Modeling Key Distribution

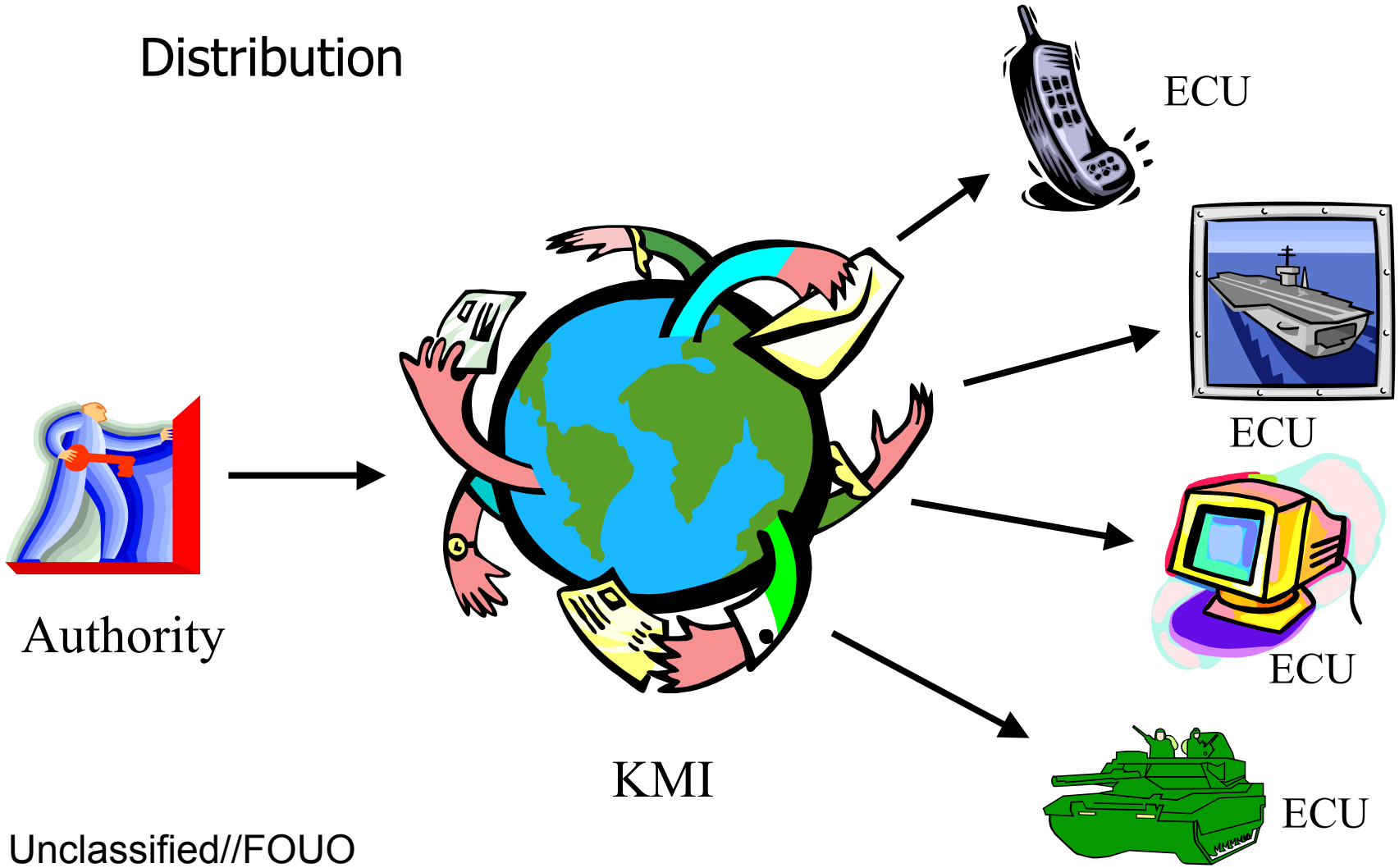
Eric Bush
Kestrel Technology LLC
April 15, 2004

The NSA Context

- Formal methods in the real world
  - Research-initiated systems have no Agency owner
  - Existing missions already have (low-level) implementations
- KMI Modernization:Existing Agency effort at an early stage
- Conceptual engineering
  - Sometimes no fact of the matter
  - Neurath's Boat
  - Philosopher/logician + domain expert

