An Operational Semantics for Trust Policies

Karl Krukow

Department of Computer Science
University of Aarhus, Denmark

A Familiar Picture

$L$
A Familiar Picture

\[
\text{Den.} \quad \vdash \quad L
\]

\[\llbracket \cdot \rrbracket_{\text{den}}\]
A Familiar Picture
A Familiar Picture

Diagram showing:
- Den.
- L

Edges:
- $[\cdot]_{\text{den}}$ from Den. to L
- $[\cdot]_{\text{op}}$ from L to Op.

($\text{den}$ and $\text{op}$ denote operational semantics for different policies.)
The domain of this work is *trust management* (TM) systems.

- Supports security decision-making in open, large-scale distributed applications.

**Key Concept:**

- Security Policy.
- A corresponding notion of compliance with security policy.

Foundations of TM has been the subject of sophisticated theoretical work.

- e.g., Mitchell; Li; Feigenbaum; Weeks; Shmatikov & Talcott; Carbone, Nielsen & Sassone; others . . .
Our work extends the trust-structure framework: a general semantic model for trust.


In line with the original ideas of TM.

- Based on distributed ‘trust-policies.’
- Flexible.
- Separates mechanism from policy (in fact, there is no mechanism at all!).

This last point is exactly the subject of this work.

- Complement the framework’s denotational semantics with a formal operational semantics.
Our work extends the trust-structure framework: a general semantic model for trust.


In line with the original ideas of TM.

- Based on distributed ‘trust-policies.’
- Flexible.
- Separates mechanism from policy (in fact, *there is no mechanism at all!*).

This last point is exactly the subject of this work.

- Complement the framework’s denotational semantics with a formal operational semantics.
Outline

1 Trust Structures
   - Introduction
   - The Formal Model
   - Motivation: The Operational Problem

2 Our Contribution
   - A Distributed Algorithm for Computing Least Fixed-Points
   - Formalization using I/O Automata
   - Correspondence result
Outline

1 Trust Structures
- Introduction
- The Formal Model
- Motivation: The Operational Problem

2 Our Contribution
- A Distributed Algorithm for Computing Least Fixed-Points
- Formalization using I/O Automata
- Correspondence result
Trust Structures: An Introduction

- Provides a generic mathematical framework, formalizing and solving the following problem.
  - Given a set \( \mathcal{P} \) of principal identities, each specifying a local trust-policy, define a unique **global trust-state** compatible with those policies.

- **A global trust-state?**
  - formally, a *global trust state* is a function \( \text{gts} : \mathcal{P} \to \mathcal{P} \to D = \text{GTS} \) for some set \( D \) of “degrees of trust” (called trust values).
  - for each \( p, q \in \mathcal{P} \) answer: “to what degree does \( p \) trust \( q \)?” as \( \text{gts}(p)(q) \).

- **A generic model?**
  - Different applications have different requirements for trust-information.
  - Obtained by choosing set \( D \) of trust degrees, and by choosing appropriate trust policies.
  - Example instances, KeyNote, SPKI, SECURE Trust model, . . .
Trust Structures: An Introduction

- Provides a generic mathematical framework, formalizing and solving the following problem.
  - Given a set $\mathcal{P}$ of principal identities, each specifying a local trust-policy, define a unique *global trust-state* compatible with those policies.

- A *global trust-state*?
  - formally, a *global trust state* is a function $\text{gts} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D = \text{GTS}$ for some set $D$ of “degrees of trust” (called trust values).
  - for each $p, q \in \mathcal{P}$ answer: “to what degree does $p$ trust $q$?” as $\text{gts}(p)(q)$.

- Trust policies?
  - Specifies how a principal defines its trust in others.
  - $\pi_p$ : “my trust in Alice is high and for anyone else, it is the minimum of what Bob and Carl think.”
Trust Structures: Trust Policies

A Simple Language

\[
\pi ::= \star : \tau \\
\quad | \ p : \tau, \pi
\]

\[
\tau ::= \ d \\
\quad | \ p?q \\
\quad | \ op^i_n(\tau_1, \tau_2, \ldots, \tau_n)
\]

E.g., the policy \(\pi_p\) of principal \(p\) could be

\[
A : \text{high}, \star : (B?\star) \land (C?\star)
\]
Trust Structures: Trust Policies

A Simple Language

\[
\begin{align*}
\pi ::= & \quad \star : \tau \\
& | \quad p : \tau, \pi \\
\tau ::= & \quad d \\
& | \quad p?q \\
& | \quad \text{op}_n(\tau_1, \tau_2, \ldots, \tau_n)
\end{align*}
\]

