HAMES Review at SRI, 7 October 2008 partly based on
Layered Assurance Workshop 13, 14 August 2008, BWI Hilton
and based on Open Group, 23 July 2008, Chicago

Component Security Integration

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Overview

- Clear formulation of the MILS idea
  - That supports Component Security Integration

- Formal basis for MILS Integration

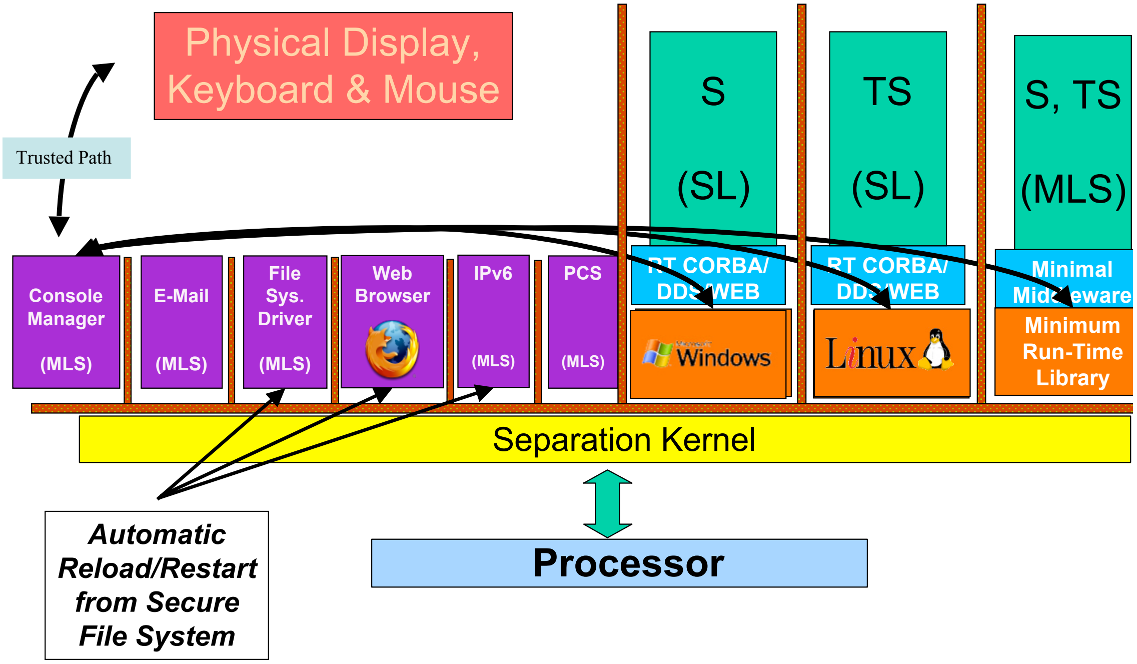
- A worked example

- Support for Protection Profile development
  - The Common Criteria Authoring Environment (CCAE)
    - ⋆ Rance DeLong
  - MILS Network Subsystem Protection Profile (MNSPP)
    - ⋆ Mark Guinther (WindRiver)

The MILS Idea

Traditionally presented as three layers

- Separation kernel, middleware, applications



The MILS Idea (ctd)

- Problem is, that doesn't compose
- i.e., it's not clear how you get a certified MILS system out of certified MILS components and subsystems
- So we developed a MILS component security integration approach based on two layers
- With the idea of MILS policy architecture as the interface
- Learned from OG and LAW meetings that some found three/two layers confusing
- Hence, the paper we will present at DASC explains these

Component Security Integration

- We build systems from components
- And we'd like security properties and assurance/certification to compose
    - That is, assurance for the whole is built on assurance for the components
- Seldom happens: assurance dives into everything
- The system security assurance argument may not decompose on architectural lines
    - So what is architecture?
    - A good one simplifies the assurance case

The MILS Idea (Two Layer Version)

- Construct an architecture so that security assurance does decompose along structural lines

- Two issues in security:
  - Enforce the security policy
  - Manage shared resources securely

- The MILS idea is to handle these separately

- Focus the system architecture on simplifying the argument for policy enforcement
  - Hence policy architecture