The KMI Domain

Kestrel Technology LLC



Unclassified//FOUO

The KMI Domain

Management

Authority

Authorization

Ordering

Generation

Protection

Changing

Destroying

KMI

ECU

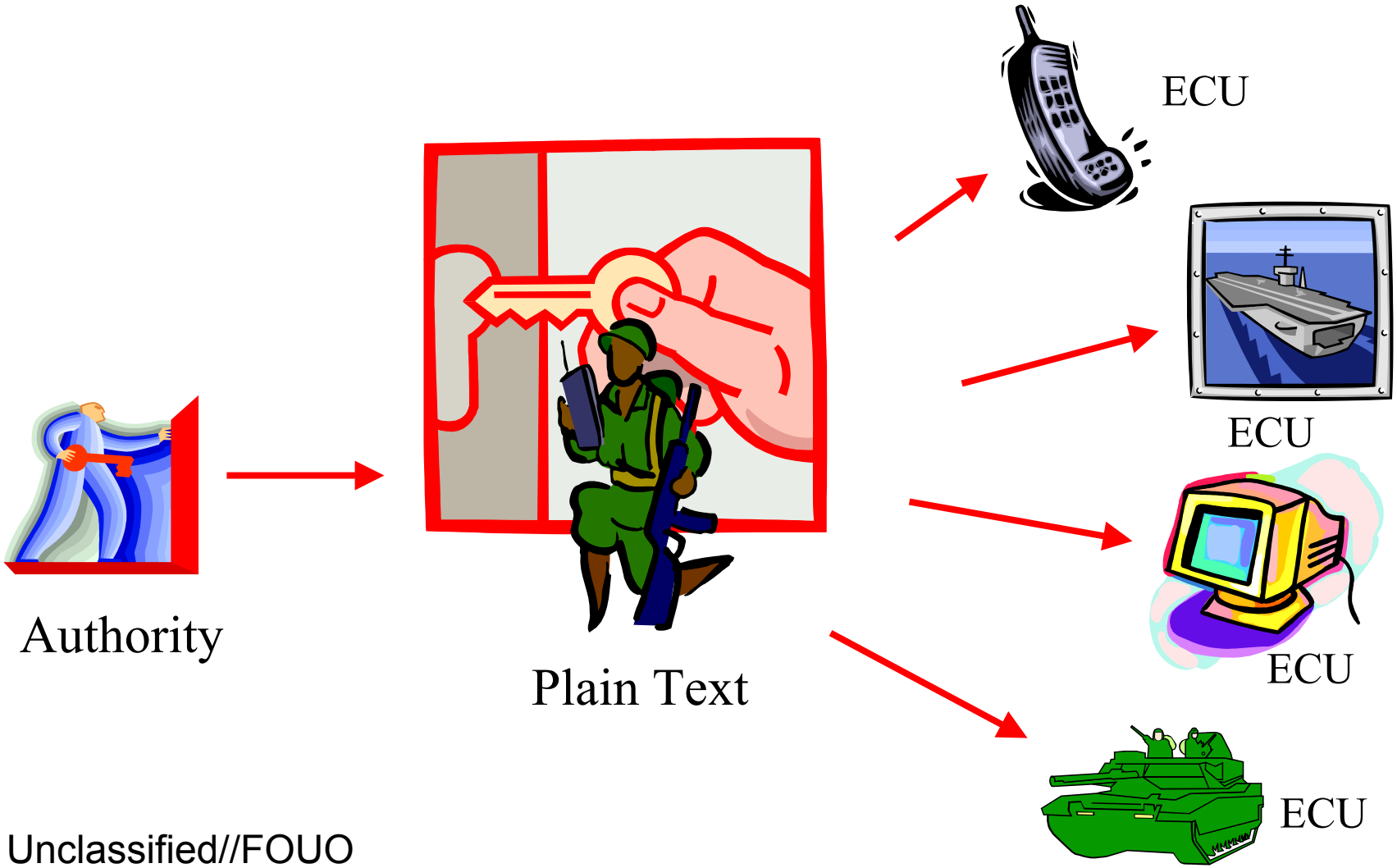
ECU

ECU

ECU

Modernization: Red Fill

Kestrel Technology LLC



Modernization : Black Fill

Kestrel Technology LLC

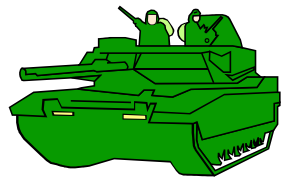
Authority

Cipher Text

ECU

ECU

ECU

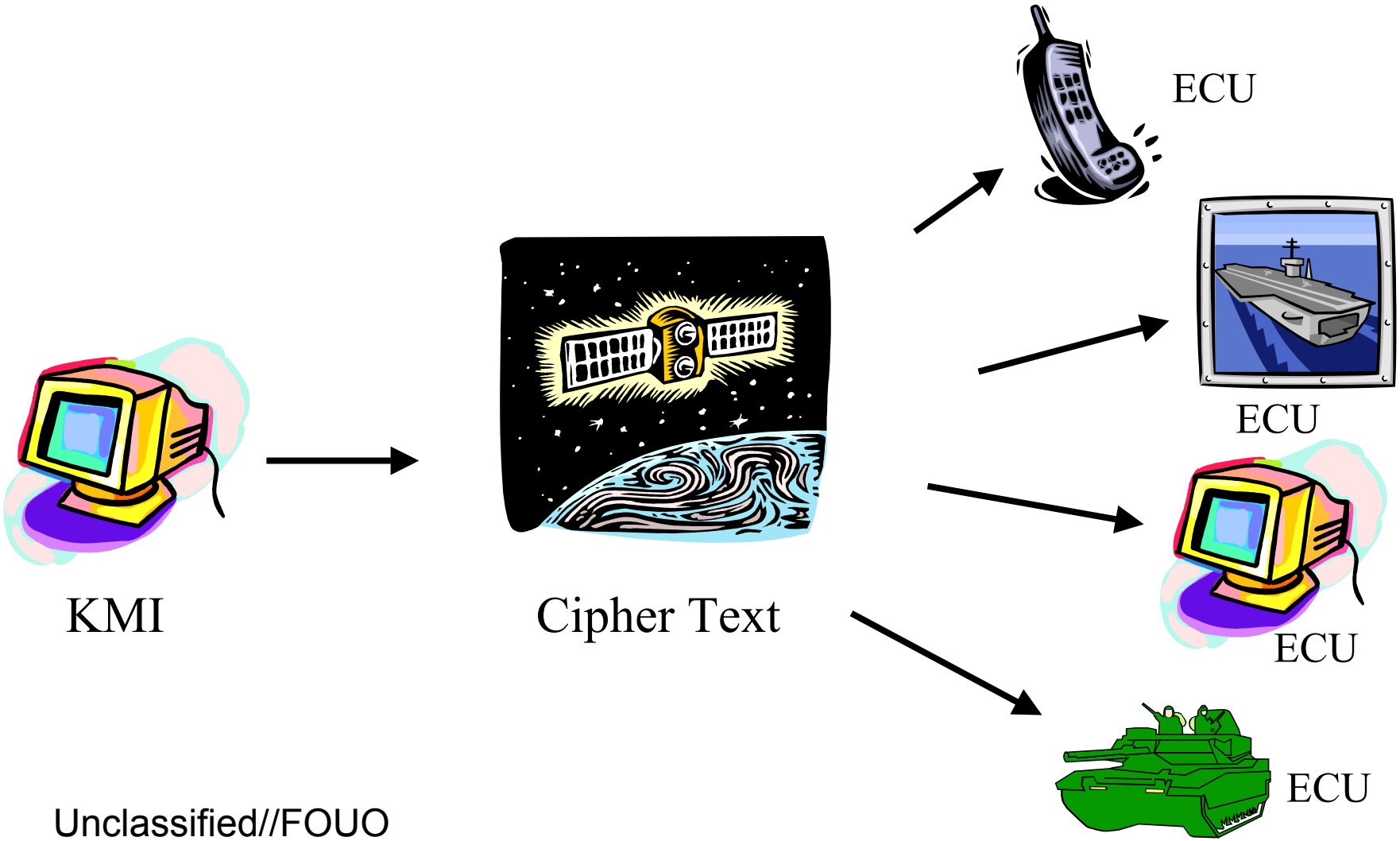


ECU

Unclassified//FOUO