E.g., the policy \(\pi_p\) of principal \(p\) could be

\[A : \text{high}, \star : (B?\star) \land (C?\star)\]

Denotationally, simply functions mapping \textit{global} trust-states to \textit{local} trust states, \([\pi_p]^{\text{den}} : \text{GTS} \rightarrow \text{LTS}\), i.e., \([\pi_p]^{\text{den}} : (\mathcal{P} \rightarrow \mathcal{P} \rightarrow \text{D}) \rightarrow \mathcal{P} \rightarrow \text{D}\).
Trust Structures: The Formal Model (1/3)

- Refining the main objective of the model.
  - Fix a set $\mathcal{P}$ of principal identities.
  - Each principal $p \in \mathcal{P}$ specifying a local trust-policy $\pi_p$.
    Let $\Pi = (\pi_p \mid p \in \mathcal{P})$ be the (global) collection of all these policies,
    $\llbracket \pi_p \rrbracket^{\text{den}} : \text{GTS} \rightarrow \text{LTS}$.
  - Goal is to define a suitable unique global trust-state, denoted $\llbracket \Pi \rrbracket^{\text{den}}$, compatible with all the policies $\Pi$.

- But... policies may have cyclic references.
  - Example: $\pi_p = \ast : q?\ast$, $\pi_q = \ast : p?\ast$.
  - In this example, the policies contain no information: this is distinct from explicitly specifying untrusted.

- This example suggests that the set $D$ of trust-values can be ordered in two fundamentally distinct ways: with respect to
  - trust / privilege ($\preceq$), e.g. low $\preceq$ high.
  - information content ($\sqsubseteq$), e.g. unknown $\sqsubseteq$ low.
Trust Structures: The Formal Model (1/3)

- Refining the main objective of the model.
  - Fix a set $\mathcal{P}$ of principal identities.
  - Each principal $p \in \mathcal{P}$ specifying a local trust-policy $\pi_p$.
    Let $\Pi = (\pi_p | p \in \mathcal{P})$ be the (global) collection of all these policies, $\llbracket \pi_p \rrbracket^{\text{den}} : \text{GTS} \rightarrow \text{LTS}$.
  - Goal is to define a suitable unique global trust-state, denoted $\llbracket \Pi \rrbracket^{\text{den}}$, compatible with all the policies $\Pi$.

- But... policies may have cyclic references.
  - Example: $\pi_p = \star : q?\star$, $\pi_q = \star : p?\star$.
  - In this example, the policies contain no information: this is distinct from explicitly specifying untrusted.

- This example suggests that the set $D$ of trust-values can be ordered in two fundamentally distinct ways: with respect to
  - trust/privilege ($\preceq$), e.g. low $\preceq$ high.
  - information content ($\sqsubseteq$), e.g. unknown $\sqsubseteq$ low.
Trust Structures: The Formal Model (1/3)

- Refining the main objective of the model.
  - Fix a set $\mathcal{P}$ of principal identities.
  - Each principal $p \in \mathcal{P}$ specifying a local trust-policy $\pi_p$.
    Let $\Pi = (\pi_p \mid p \in \mathcal{P})$ be the (global) collection of all these policies, $[\pi_p]^{\text{den}} : \text{GTS} \to \text{LTS}$.
  - Goal is to define a suitable unique global trust-state, denoted $[\Pi]^{\text{den}}$, compatible with all the policies $\Pi$.

- But... policies may have cyclic references.
  - Example: $\pi_p = \star : q?\star$, $\pi_q = \star : p?\star$.
  - In this example, the policies contain no information: this is distinct from explicitly specifying untrusted.

- This example suggests that the set $D$ of trust-values can be ordered in two fundamentally distinct ways: with respect to
  - trust / privilege ($\preceq$), e.g. low $\preceq$ high.
  - information content ($\sqsubseteq$), e.g. unknown $\sqsubseteq$ low.
**Definition [Trust Structure]**

A *trust structure* is a triple \( T = (D, \preceq, \sqsubseteq) \), consisting of a set \( D \) and two orderings on \( D \), called the *trust ordering* (\( \preceq \)) and the *information ordering* (\( \sqsubseteq \)). The trust ordering preorders \( D \) and the information order makes \( (D, \sqsubseteq) \) a complete partial order.

Example \( D = \{ \text{low}, \text{mid}, \text{high}, \text{unknown}, \text{midORhigh} \} \).