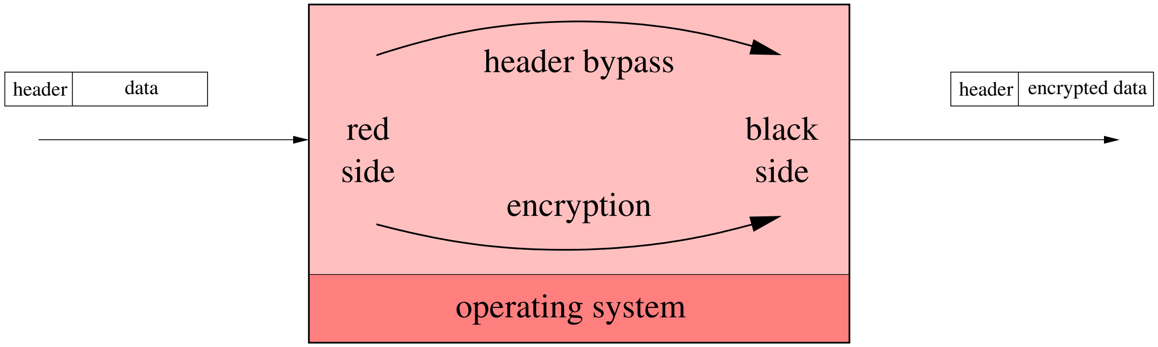
- The policy architecture becomes the interface between the two issues

Policy Architecture

- Intuitively, a boxes and arrows diagram
    - There is a formal model for this
- Boxes encapsulate data, information, control
    - Access only local state, incoming communications
    - i.e., they are state machines
- Arrows are channels for information flow
    - Strictly unidirectional
    - Absence of arrows is often crucial
- Some boxes are trusted to enforce local security policies
- Want the trusted boxes to be as simple as possible
- Decompose the policy architecture to achieve this
- Assume boxes and arrows are free

Crypto Controller Example: Step 1

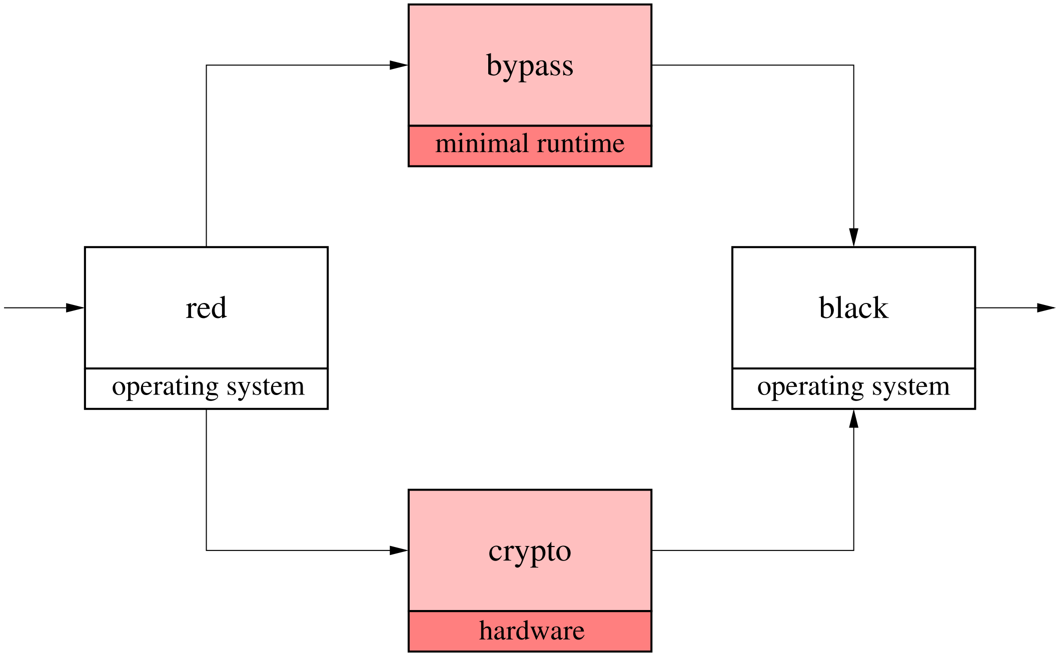
Policy: no plaintext on black network



No architecture, everything trusted

Crypto Controller Example: Step 2

Good policy architecture: fewer things trusted



Local policies (notice these are intransitive):

Header bypass: low bandwidth, data looks like headers

Crypto: all output encrypted

Policy Architecture: Compositional Assurance

- Construct assurance for each trusted component individually
    - i.e., each component enforces its local policy