Modernization : Benign Fill

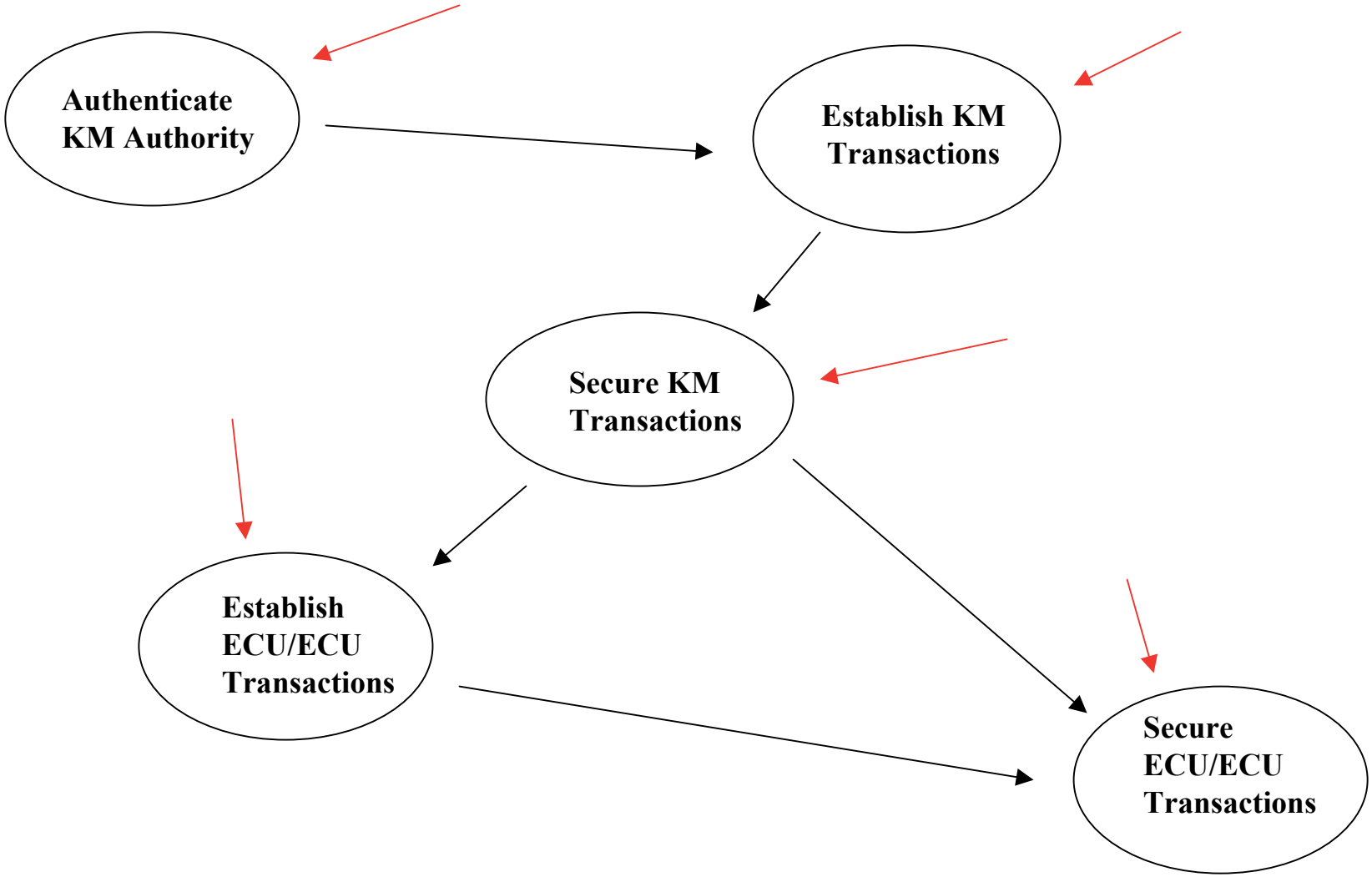
Kestrel Technology LLC



Need for a Common Model

- No language for expressing KMI requirements independent of solutions
- "Stovepipe" solutions
- Requirements not abstract enough
- No evaluation of alternatives
- No analysis of requirements/solution match

Authorization/Capability Model



Existing Effort

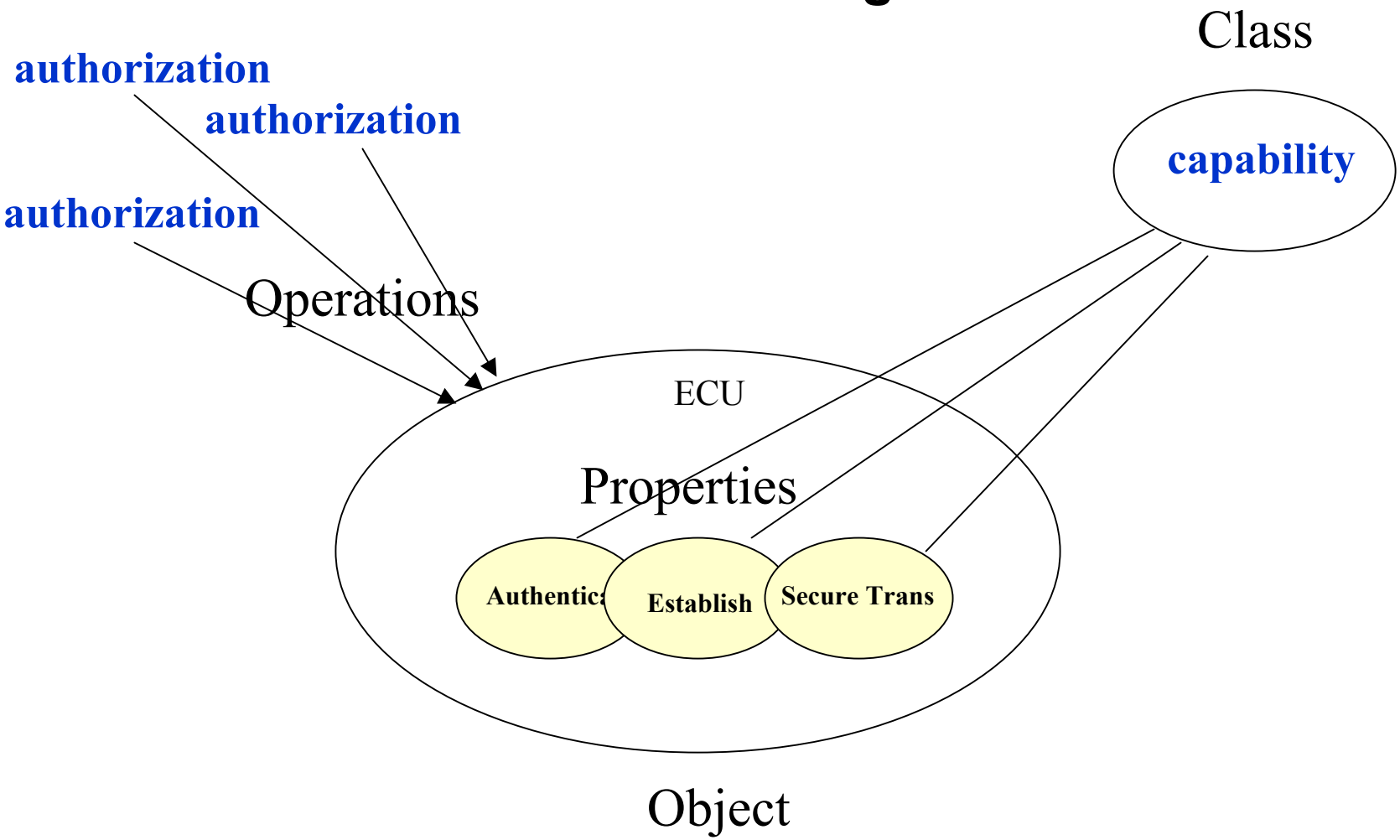
- *Managed Object* methodology
- Natural language distribution
- Informal analysis program

