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

Distribution

Authority

KMI

Unclassified//FOUO
Management

Authorization
Ordering
Generation
Protection
Changing
Destroying

KMI

Authority

Unclassified//FOUO
Modernization: Red Fill

Authority

Plain Text

Unclassified//FOUO
Modernization: Black Fill

Authority → Cipher Text

Unclassified//FOUO
Modernization: Benign Fill

KMI → Cipher Text → ECU

ECU → ECU → ECU → ECU

Unclassified//FOUO
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

Unclassified//FOUO
Authorization/Capability Model

Authenticate KM Authority

Establish KM Transactions

Secure KM Transactions

Establish ECU/ECU Transactions

Secure ECU/ECU Transactions

Unclassified//FOUO
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

Unclassified//FOUO
Informal: OO Design

Class

capability

Object

ECU

Properties

Authentication
Establish
Secure Trans

Operations

authorization

authorization

authorization

Managed Object Architecture
Abstract Protocols Architecture

Formal: 1st Order Predicates

authorization

authorization(\text{agent1, agent2, …})

authorization

capability(\text{agent})

key establishment

challenge/response

key agreement

send \rightarrow receive

send \rightarrow receive

send

send
Specware

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

- KMI Theory
  - User Requirements
    - Protocol Specifications
      - State Machine Software
Two Specware Theories

- KMI Theory
  - User Requirements
    - Protocol Specifications
      - State Machine Software
  - Communicating Agents Theory
\texttt{\textbf{trans}(agent, stateIn, messageReceived, stateOut, messageSent)}
sort AgentSlice = Agent * FSeq(Message) * State * Option(Message)

sort TraceSlice = FSet(AgentSlice)
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

Unclassified//FOUO
“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 capabilities
- 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

Authenticate KM Authority

Establish KM Authority

Key: trustAnchor
Mission: KM

Establish ECU/ECU Transactions

Secure KM Transactions

Key: longTerm
Mission: KM

Secure ECU/ECU Transactions

Key: session
Mission: nonKM

Establish ECU/ECU Transactions

Secure KM Transactions

Key: session
Mission: nonKM

Key: longTerm
Mission: nonKM

Unclassified//FOUO
KMI Theory: ECU Capabilities

Predicates:

\[
\text{op authenticatedCapability: ECU * KMIAgent * Mission * Trace -> Boolean}
\]
\[
\text{op estKmTransCapability: ECU * Mission * Trace -> Boolean}
\]
\[
\text{op kmTransCapability: ECU * Mission * Trace -> Boolean}
\]
\[
\text{op estMissionCapability: ECU * Mission * Trace -> Boolean}
\]
\[
\text{op missionCapability: ECU * Mission * Trace -> Boolean}
\]

Example Axiom:

\[
\forall (\text{ecu}, \text{mission}, t) \quad \text{missionCapability} (\text{ecu}, \text{mission}, t) \equiv
\]
\[
\quad \text{hasSessionKey} (\text{ecu}, \text{mission}, t) \land
\]
\[
\quad \neg \text{kmMission?} (\text{mission})
\]

Unclassified//FOUO
Predicates:

\[
\begin{align*}
&\text{op hasSessionKey: } \text{EAgent} \times \text{Mission} \times \text{Trace} \rightarrow \text{Boolean} \\
&\text{op hasLongTermKey: } \text{EAgent} \times \text{Mission} \times \text{Trace} \rightarrow \text{Boolean} \\
&\text{op hasTrustAnchor: } \text{EAgent} \times \text{KMIAgent} \times \text{Mission} \times \text{Trace} \rightarrow \text{Boolean} \\
&\text{op authorized: } \text{KDC} \times \text{Capability} \times \text{Mode} \times \text{EAgent} \times \text{Mission} \times \text{Trace} \rightarrow \text{Boolean} \\
&\text{op delegated: } \text{KDC} \times \text{KDC} \times \text{Capability} \times \text{Mode} \times \text{EAgent} \times \text{Mission} \times \text{Trace} \rightarrow \text{Boolean}
\end{align*}
\]
Example Axiom:

\[
\forall (agent_1, mission, t) \\
\text{hasSessionKey}(agent_1, mission, t) \equiv \\
\exists (agent_2, t_1, t_0) \\
\text{extends}(t, t_1) \land \\
\text{establishesSessionKey}(agent_2, agent_1, mission, t_1, t_0) \land \\
\neg \exists (t_2, key) \\
\text{extends}(t, t_2) \land \\
\text{extends}(t_2, t_1) \land \\
\text{sessionKey?}(mission, key, t_1) \land \\
\text{expires}(key, t_2)
\]
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

Unclassified//FOUO
KMI Theory: Agent Actions

Example Axiom:

\[ \forall (agent1, agent2, mission, t, t1) \]

\[ \text{establishesSessionKey}(agent1, agent2, mission, t, t1) \rightarrow \]

\[ \text{extends}(t, t1) \land \]

\[ \exists (t0, s) \]

\[ t1 = \text{cons}(s, t0) \land \]

\[ (\text{hasLongTermKey}(agent1, mission, t0) \land \]

\[ \text{hasLongTermKey}(agent2, mission, t0) \lor \]

\[ (\exists (kmMission) \]

\[ \text{kmMission}? (kmMission) \land \]

\[ \text{kdc?}(agent1) \land \]

\[ \text{hasSessionKey}(agent1, kmMission, t0) \land \]

\[ \text{hasSessionKey}(agent2, kmMission, t0)) \]

\[ \text{authorized}(agent1, sessionKey, establish, agent2, mission, t0) \]

Unclassified//FOUO
Secrecy and Authentication

- **Secrecy properties:**
  - *Key* parameter exposed in *Mission* parameter
  - $\forall (key) \ action(..., mission,...) \ & \ key?(key, mission) \ \rightarrow \ \neg knows(spy, key)$

- **Authentication properties:**
  - Existing axioms constrain all actions and capabilities to be anchored in an initial authorization
  - Add: $\forall (x, y, ..) \ \neg authorizes(spy, x, y, ..)$
  - Spoofable implementations will fail refinement proof

Unclassified//FOUO
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