- Then provide an argument that the local policies
    - In the context of the policy architecture

  Combine to achieve the overall system policy

- Medium robustness: this is done informally

- High robustness: this is done formally
    - Compositional verification

- Cf. layered assurance

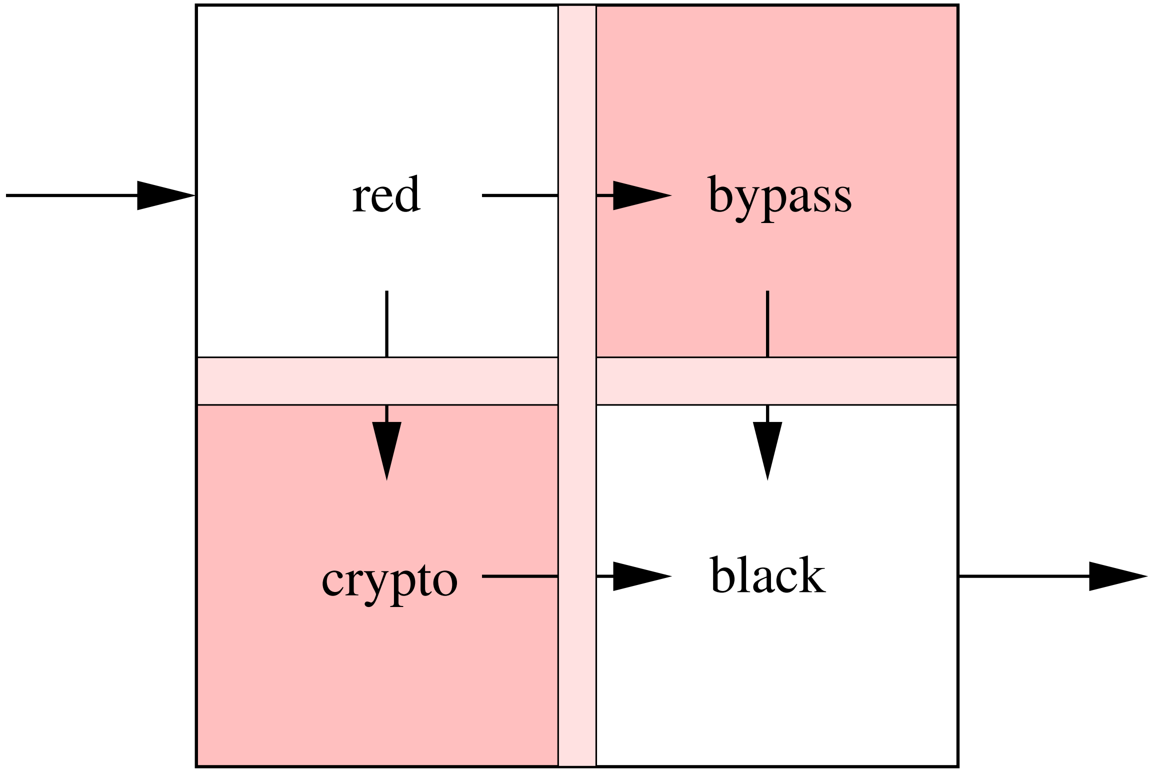
Resource Sharing

- Next, we need to implement the logical components and the communications of the policy architecture in an affordable manner