Benefits of Formalization

- Formal modeling done in parallel
- Improve model as it is created
- Formal verification
- Software synthesis

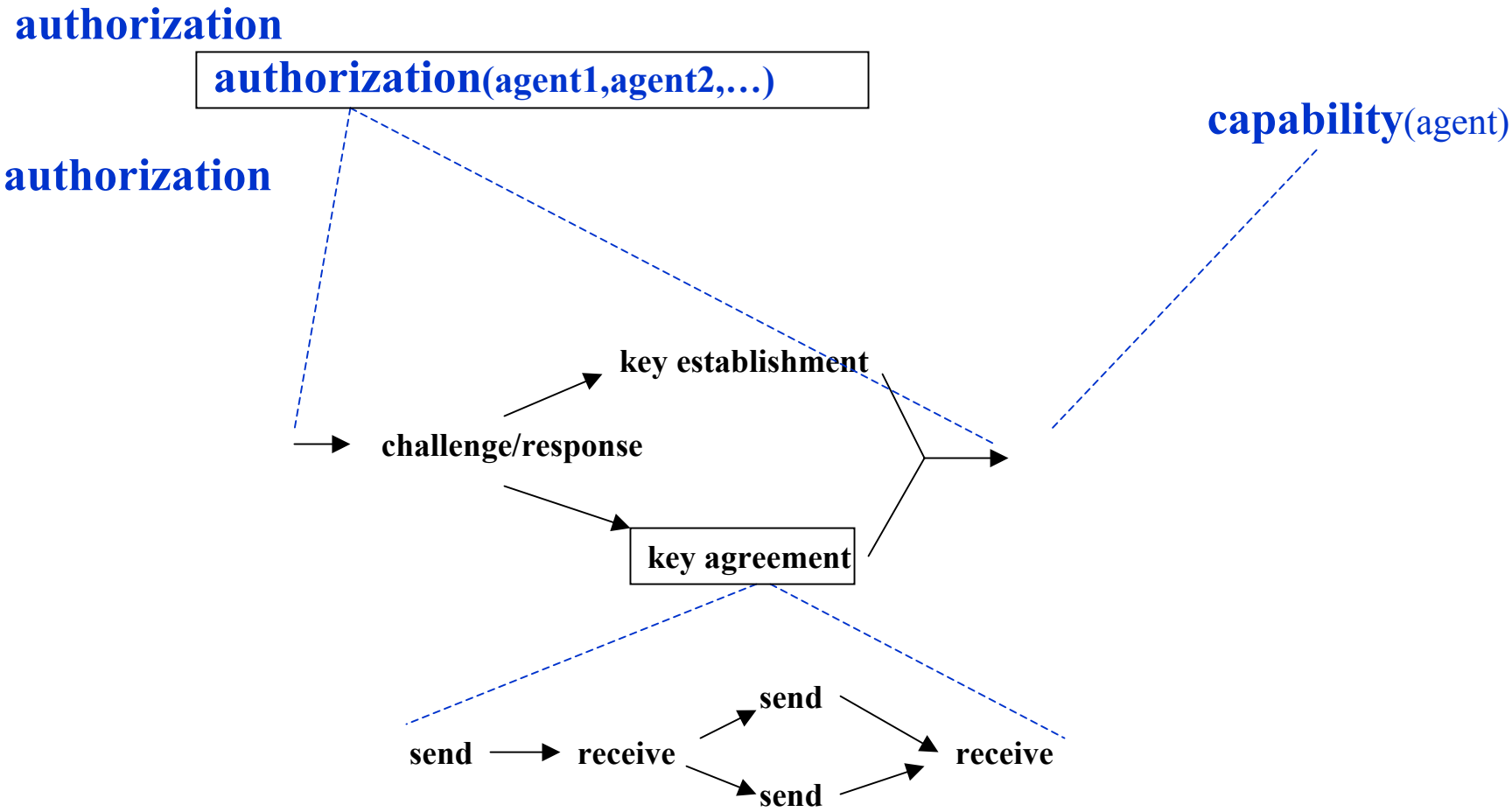
Managed Object Architecture

Informal: OO Design



Abstract Protocols Architecture

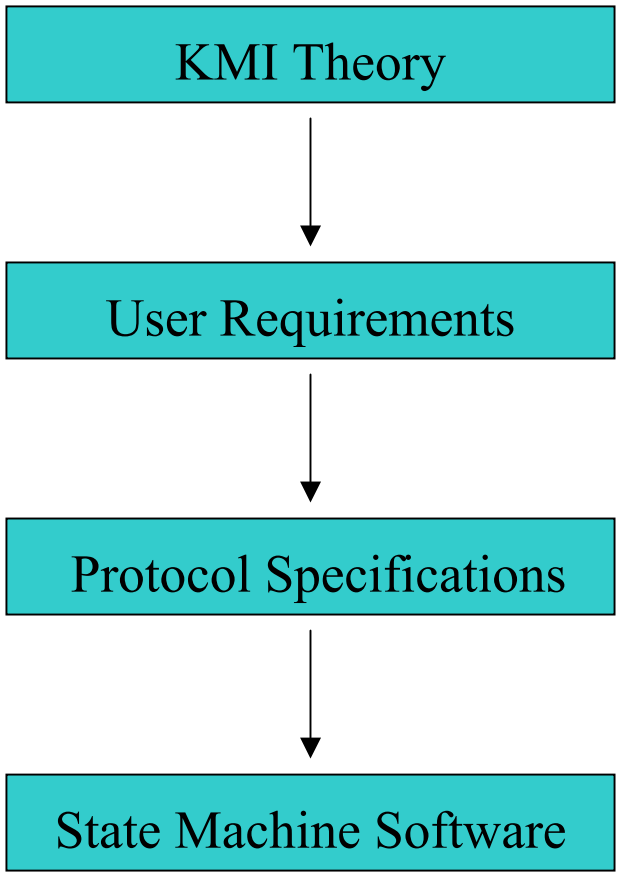
