the last node of s, there may be a transition to $(s',0)$
  - where s' is a service strand having the same owner.

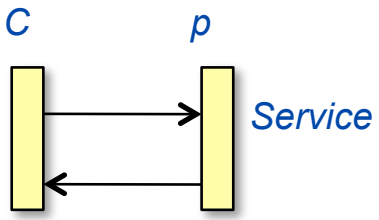


Initialization and Services

For each principal, an ALIS has one initialization strand and zero or more service strands.



Service strands differ from initialization strands because they communicate with unknown (variable) callers.



"Provable"

If the formula is a call or guarantee, it must be implied by prior formulas on the same strand. (Node 0 formulas are just assumed.)

$$\bigwedge_{j<i} f_{s,j} \supset f_{s,i}$$

If the formula is a rely formula *p says* ϕ on (s',j)

then it must match a guarantee on the same message:

i.e., there must be an interaction $\langle (s,i), (s',j), m \rangle$

such that $\phi = f_{s,i}$ and $p_s = p$.

Says-trust meta-rule: $p \text{ says } \phi \wedge \textit{trusted}(p) \supset \phi$

Objective

Desired conclusion: if a state is reachable, the formula annotating each node in it is provable from the initial assumptions of each trusted principal.

(A principal is trusted at a node if trusted(p) is provable there.)

Intended application:

- Express and explain trust arguments
- Identify assumptions and computational needs

Software Support

There is a software tool, written in Prolog, that

- Draws a sequence diagram
- Checks dataflow (parameters are defined before use) and
- Checks usage of rules (presence of hypotheses) and "says"

The tool uses S-expression ALIS specifications

Example Specification

```
(principals MA CTL OS)
(constants start)
(functions meas loaded hashAt)
(imports)

(diagram
  (msg CTL MA (guar)  (data OS)  (rely))
  (msg MA CTL (guar (meas OS H)) (data H) (rely (says MA (meas OS H))))
  (call CTL  (guar (meas OS H) ) (loaded OS X))
  (msg CTL OS
    (guar (hashAt X H )
          (trusted OS))
    (data start)  (rely ))
 )

(lt MA (meas Y H))
(lt CTL (loaded Y Z))
(lt CTL (hashAt X H))
(lt CTL (trusted MA))

(rule (trusted M)  (hyp (meas M H)(loaded M X)(hashAt X H)))
```

Sxdisp Window