- Allow different components and communications to share resources

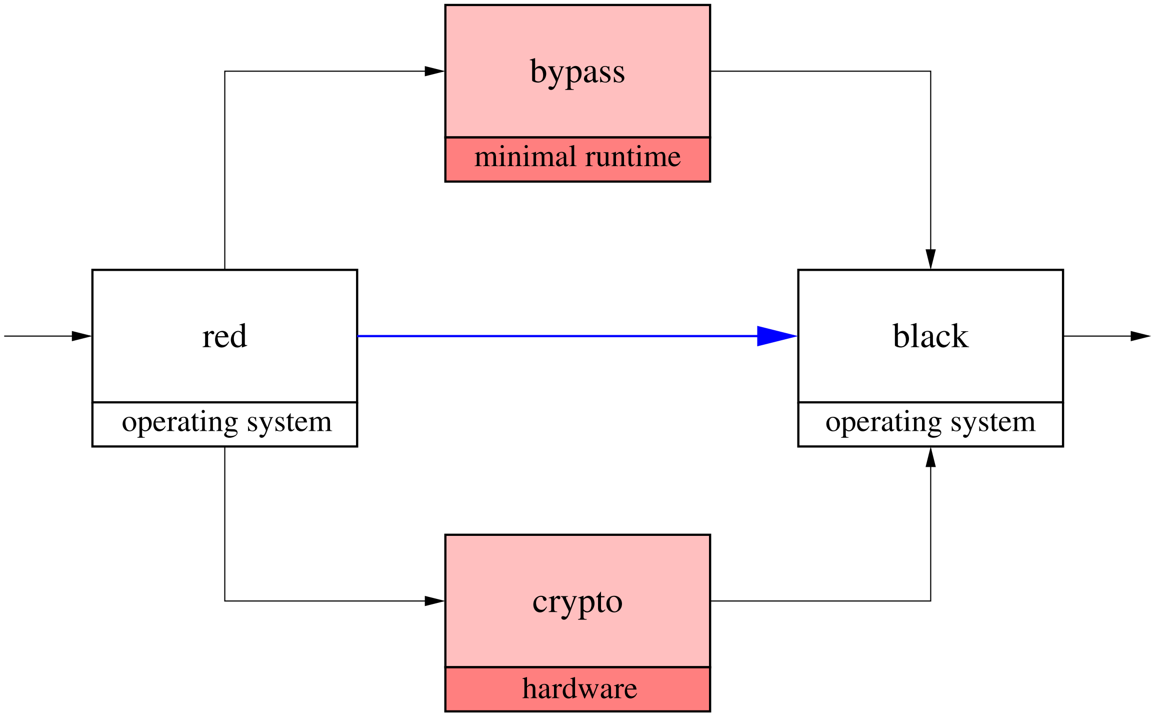
- Need to be sure the sharing does not violate the policy architecture
  - Flaws might add new communications paths
  - Might blur the separation between components

Poorly Controlled Resource Sharing

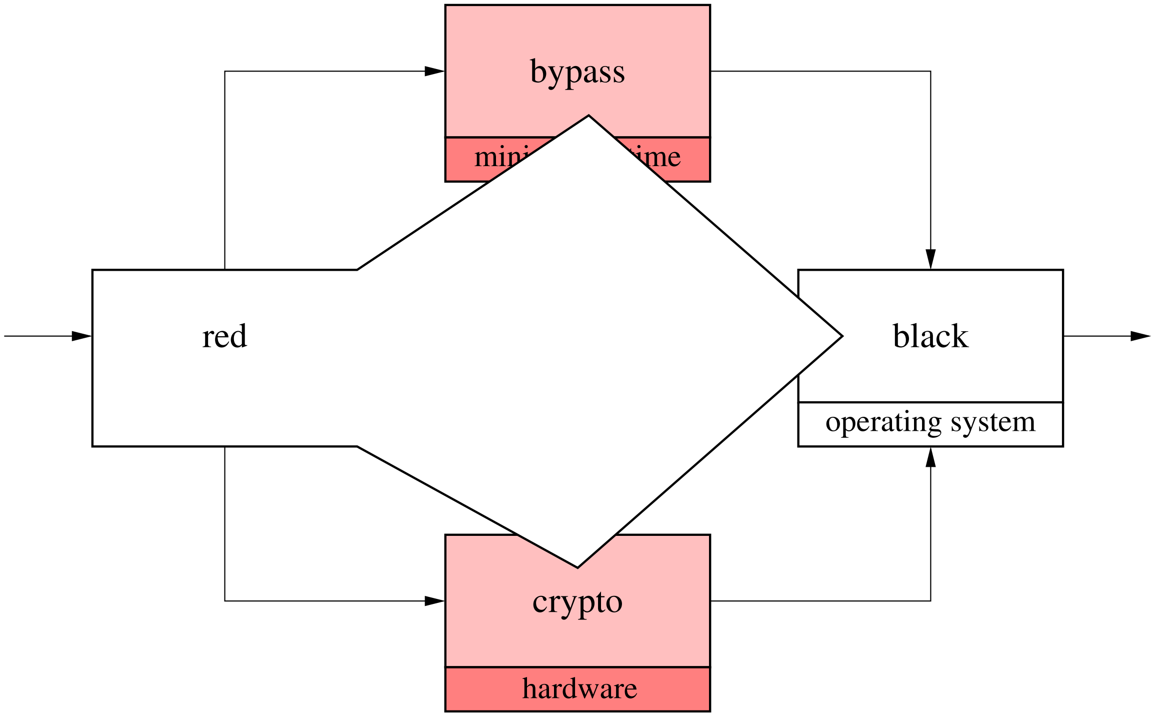


Naive sharing could allow direct red to black information flow,
or could blur the integrity of the components

