

# Dynamic Trust Management

*Based on Progress Report “On foundations for dynamic trust management”, available*

<http://www.brics.dk/~krukow>

Karl Krukow

`krukow@brics.dk`

BRICS, University of Aarhus, Denmark

- Essence of SECURE:
  - Explore to which extent (intuition about) the *human notion of trust* can guide *security-related decision-making* for computational entities in *global-computing environments*.

# SECURE Project

- Essence of SECURE:
  - Explore to which extent (intuition about) the *human notion of trust* can guide *security-related decision-making* for computational entities in *global-computing environments*.
- **Security-related decision-making.**
  - Passive decisions.
    - e.g. “should I allow  $p$  to access my resource  $res$ ?”.
  - Active decisions.
    - e.g. “which of  $p$ ,  $q$  and  $r$  is most likely to provide the best service for me?”.

# SECURE Project

- Essence of SECURE:
  - Explore to which extent (intuition about) the *human notion of trust* can guide *security-related decision-making* for computational entities in *global-computing environments*.
- Properties of **global-computing environments**
  - Vast numbers of interacting entities.
    - impossible to have *complete information* about every potential collaborator.
  - Entities are mobile and networked, but decisions are made *autonomously*.

# SECURE Project

- Essence of SECURE:
  - Explore to which extent (intuition about) the *human notion of trust* can guide *security-related decision-making* for computational entities in *global-computing environments*.
- Intuition about **human notion of trust**.
  - Locality: trust exist *between principals*.
    - e.g.  $p$ 's trust in  $q$  may be different from  $r$ 's trust in  $q$ .
  - Dynamics: reflects *behavior*.
  - Contextual.

# Overview of the rest of the talk

- The SECURE trust model.
- Topics not covered by this talk:
  - Transfer of information between contexts.
  - An abstract denotational framework for trust.
  - Operational aspects of the denotational models.
  - A canonical construction: intervals.
- The future?

# SECURE project: Decision making

- Model: a decision involving entity  $p$  has a number of possible *outcomes*,  $o_1, o_2, \dots, o_n$ .
- Each outcome  $o_i$  has an associated cost or benefit, say  $\text{cost}(o_i)$ .
- Trust values must convey enough information, that estimation of probabilities of outcomes be possible, e.g.

$$\text{expected-cost} = \sum_{i=1}^n \text{cost}(o_i) \cdot \text{likelihood}(o_i)$$

# SECURE Trust Model

- A trust model: mathematical framework that specifies a global trust state:  $gts : \mathcal{P} \rightarrow \mathcal{P} \rightarrow T$ .
- $T$ ? - Use the trust-structure framework,  $TS = (T, \sqsubseteq, \preceq)$ ?
- However, an arbitrary complete lattice is too abstract:
  - How does one estimate probabilities of outcomes?
  - How does one update trust information based on behaviour?
  - Must formalise: *outcomes, behaviour*.

# SECURE Trust Model

- A trust model: mathematical framework that specifies a global trust state:  $gts : \mathcal{P} \rightarrow \mathcal{P} \rightarrow T$ .
- $T$ ? - Use the trust-structure framework,  $TS = (T, \sqsubseteq, \preceq)$ ?
- However, an arbitrary complete lattice is too abstract:
  - How does one estimate probabilities of outcomes?
  - How does one update trust information based on behaviour?
  - Must formalise: *outcomes*, *behaviour*.
- Require additional structure...
  - $T = Outcomes \rightarrow EvidenceValues$
  - *Outcomes* and *EvidenceValues* also have structure...

# Example: E-Purse

- A scenario where entities store electronic cash in an electronic 'purse'.
- Entities can transfer money from one purse to another, e.g. to purchase services.
- Entities can request a transfer of 'real' money from their bank account to their e-purse.
- Scenario: User  $p$  wants to withdraw an amount,  $m$ , from its bank-account to its purse.
  - For this decision, what are the possible outcomes?