Formal: 1st Order Predicates



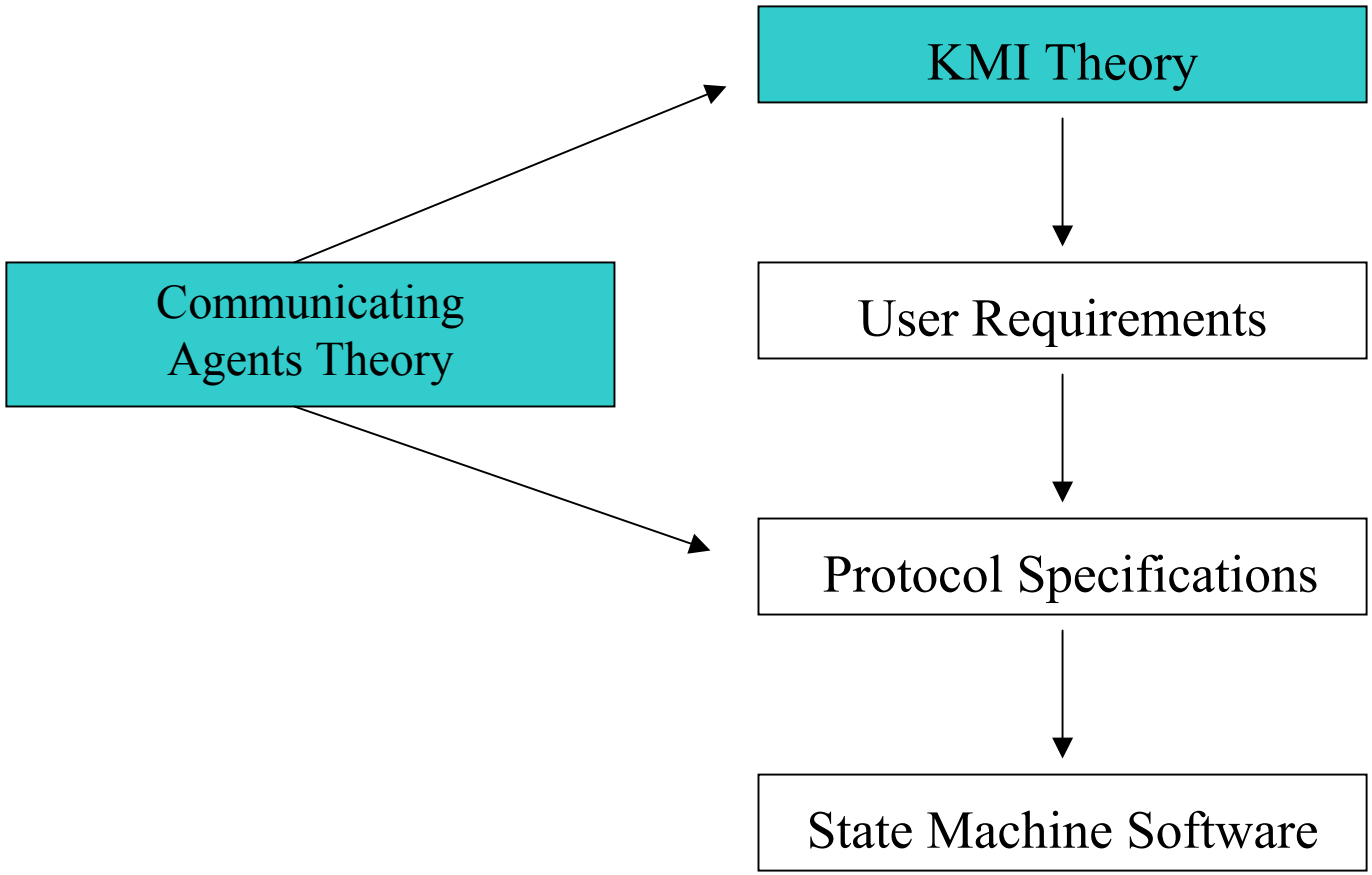
Specware

- Many-sorted, higher-order logic
- ML-like function definition
- Specs = axiomatic theories
- Composition & refinement via morphisms
- Generated proof obligations
- Code synthsis in Lisp, C, Java