Unintended Communications Paths



Blurred Separation Between Components

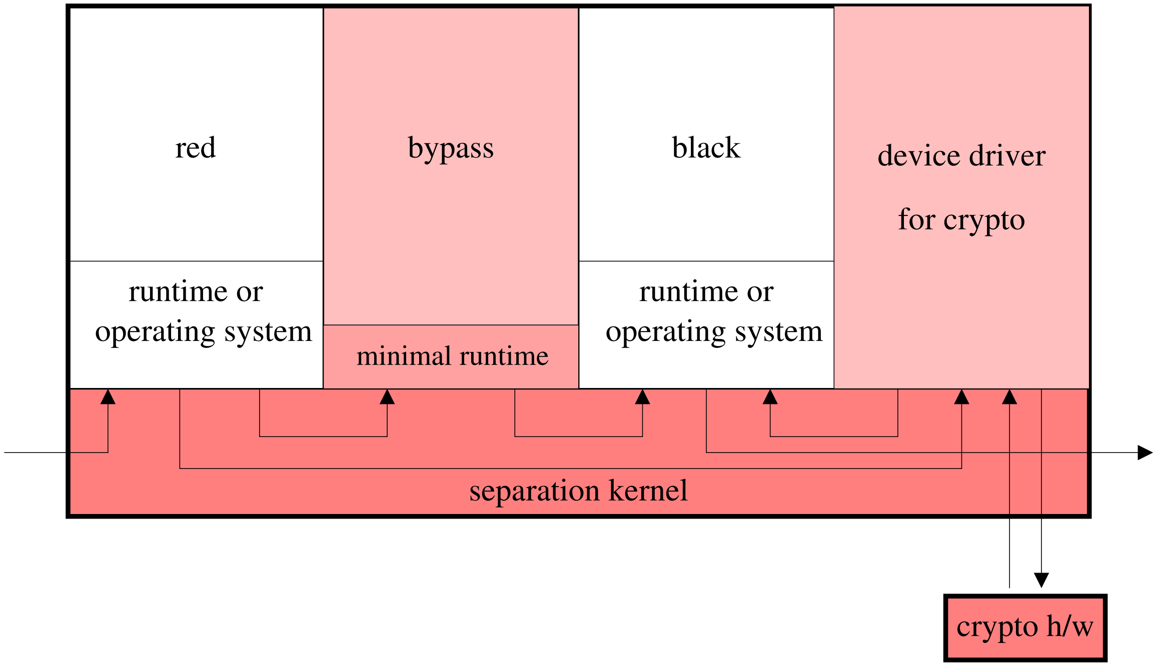


Secure Resource Sharing

- For broadly useful classes of resources
  - e.g., file systems, networks, consoles, processors
- Provide implementations that can be shared securely
- Start by defining what it means to partition specific kinds of resource into separate logical components
- Definition in the form of a protection profile (PP)
  - e.g., separation kernel protection profile (SKPP)
  - or network subsystem PP, filesystem PP, etc.
- Then build and evaluate to the appropriate PP