# Example: E-Purse – outcomes

- From the user's point of view, various events may occur:
  - Request may be *denied*:
    - Insufficient funds on account.
    - Server down.
    - Timeout.
    - . . .
  - Request may be *granted*, and  $m$  units are transferred.
    - Bank withdraws  $n \neq m$  from account.
    - Bank withdraws  $m$  from account.
    - Transferred cash is forged.
    - . . .

# Structure of outcomes

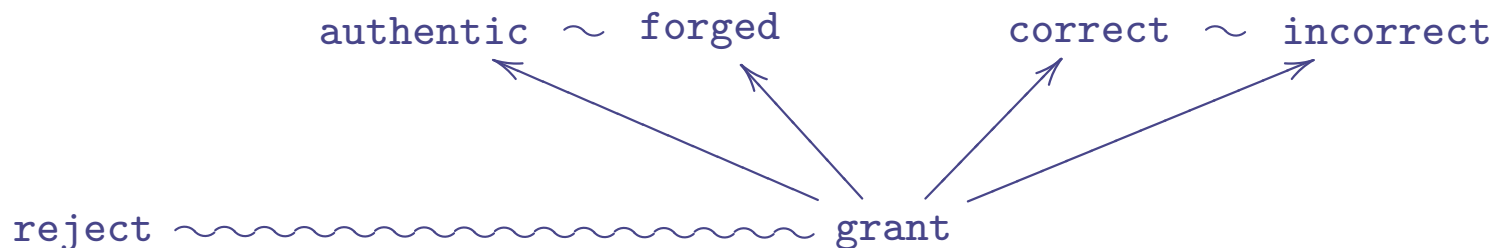
- An *outcome* can be described by a *set of observable events*.
- These events have structure.
  - Conflict: both cannot occur.
    - e.g. 'denied' vs. 'granted'.
  - Dependence: a pre-condition for an event to occur.
    - e.g. 'granted' before 'forged'.
  - Independence: none of the above.
    - e.g. 'forged' and 'correct amount withdrawn'.

# Modelling outcomes and behaviour 1/2

- Model: Event structures.

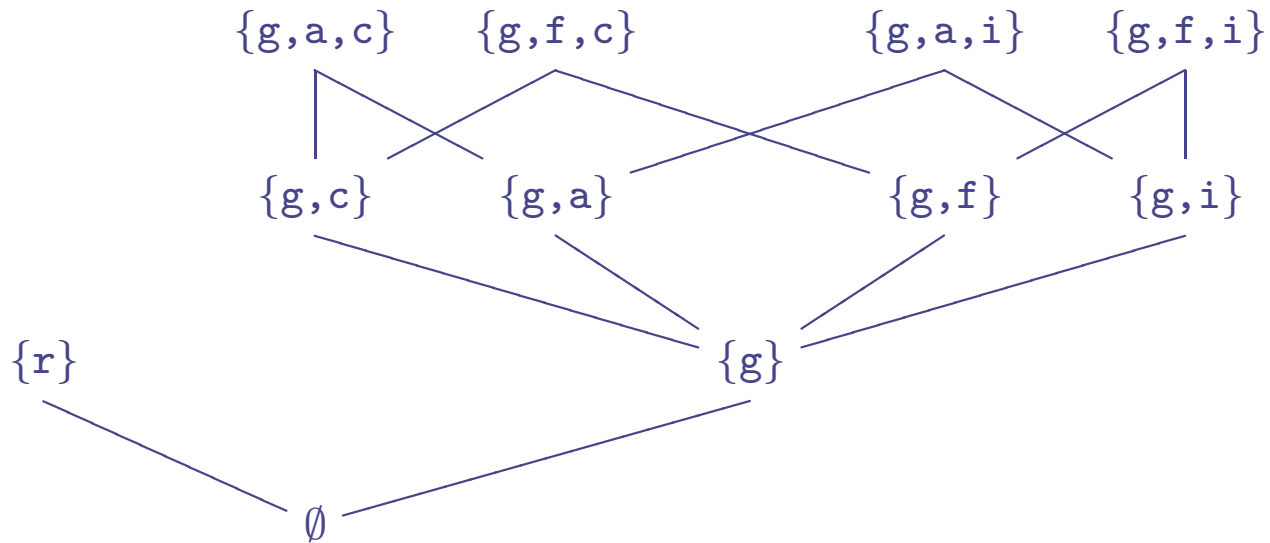
- $ES = (E, \leq, \#)$ .
- $E$  models the set of 'observable events'.
- $\leq \subseteq E \times E$ : dependency relation.
- $\# \subseteq E \times E$ : conflict relation.