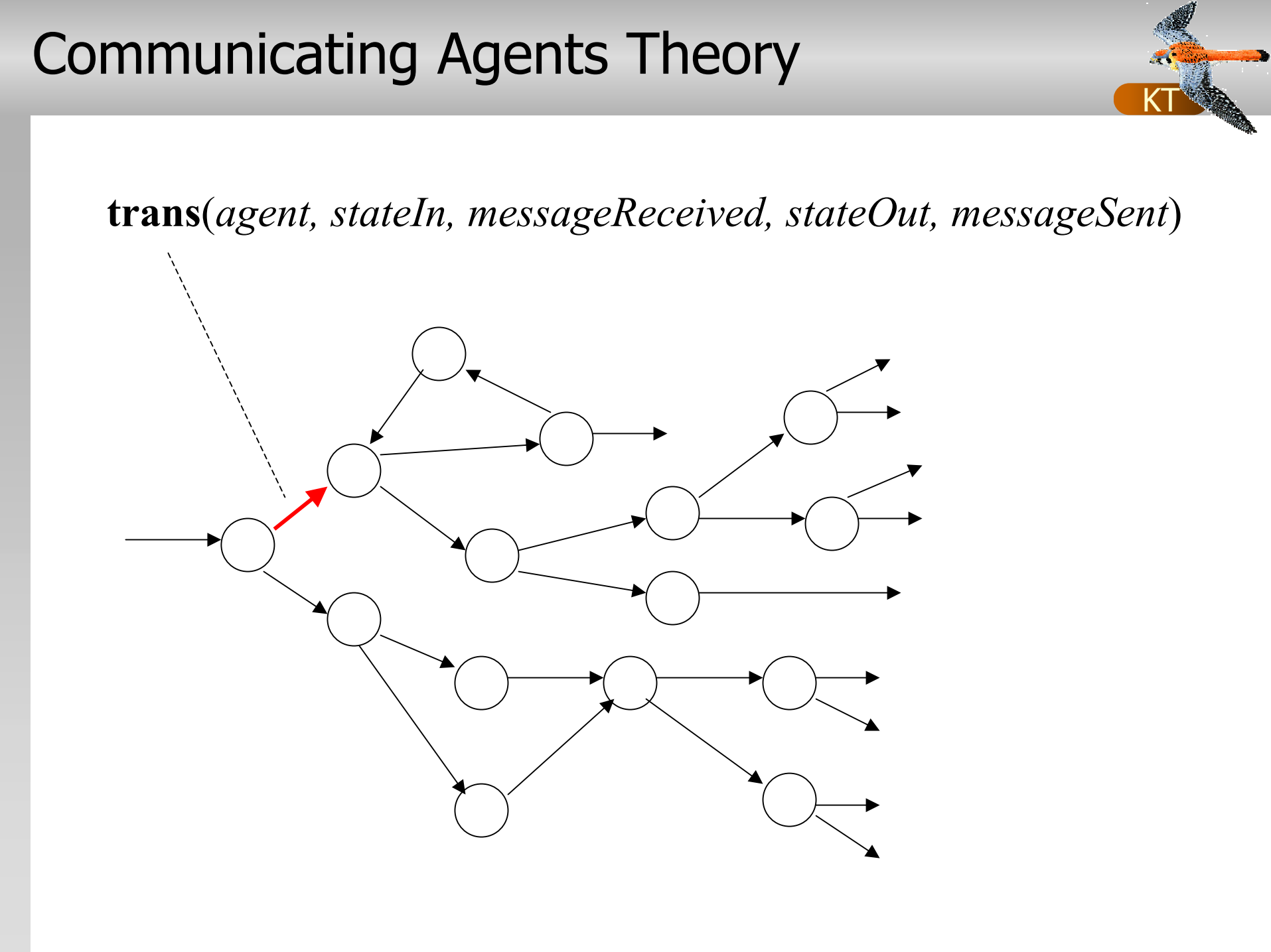
Long Term Architecture



Two Specware Theories



Communicating Agents Theory

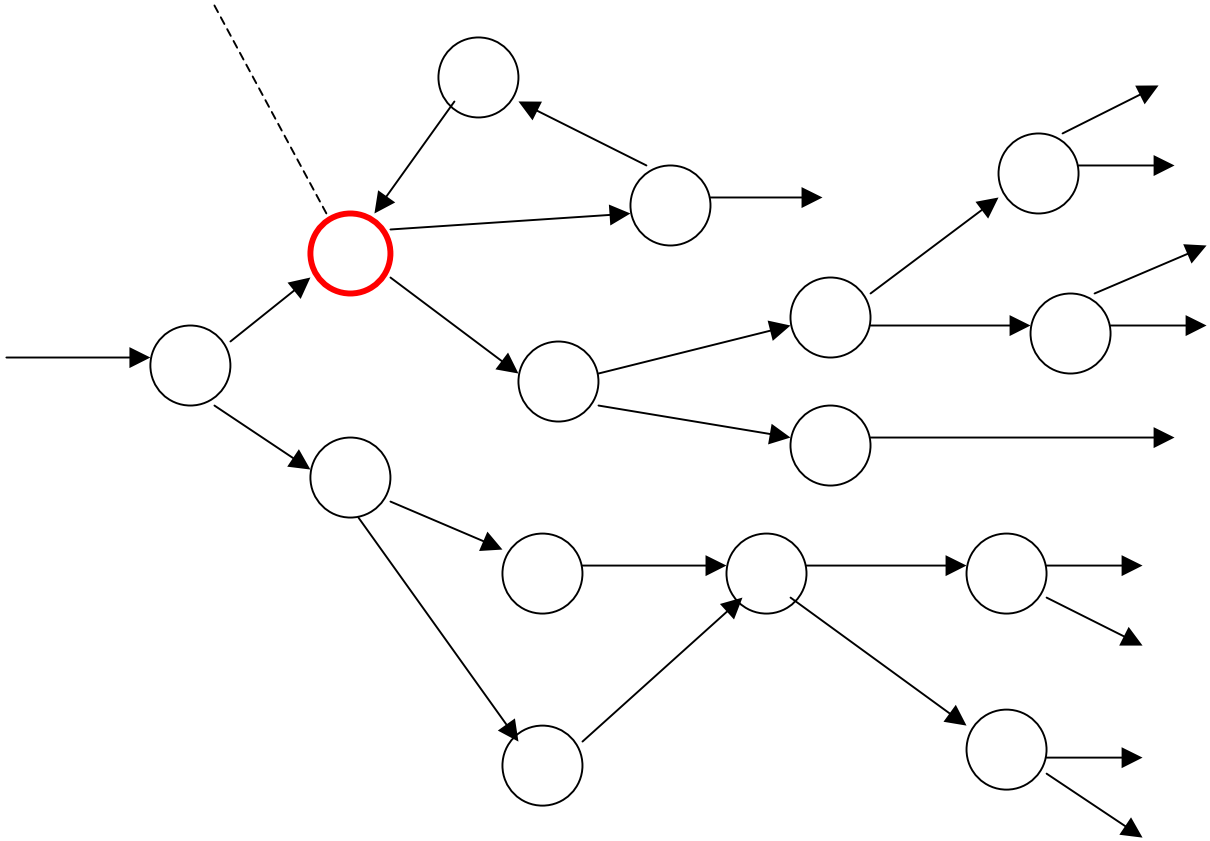


trans(*agent, stateIn, messageReceived, stateOut, messageSent*)

Communicating Agents Theory

sort AgentSlice = Agent * FSeq(Message) * State * Option(Message)