<table>
<thead>
<tr>
<th>Trust Order, ( \preceq )</th>
<th>Information Order, ( \sqsubseteq )</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>midORhigh</td>
<td>midORhigh</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
</tr>
<tr>
<td>mid</td>
<td>unknown</td>
</tr>
<tr>
<td>low</td>
<td>low</td>
</tr>
</tbody>
</table>
A trust structure is a triple $T = (D, \preceq, \sqsubseteq)$, consisting of a set $D$ and two orderings on $D$, called the trust ordering ($\preceq$) and the information ordering ($\sqsubseteq$). The trust ordering preorders $D$ and the information order makes $(D, \sqsubseteq)$ a complete partial order.

Example: $D = \{\text{low, mid, high, unknown, midORhigh}\}$.
Let $T = (D, \preceq, \sqsubseteq)$ be a trust structure, and $\Pi = (\pi_p \mid p \in \mathcal{P})$ be a collection of trust policies.

A fixed point of $\Pi$ is a global trust-state $gts$ so that for all $p \in \mathcal{P}$ we have $\pi_p(gts) = gts(p)$.

- Any fixed point of $\Pi$ is consistent with all policies.

Assuming that all policies are continuous with respect to $\sqsubseteq$.

**Definition $[[\Pi]]^{\text{den}}$**

Define the global trust-state of $\Pi$ in $T$ as the $\sqsubseteq$-least fixed-point of $\Pi$.

$$[[\Pi]]^{\text{den}} \overset{\text{def}}{=} \text{lfp}_{\sqsubseteq} \Pi$$
Let $T = (D, \preceq, \sqsubseteq)$ be a trust structure, and $\Pi = (\pi_p \mid p \in \mathcal{P})$ be a collection of trust policies.

A fixed point of $\Pi$ is a global trust-state $\text{gts}$ so that for all $p \in \mathcal{P}$ we have $\pi_p(\text{gts}) = \text{gts}(p)$.

▶ Any fixed point of $\Pi$ is consistent with all policies.

Assuming that all policies are continuous with respect to $\sqsubseteq$.

**Definition $[\mathcal{P}]^{\text{den}}$**

Define the global trust-state of $\Pi$ in $T$ as the $\sqsubseteq$-least fixed-point of $\Pi$.

$$[[\Pi]]^{\text{den}} \overset{(\text{def})}{=} \text{lfp}_{\sqsubseteq} \Pi$$
The Operational Problem (1/2)

- Suppose principal $p$ has to make a *trust-based* access-control decision about another principal $q$.

- A *trust-based security policy* is a function of the trust in the requestor.
  - $\sigma : D \rightarrow \{\top, \bot\}$
  - A monotonic security policy satisfies $d \preceq d' \Rightarrow \sigma(d) \Rightarrow \sigma(d')$.
  - E.g., threshold policies: Allow access to $q$ if my trust in $q$ is trust-wise above threshold $t \in D$. (e.g., $t = \text{high}$)

- So one *mechanism* for deciding a request could be
  - compute $x := \llbracket \Pi \rrbracket^{\text{den}}(p)(q)$
  - feed this value to policy, $\sigma(x) = \top$?
The Operational Problem (2/2)

- A denotational model:
  - $\llbracket \Pi \rrbracket^{\text{den}}$ is a well defined, unique mathematical object.
- But there is “no recipe for getting there”?
  - One might say there is no mechanism for evaluating policies.

- How do we actually compute the trust values, when
  - $\Pi$ is distributed.
  - $|\mathcal{P}|$ is large.
  - no centralized authority.
  - policy updates . . .

- The standard technique for fixed point computation:
  - $\bot \subseteq \Pi(\bot) \subseteq \Pi^2(\bot) \cdots$?
  - Inadequate!
The Operational Problem (2/2)

- A denotational model:
  - $\llbracket \Pi \rrbracket^{\text{den}}$ is a well defined, unique mathematical object.
- But there is “no recipe for getting there”?
  - One might say there is no mechanism for evaluating policies.

- How do we actually compute the trust values, when
  - $\Pi$ is distributed.
  - $|\mathcal{P}|$ is large.
  - no centralized authority.
  - policy updates . . .

- The standard technique for fixed point computation:
  \[ \bot \subseteq \Pi(\bot) \subseteq \Pi^2(\bot) \cdots ? \]
  - Inadequate!
Outline

1 Trust Structures
   - Introduction
   - The Formal Model
   - Motivation: The Operational Problem

2 Our Contribution
   - A Distributed Algorithm for Computing Least Fixed-Points
   - Formalization using I/O Automata
   - Correspondence result
Overview: operational techniques

- An asynchronous distributed algorithm for computing $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$ for arbitrary but fixed $p, q \in \mathcal{P}$.
  - Only nodes necessary for computing $p$’s trust in $q$ are involved.