- Example:



# Modelling outcomes and behaviour 2/2

- Model: Outcomes are configurations.
- Example:



- Model: Behaviour is a sequence of outcomes.

# Choosing trust values

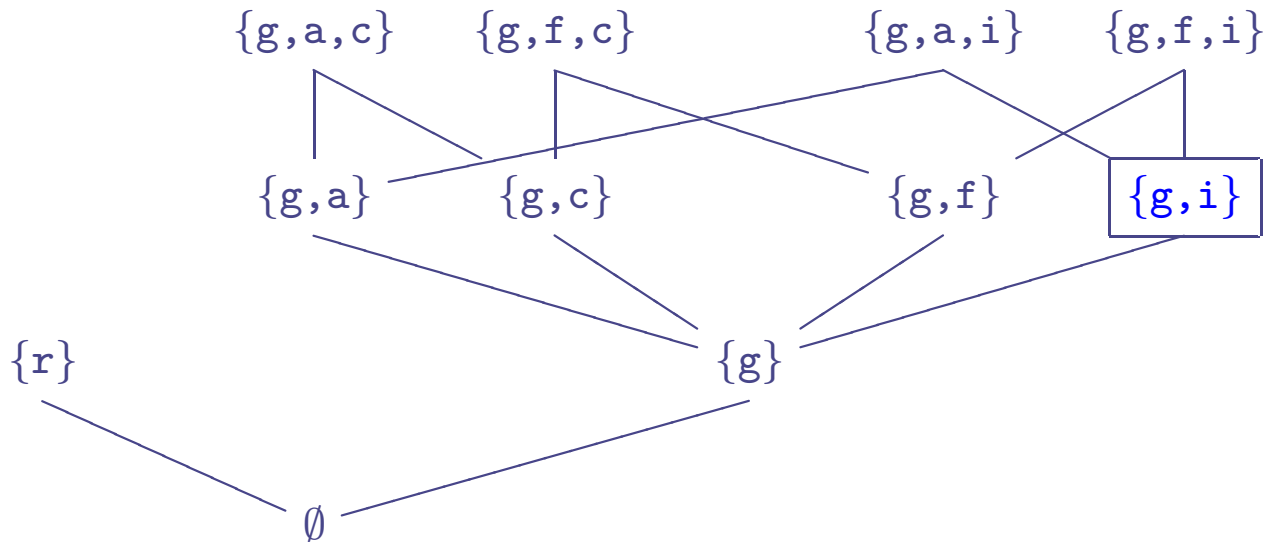
- Trust values: *Outcomes*  $\rightarrow$  *EvidenceValues*.

# Choosing trust values

- Trust values:  $C_{ES} \rightarrow EvidenceValues$ .