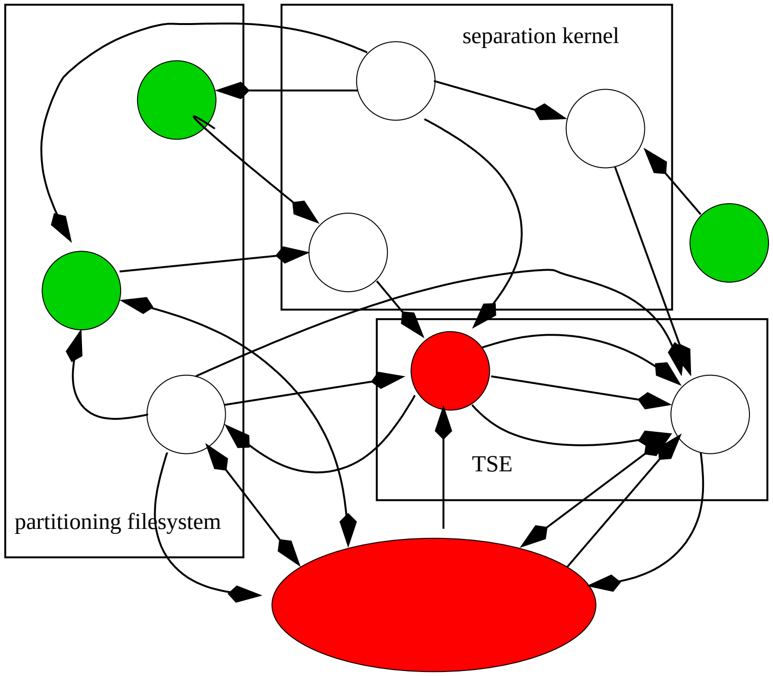
Crypto Controller Example: Step 3

Separation kernel securely partitions the processor resource



The integrity of the policy architecture is preserved

A Generic MILS System



Care and skill needed to determine which logical components share physical resources (performance, faults)

Resource Sharing: Compositional Assurance

- Construct assurance for each resource sharing component individually
  - i.e., each component enforces separation

- Then provide an argument that the individual components
  - Are additively compositional

  And therefore combine to create the policy architecture

- Medium robustness: this is done informally

- High robustness: this is done formally
  - Compositional verification

- Cf. layered assurance

MILS Business Model

- DoD moves things forward by supporting development of protection profiles
  - Separation kernels, partitioning communications systems, TCP/IP network stacks, file systems, consoles, publish-subscribe
- Then vendors create a COTS marketplace of compliant components
- Currently they are all resource sharing components
- Should be some policy components, too
  - E.g., filters, downgraders for CDS
    - ⋆ Could be a standardized CDS engine, many rule sets
    - ⋆ Rule sets derived from goals, not hand coded
    - ⋆ e.g., Ontologically-driven purpose and anti-purpose
  - Or even MLS

MILS In The Enterprise

- Separation kernels are like minimal hypervisors (cf. Xen)
  - MILS separation kernel (4 KSLOC), EAL7
  - Avionics partitioning kernel (20 KSLOC),
    DO-178B Level A (≈ EAL4)
  - Hypervisor (60–250 KSLOC), EAL?
- Can expect some convergence of APIs (cf. ARINC 653)
- Different vendors will offer different functionality/assurance tradeoffs
- Can extend use of hypervisors from providing isolated virtual hosts to supporting the policy architecture of a secure service

Recent Progress

- Initial development of mathematical theory for compositional assurance of MILS systems
- Technical report available
- Policy integration
- Resource sharing integration

Policy Integration

- Need to specify what it means for a component to satisfy a policy under assumptions about its environment

- Then show how these compose (policy of one component becomes the assumptions of anther)

- Fairly standard Computer Science, MILS is agnostic on the exact approach used
  - Policies/assumptions as properties
  - Or as abstract components