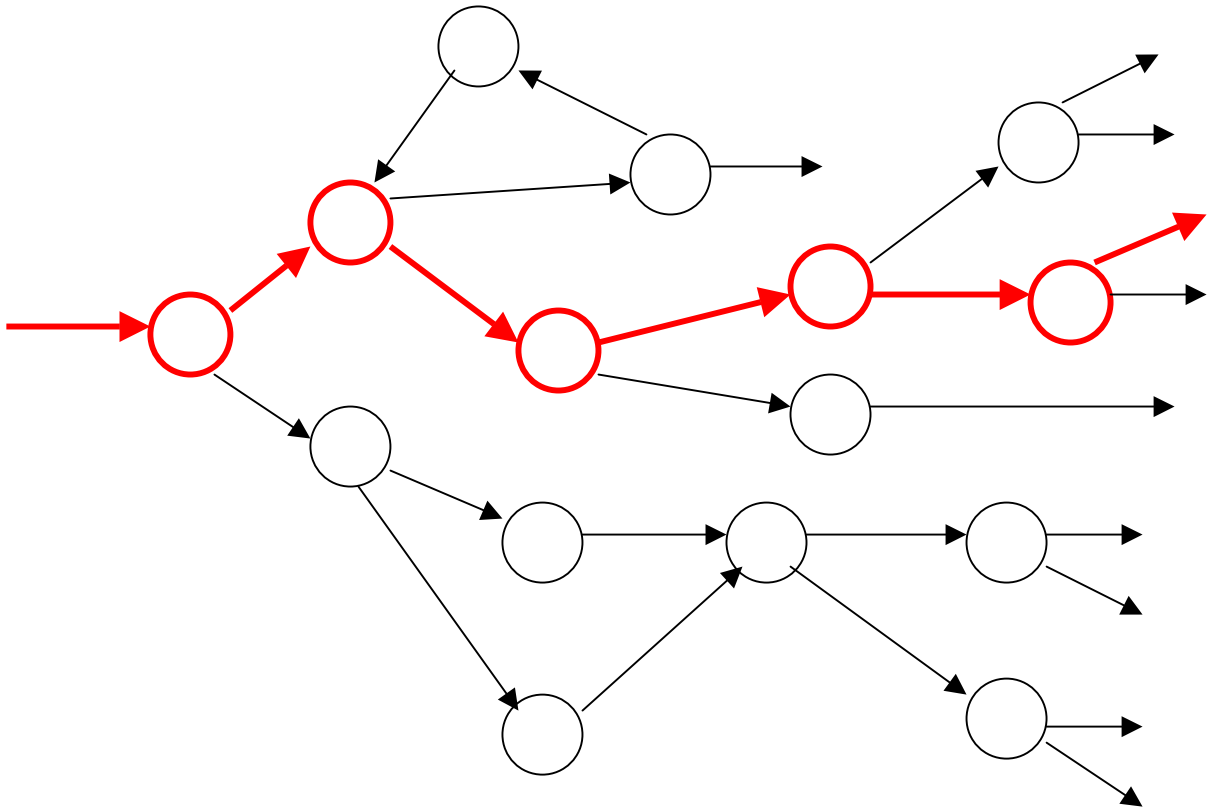
sort TraceSlice = FSet(AgentSlice)



Communicating Agents Theory

sort **Trace** = FSeq(**TraceSlice**)

possibleTrace(*trace*)



KMI Theory: Agents

- Authorities
  - Humans at the periphery of KMI
  - State not modeled
  - Actions are atomic, self-initiated
- ECUs (end cryptographic unit)
  - Electronic agents at and near the periphery of KMI
  - State is modeled
  - End consumers of KM transactions
- KDCs (key distribution center)
  - Electronic agents at the center of KMI
  - State is modeled
  - Can perform all actions except initiation of authority and end-mission operations