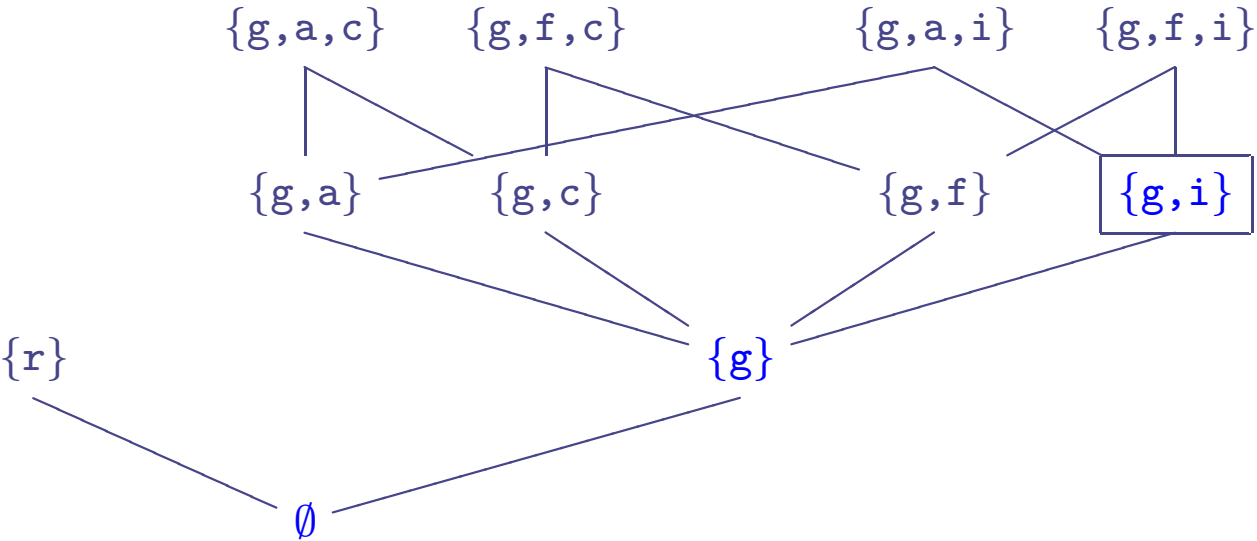
# Choosing trust values

- Trust values:  $\mathcal{C}_{ES} \rightarrow EvidenceValues$ .
- EvidenceValues?



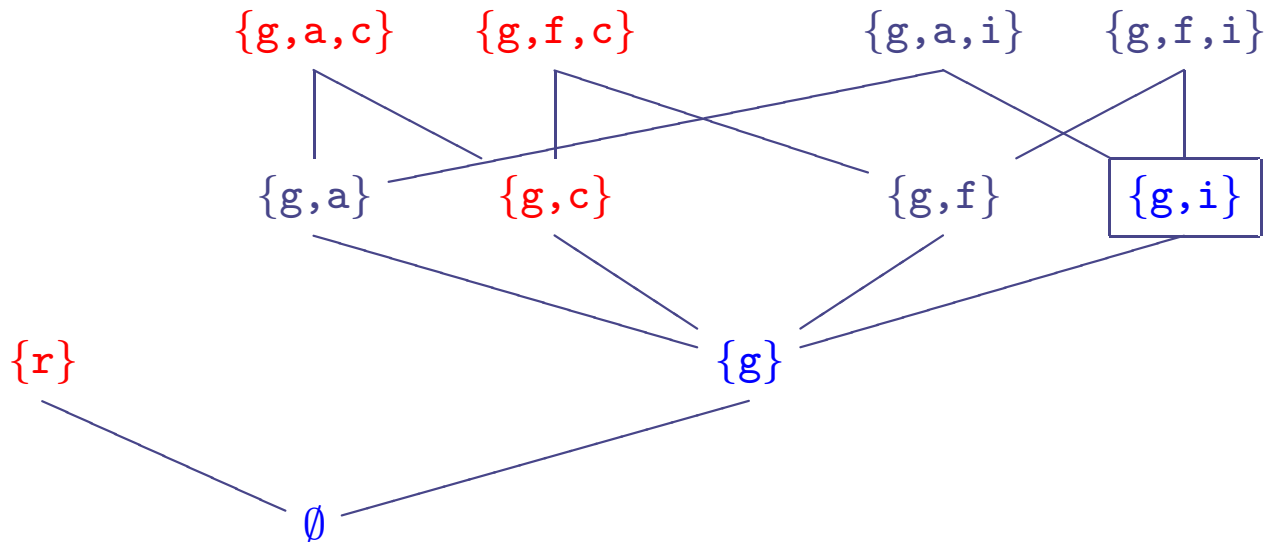
# Choosing trust values

- Trust values:  $\mathcal{C}_{ES} \rightarrow EvidenceValues$ .
- EvidenceValues?