Resource Sharing Integration

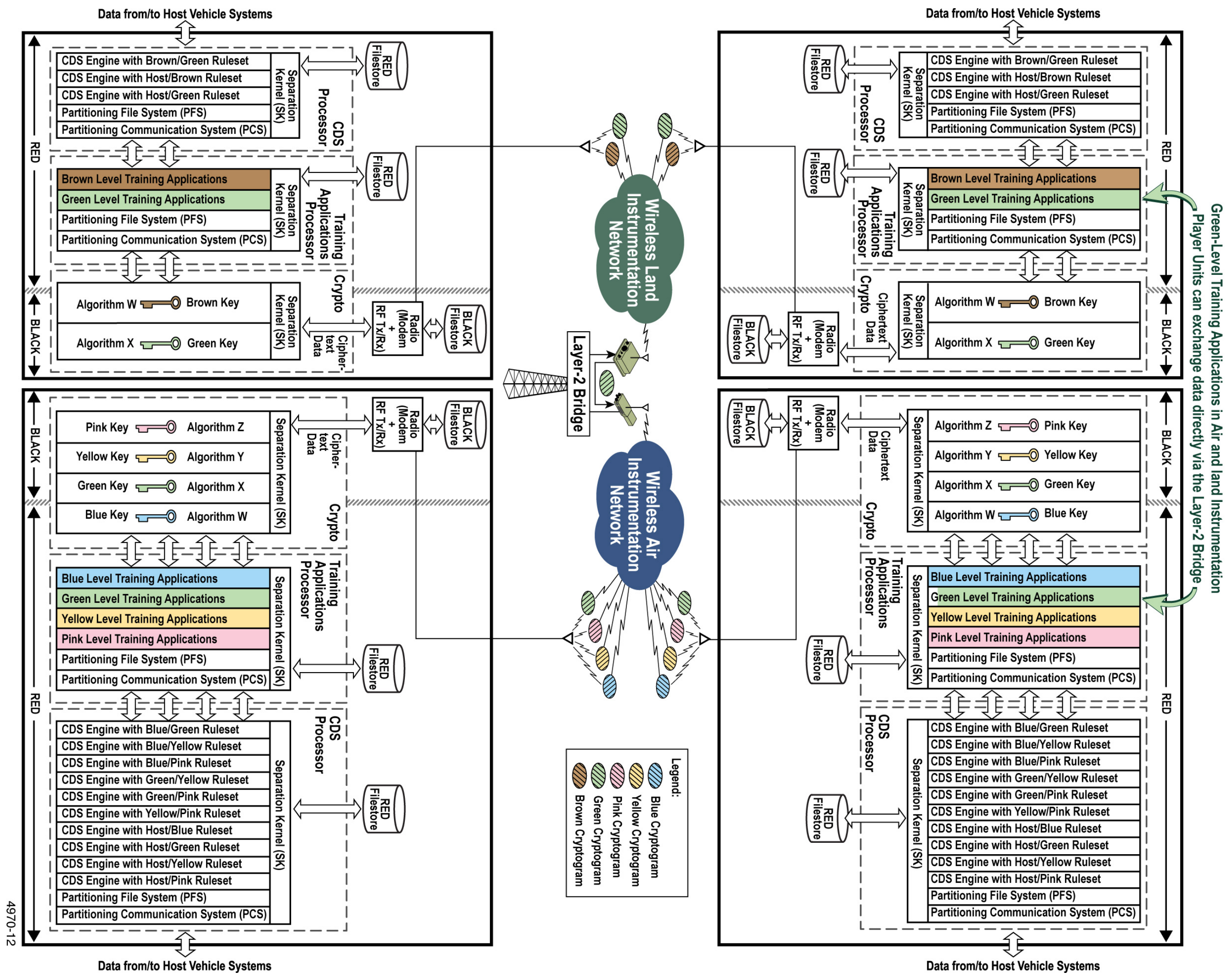
- Formal policy architecture model
- Components are state machines
- Communications channels are shared variables
- Asynchronous composition
- Definition of well-formed policy architecture
- And of implementation respecting and enforcing a policy architecture
- Argument that these compose