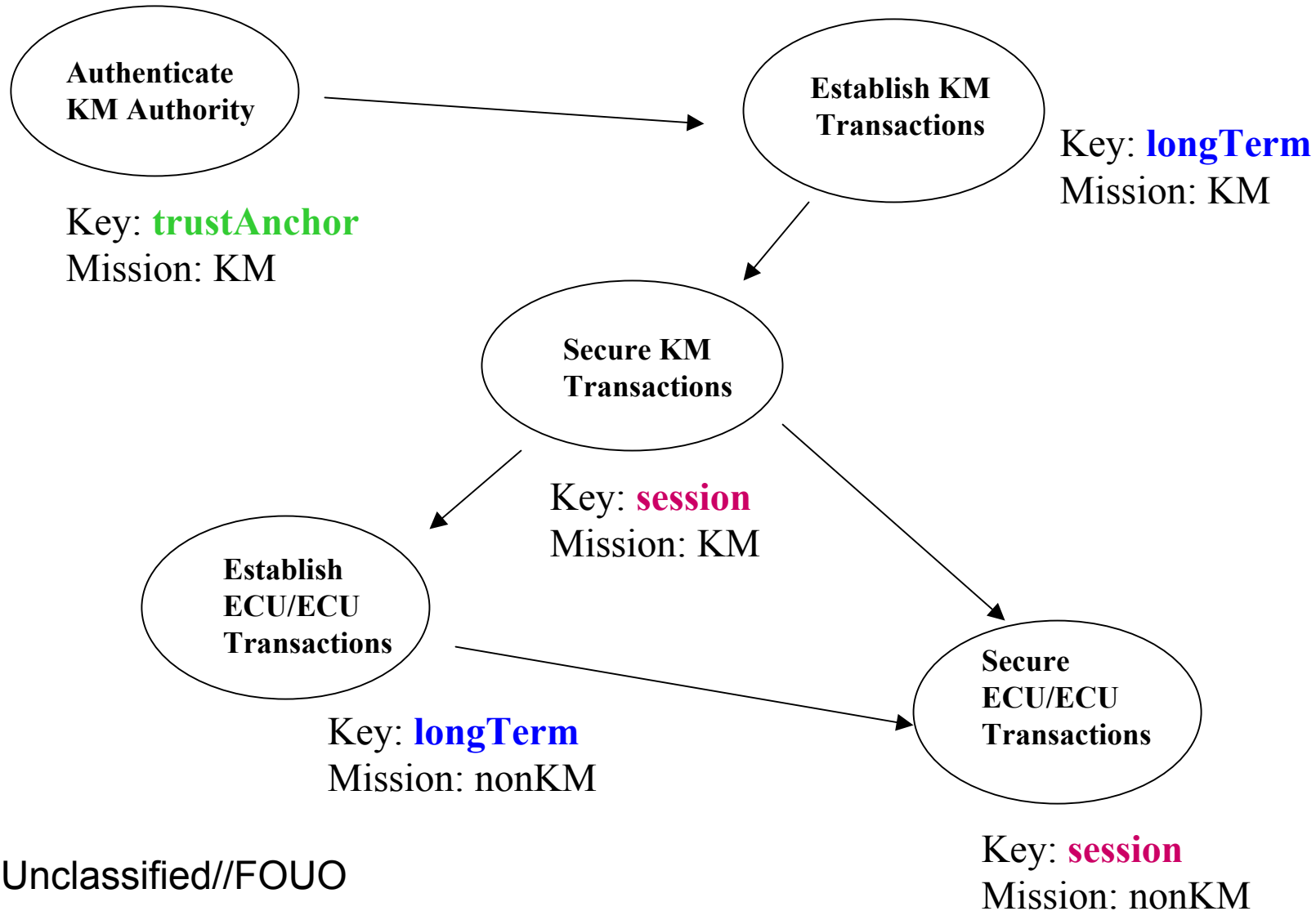
KMI Theory: Keys

- "Shared secrets" role replaced by *Mission*
- Generalizes *public* and *symmetric* key concepts
- Three kinds:
  - sessionKey?(*mission, key, trace*)
  - longTermKey?(*mission, key, trace*)
  - trustAnchor?(*mission, key, trace*)

KMI Theory: Missions

- Primary concept explicating *shared* ECU capabilites
- Currently: sets of agents, but sets not unique
- Partial characterization:
  - kmMission?(*mission*)
  - missionAuthority?(*mission,agent,trace*)
  - sessionKey?(*mission, key, trace*)
  - longTermKey?(*mission, key, trace*)
  - trustAnchor?(*mission, key, trace*)

KMI Theory: ECU Capabilities



KMI Theory: ECU Capabilities

Predicates:

```
op authenticateCapability: ECU * KMIAgent * Mission *
                                    Trace -> Boolean
op estKmTransCapability: ECU * Mission * Trace -> Boolean
op kmTransCapability: ECU * Mission * Trace -> Boolean
op estMissionCapability: ECU * Mission * Trace -> Boolean
op missionCapability: ECU * Mission * Trace -> Boolean
```

Example Axiom:

```
∀(ecu,mission,t) missionCapability(ecu,mission,t) ≡
                        hasSessionKey(ecu,mission,t) ∧
                        ¬kmMission?(mission)
```

KMI Theory: Key Possession and Authorization

Predicates:

```
op hasSessionKey: EAgent * Mission * Trace -> Boolean

op hasLongTermKey: EAgent * Mission * Trace -> Boolean

op hasTrustAnchor: EAgent * KMIAgent * Mission * Trace ->
                                                  Boolean

op authorized: KDC * Capability * Mode * EAgent *
                                  Mission * Trace -> Boolean

op delegated: KDC * KDC * Capability * Mode * EAgent *
                                    Mission * Trace -> Boolean
```

KMI Theory: Key Possession and Authorization

Example Axiom:

```
∀(agent1,mission,t)
        hasSessionKey(agent1,mission,t) ≡
            ∃(agent2,t1,t0)
                extends(t,t1) ∧
              establishesSessionKey(agent2,agent1,mision,t1,t0) ∧
                ¬∃(t2,key)
                  extends(t,t2) ∧
                    extends(t2,t1) ∧
                    sessionKey?(mission,key,t1) ∧
                  expires(key,t2)
```

KMI Theory: Agent Actions

Predicates:

```
op establishesSessionKey: KMIAgent * EAgent * Mission *
                                                Trace * Trace -> Boolean
op establishesLongTermKey: ManagementAgent * EAgent * Mission *
                                                Trace * Trace -> Boolean
op revokesLongTermKey: ManagementAgent * EAgent * Mission *
                                                Trace * Trace -> Boolean
op establishesTrustAnchor: ManagementAgent * EAgent * KMIAgent *
                                        Mission * Trace * Trace -> Boolean
op revokesTrustAnchor: ManagementAgent * EAgent * KMIAgent *
                                        Mission * Trace * Trace -> Boolean
op authorizes: ManagementAgent * EAgent * Capability * Mode *
                                EAgent * Mission * Trace * Trace -> Boolean
op deauthorizes: ManagementAgent * KDC * Capability * Mode *
                                EAgent * Mission * Trace * Trace -> Boolean
op delegates: ManagementAgent * KDC * KDC * Capability * Mode *
                                EAgent * Mission * Trace * Trace -> Boolean
op revokesDelegation: ManagementAgent * KDC * KDC * Capability *
                    Mode * EAgent * Mission * Trace * Trace -> Boolean
```

KMI Theory: Agent Actions

Example Axiom:

```
∀(agent1,agent2,mission,t,t1)
        establishesSessionKey(agent1,agent2,mission,t,t1) →
            extends(t,t1) ∧
            ∃(t0,s)
                 t1 = cons(s,t0) ∧
                 (hasLongTermKey(agent1,mission,t0) ∧
                  hasLongTermKey(agent2,mission,t0) ∨
                  (∃(kmMission)
                          kmMission?(kmMssion) ∧
                          kdc?(agent1) ∧
                          hasSessionKey(agent1,kmMission,t0) ∧
                          hasSessionKey(agent2,kmMission,t0)))
                 authorized(agent1,sessionKey,establish,agent2,mission,t0)
```

Secrecy and Authentication

- Secrecy properties:
  - *Key* parameter exposed in *Mission* parameter
  - ∀(key) *action*(…,mission,…) & key?(key,mission) → ¬knows(spy,key)
- Authentication properties:
  - Existing axioms constrain all actions and capabilities to be anchored in an initial authorization
  - Add: ∀(x,y,..) ¬authorizes(spy,x,y,…)
  - Spoofable implementations will fail refinement proof

Dolev-Yao Spy

- Change communicatingAgents theory:
  - Every message sent is received first by spy
  - Every message received is sent by spy
  - Spy may alter message
  - Spy may fail to forward message
- Red fill refinements require: $\forall(x)$ initialKey(x) $\rightarrow$ $\neg$knows(spy,x) as axiom
- Benign fill refinements don't
- Can't avoid: $\forall(x,y,..)$ $\neg$authorizes(spy,x,y,…)

Some Observations

- Alternate ontologies/axioms facilitate different proofs
- Often inarticulate about what we would like to prove
- Deriving atomic instances can stimulate thinking about desirable conjectures