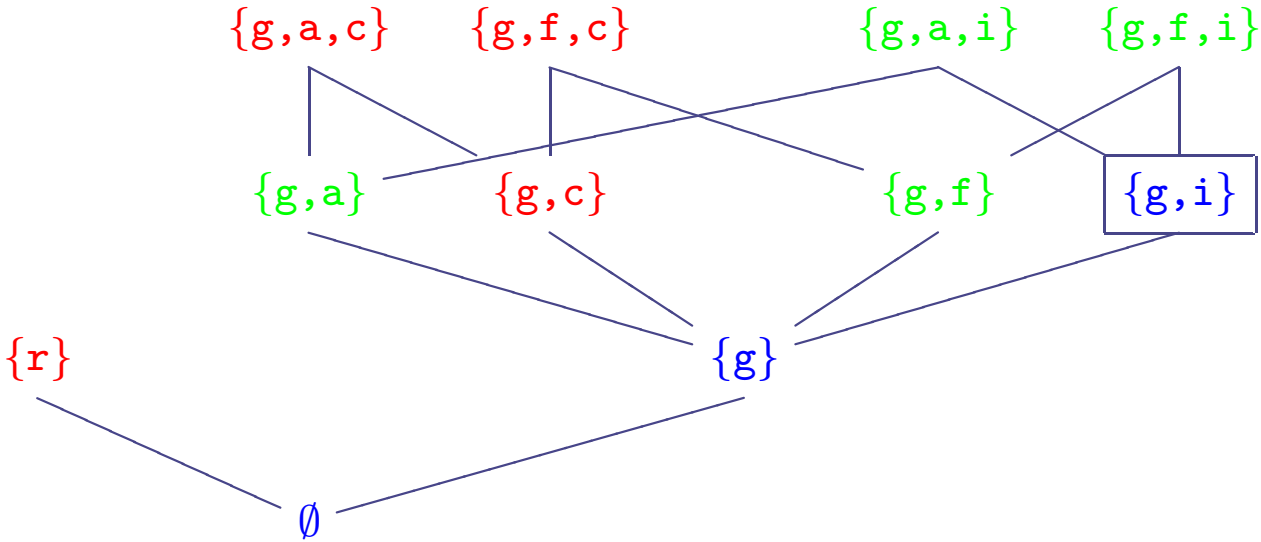
# Choosing trust values

- Trust values:  $\mathcal{C}_{ES} \rightarrow EvidenceValues$ .
- EvidenceValues?



# Choosing trust values

- Trust values:  $\mathcal{C}_{ES} \rightarrow EvidenceValues$ .
- EvidenceValues?



# SECURE Trust Model: Evidence values

- for any  $x \in \mathcal{C}_{ES}$  define the effect of  $x$  as a function  $\mathbf{eff}_x : \mathcal{C}_{ES} \rightarrow \mathbb{N}^3$ :

$$\mathbf{eff}_x(w) = \begin{cases} (1, 0, 0) & \text{if } w \subseteq x \\ (0, 0, 1) & \text{if } x \# w \text{ (i.e. } \exists e \in x, e' \in w : e \# e') \\ (0, 1, 0) & \text{otherwise} \end{cases}$$

- for a history  $b = x_1x_2 \cdots x_n$ , define  $\mathbf{eval} : \mathcal{C}_{ES}^0 \rightarrow (\mathcal{C}_{ES} \rightarrow \mathbb{N}^3)$  by

$$\mathbf{eval}(x_1x_2 \cdots x_n) = \lambda w. \sum_{i=1}^n \mathbf{eff}_{x_i}(w)$$