Worked Example

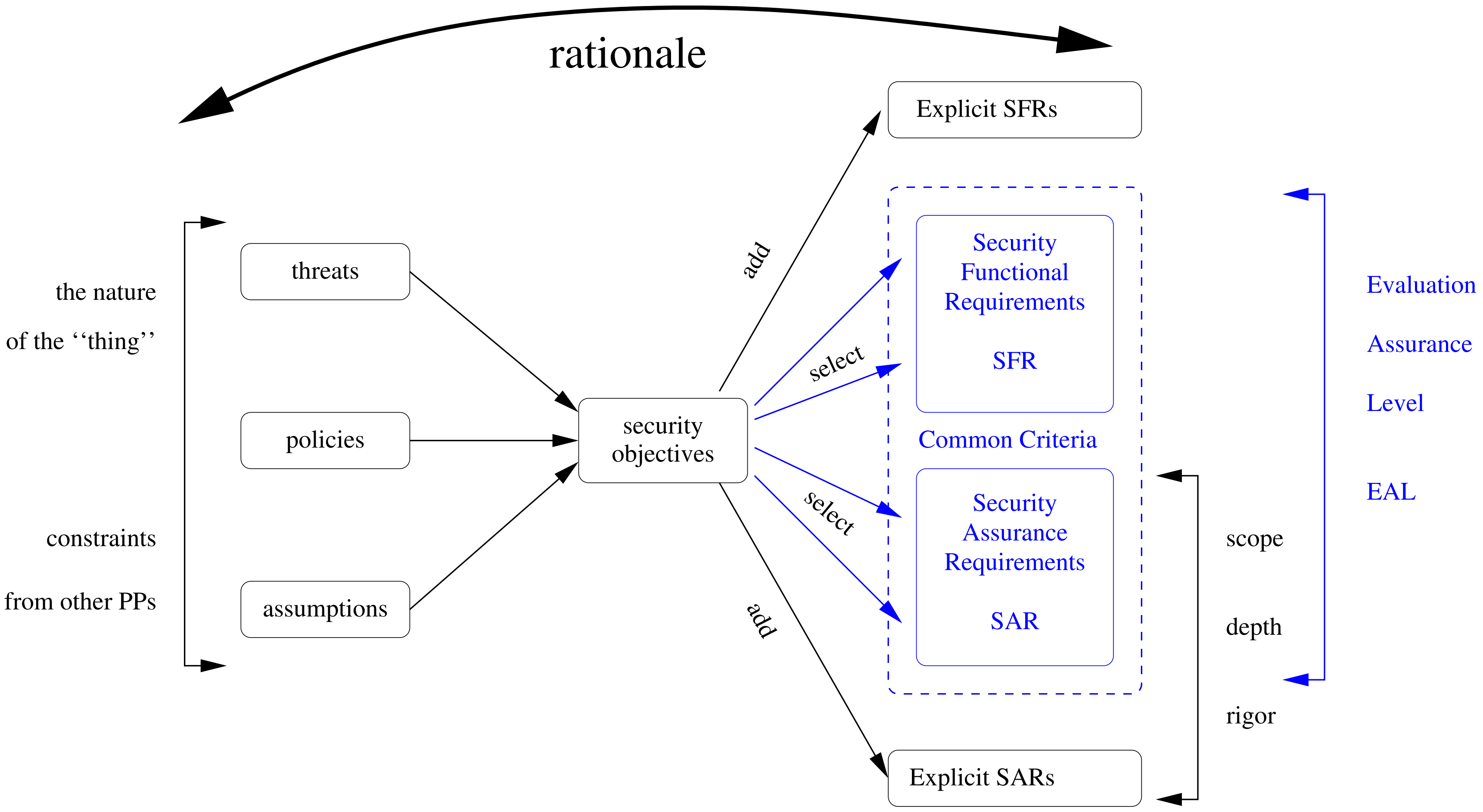
- Cross Domain Sharing for joint training exercises
  - JTRS radios at bottom

- With Dave Hanz, SRI ESD
  - An example for us
  - A MILS roadmap for them

MILS Architecture for Joint Training Exercises


Data from/to Host Vehicle Systems
CDS Engine with Brown/Green Ruleset
CDS Engine with Host/Brown Ruleset
CDS Engine with Host/Green Ruleset
Partitioning File System (PFS)
Partitioning Communication System (PCS)
Separation Kernel (SK)
CDS Processor
RED Filestore
Brown Level Training Applications
Green Level Training Applications
Training Applications Processor
Algorithm W
Brown Key
Algorithm X
Green Key
Crypto
Ciphertext Data
Radio (Modem + RF Tx/Rx)
BLACK Filestore
RED
BLACK
Wireless Land Instrumentation Network
Layer-2 Bridge
Wireless Air Instrumentation Network
Pink Key
Algorithm Z
Yellow Key
Algorithm Y
Blue Key
Blue Level Training Applications
Yellow Level Training Applications
Pink Level Training Applications
CDS Engine with Blue/Green Ruleset
CDS Engine with Blue/Yellow Ruleset
CDS Engine with Blue/Pink Ruleset
CDS Engine with Green/Yellow Ruleset
CDS Engine with Green/Pink Ruleset
CDS Engine with Yellow/Pink Ruleset
CDS Engine with Host/Blue Ruleset
CDS Engine with Host/Green Ruleset
CDS Engine with Host/Yellow Ruleset
CDS Engine with Host/Pink Ruleset
Green-Level Training Applications in Air and land Instrumentation Player Units can exchange data directly via the Layer-2 Bridge
Legend:
Blue Cryptogram
Yellow Cryptogram
Pink Cryptogram
Green Cryptogram
Brown Cryptogram
4970-12

Protection Profile Development



Protection Profile Topics

- Presentation by Rance DeLong on the Common Criteria Authoring Environment (CCAE) to assist construction of coherent PPs

- Presentation by Mark Guinther (WindRiver) of progress on MILS Network Subsystem Protection Profile (MNSPP)

Looking Forward: Needs

- Completed roadmap and example

- Notions/mechanisms for MILS-coherent PPs
  - Further development of the formal basis

- Complete suite of resource-sharing PPs

- Policy PPs, notably CDS
  - Further development of the formal basis
  - Ontologically-driven purpose and anti-purpose for CDS

- Dialog with CC V4