- Approximation Algorithms.
  - ‘Proof-carrying’ requests (client presents proof to speed-up access decision).
  - Snapshot-based approximation.
  - Generalized Protocol (combination of both).

- Techniques described previously (no rigorous formalization) (Krukow and Twigg, 2005).

- Our contribution here is a formalization and proof of correctness of the asynchronous algorithm, using I/O Automata.

- See the full paper (Krukow and Nielsen, 2006) for the complete formalization.
Overview: operational techniques

- An asynchronous distributed algorithm for computing $\prod_{\text{den}}(p)(q)$ for arbitrary but fixed $p, q \in \mathcal{P}$.
  - Only nodes necessary for computing $p$’s trust in $q$ are involved.

- Approximation Algorithms.
  - ‘Proof-carrying’ requests (client presents proof to speed-up access decision).
  - Snapshot-based approximation.
  - Generalized Protocol (combination of both).

- Techniques described previously (no rigorous formalization) (Krukow and Twigg, 2005).

- Our contribution here is a formalization and proof of correctness of the asynchronous algorithm, using I/O Automata.

- See the full paper (Krukow and Nielsen, 2006) for the complete formalization.
\[ \pi = A : \text{high}, B? \star \land C? \star \]

\[ L_\pi \]
Overview

\[ \pi = A: \text{high}, \star : B? \star \land C?\star \]

World of Mogens: Ideas

\[ f : (\mathcal{P} \to \mathcal{P} \to D, \sqsubseteq) \to (\mathcal{P} \to D, \sqsubseteq) \]

\[ \bot_{\sqsubseteq} \]
Overview

\[ \pi = A : \text{high}, \star : B? \star \land C?\star \]

World of Mogens: Ideas

\[ f : (\mathcal{P} \to \mathcal{P} \to D, \sqsubseteq) \to (\mathcal{P} \to D, \sqsubseteq) \]

World of Karl: Pragmatics

Operational Semantics for Trust Policies

WITS'06, Vienna, Austria
Overview

\[ \pi = A: \text{high}, \star: B? \star \land C?\star \]

World of Mogens: Ideas

World of Karl: Pragmatics

\[ f : (\mathcal{P} \rightarrow \mathcal{P} \rightarrow D, \sqsubseteq) \rightarrow (\mathcal{P} \rightarrow D, \sqsubseteq) \]
A Distributed Asynchronous Algorithm

Scenario.
- Principal $p$ needs to compute its trust in $q$, i.e., the value $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$.
- Principals are network nodes with computational and communication capacity.
  - Asynchronous, but reliable network.

Basic observation.
- Compute the local value $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$ directly, rather than computing the global state $\llbracket \Pi \rrbracket^{\text{den}}$ and then looking up “entry” $(p, q)$.
- Excludes principals that are not relevant for the specific computation.

Algorithm: two-step computation.
- Dependency analysis distributedly computes a sub-graph $G_{(p,q)}$ of the dependency graph $G$ for the policies $\Pi$.
- Asynchronous fixed-point algorithm in dependency graph.
A Least-Fixed-Point Algorithm (1/2)

- Very simple algorithm; instance of a framework for asynchronous fixed-point algorithms (Bertsekas and Tsitsiklis, 1989).
- Tailored for local least fixed-points ($\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$).
- Essentially a distributed, asynchronous version of the synchronous iteration:
  \[
  \bot \subseteq \Pi(\bot) \subseteq \cdots \subseteq \Pi^{i}(\bot)
  \]
- Proved correct using the Asynchronous Convergence Theorem of Bertsekas; the following invariant is essentially maintained:
  - Node $p$, is able to compute a sequence of values
    \[
    \bot \subseteq t_0 \subseteq \cdots \subseteq t_n = \llbracket \Pi \rrbracket^{\text{den}}(p)(q)
    \]
    converging towards its local fixed point value.
  - Similar convergence holds at other nodes.
A Least-Fixed-Point Algorithm (2/2)

$\text{recv}(p, s, A, d)$

$\text{send}(p, q, A, e)$

$\text{gts} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D$

$\llbracket \pi_p \rrbracket^{\text{op}}$
A Least-Fixed-Point Algorithm (2/2)

\[ \text{recv}(p, s, A, d) \mapsto [\pi_p]^{\text{op}} \]

\[ \text{send}(p, q, A, e) \mapsto [\pi_A]^{\text{op}}, [\pi_B]^{\text{op}}, [\pi_C]^{\text{op}}, [\pi_D]^{\text{op}} \]

\[ \text{gts} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D \]
A Least-Fixed-Point Algorithm (2/2)

\[ \text{gts} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D \]

\[ \text{recv}(p, s, A, d) \]

\[ \text{send}(p, q, A, e) \]

\[ [\pi_p]^\text{op} \]

\[ [\pi_A]^\text{op} \]

\[ [\pi_B]^\text{op} \]

\[ [\pi_C]^\text{op} \]

\[ [\pi_D]^\text{op} \]
A Least-Fixed-Point Algorithm (2/2)

\[ e = \pi_{pq}(gts) \]

\[ gts : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D \]

recv\((p, s, A, d)\)

send\((p, q, A, e)\)

\([\pi_p]^{op}\)

\([\pi_A]^{op}\)

\([\pi_B]^{op}\)

\([\pi_C]^{op}\)

\([\pi_D]^{op}\)
A Least-Fixed-Point Algorithm (2/2)

\[ gts : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D \]

\[ \text{recv}(p, s, A, d) \]

\[ [\pi_p]^{\text{op}} \]

\[ \ldots \]

\[ \text{send}(p, q, A, e) \]

\[ [\pi_A]^{\text{op}} \]

\[ [\pi_B]^{\text{op}} \]

\[ [\pi_C]^{\text{op}} \]

\[ [\pi_D]^{\text{op}} \]
Each policy $\pi_p$ of each principal $p \in \mathcal{P}$ is mapped to an I/O automaton, modelling a node in the algorithm.

A web of policies, $\Pi = (\pi_p \mid p \in \mathcal{P})$ is mapped to a composition of its parts, $\llbracket \pi_p \rrbracket^{op}$. 
Formalization

Each policy $\pi_p$ of each principal $p \in \mathcal{P}$ is mapped to an I/O automaton, modelling a node in the algorithm.

\[
\begin{align*}
\text{recv}(p, s, q, d) & \quad \text{eval}(p, q), q \in \mathcal{P} & \quad \text{send}(p, r, q, d) \\
 s, q \in \mathcal{P}, d \in D & & r, q \in \mathcal{P}, d \in D
\end{align*}
\]

A web of policies, $\Pi = (\pi_p \mid p \in \mathcal{P})$ is mapped to a composition of its parts, $\llbracket \pi_p \rrbracket^{\text{op}}$. 
Formalization: Composition

Example: two principals

For a collection, say, $\Pi = [A \mapsto \pi_A, B \mapsto \pi_B]$.
Define $[\Pi]^{op}$ as a composition of the policy-automata, e.g., $[\pi_A]^{op}$, and channel automata e.g., $\text{Channel}(A, B)$:

$$\begin{align*}
\text{send}(A, B, q, d) & \rightarrow \begin{array}{c} d_0 d_1 \cdots d_n \\ \text{recv}(B, A, q, d) \end{array} \\
\text{recv}(A, B, q, d) & \rightarrow \begin{array}{c} \text{send}(B, A, q, d) \\
\text{eval}(A, q) & \rightarrow \text{eval}(B, q) \\
\text{Channel}(A, B) & \rightarrow \text{Channel}(B, A) \\
\end{array}
\end{align*}$$

$[\Pi]^{op}$ is the following composition:
Formalization: Composition

Example: two principals

For a collection, say, $\Pi = [A \mapsto \pi_A, B \mapsto \pi_B]$. Define $[\Pi]^{op}$ as a composition of the policy-automata, e.g., $[\pi_A]^{op}$, and channel automata e.g., $\text{Channel}(A, B)$:

$[\Pi]^{op}$ is the following composition:
Correspondence of Semantics

Proof Structure

\[ L \]

\[ \text{Den.} \rightarrow \text{L} \rightarrow \text{Op.} \]

\[ \llbracket \cdot \rrbracket_{\text{den}} \]

\[ \llbracket \cdot \rrbracket_{\text{op}} \]

\[ \geq \]

Karl Krukow (University of Aarhus)

Operational Semantics for Trust Policies

WITS'06, Vienna, Austria
Correspondence of Semantics

Proof Structure

Diagram:

- L
- Den.

Relations:

- \([\cdot]\)_{\text{den}}
- \([\cdot]\)_{\text{op}}
- \([\cdot]\)_{\text{op-abs}}
Correspondence of Semantics
Proof Structure

\[ L \]

Den. \[ \llbracket \cdot \rrbracket_{\text{den}} \]

\[ \llbracket \cdot \rrbracket_{\text{op}} \]

Abs. Op. \[ \llbracket \cdot \rrbracket_{\text{op-abs}} \]

Op. \[ \lor \]
Summary

- The trust-structure framework is a denotational semantic model for trust-management systems.
- We have provided a formal operational semantics for a general language of policies.
- Operational Semantics is proven to converge to the denotational semantics.
For Further Reading