- $\mathbf{eval}(b) : \mathcal{C}_{ES} \rightarrow \mathbb{N}^3$  i.e.  $\mathbf{eval}(b)(w) = (s, i, c)$

# SECURE Trust Model

- Model:
  - Decision  $\sim$  event structure  $ES = (E, \leq, \#)$ .
  - (Partial) Outcomes  $\sim$  configurations  $\mathcal{C}_{ES}$ .
  - Behaviour  $\sim$  sequences of (finite) outcomes.
- We can derive from  $b \in \mathcal{C}_{ES}^0$ , an evidence value for each outcome,  $\text{eval}(b) : \mathcal{C}_{ES} \rightarrow \mathbb{N}^3$ .
- Trust values  $\sim t : \mathcal{C}_{ES} \rightarrow \mathbb{N}^3$ .

# SECURE Trust Model - Orderings

- We can order trust values  $T_0 = \mathcal{C}_{ES} \rightarrow \mathbb{N}^3$ .

- Define  $\sqsubseteq$  on  $\mathbb{N}^3$  by

$$(s, i, c) \sqsubseteq (s', i', c') \iff (s \leq s') \wedge (c \leq c') \wedge (s+i+c \leq s'+i'+c')$$

- Lift  $\sqsubseteq$  to  $T_0$  by ordering pointwise.

- $(T_0, \sqsubseteq)$  is a partial order, we complete this by adding a top element,  $\top_{\sqsubseteq}$  to  $\mathbb{N}^3$ , resulting in  $(\widehat{\mathbb{N}^3}, \sqsubseteq)$ , which is complete.

- We can define also a trust order  $\preceq$ .

# Road map

- The SECURE trust model.
- Topics not covered by this talk:
  - Transfer of information between contexts
  - An abstract denotational framework for trust.
  - Operational aspects of the denotational models.
  - A canonical construction: intervals.
- The future?

# Transfer of information

- Two contexts  $\sim$  two event structures  $ES_1, ES_2$ .
- Sometimes one has information about  $p \in \mathcal{P}$  regarding  $ES_1$  but not  $ES_2$ .
- Morphisms of event structures as information transfer functions.
  - $\eta : ES_1 \rightarrow ES_2$  is a “backwards” function  $\eta : E_2 \rightarrow 2^{E_1}$  + axioms.
  - $e_1 \in \eta(e_2)$  means that  $e_2$  occurs in  $E_2$  when  $e_1$  occurs in  $E_1$ .
  - ITFs and event structures form a category – compose and have identities.
- Useful?

# A mathematical framework - Trust Structures

- An instance must define a set  $\mathcal{P}$  of *principal names* and a set  $T$  of possible *trust values*, ordered by  $\preceq$  and  $\sqsubseteq$ .
- $(T, \sqsubseteq)$  must be a complete lattice.
- For any collection  $\Pi$  of monotonic policies there is a unique global trust state, given by  $\text{gts} = \text{lfp } \Pi : \mathcal{P} \rightarrow \mathcal{P} \rightarrow T$ .
  - Interpretation:  $\text{gts}(p)(q)$  is  $p$ 's trust in  $q$ .
- The framework support the specification of imprecise or uncertain trust values.

# Operational Aspects

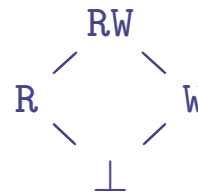
- Trust-structures  $TS = (T, \preceq, \sqsubseteq)$  give a framework for denotational semantics for collections of mutually referring trust policies.
- No good if principals are unable to reason about their own trust in others.
- $p \in \mathcal{P}$  wants to compute  $(\text{lfp } \Pi_\lambda)_p : \mathcal{P} \rightarrow T$ 
  - Problem: function  $\Pi_\lambda$  is distributed as  $\pi_q, q \in \mathcal{P}$ .
  - Problem: in principle  $(\text{lfp } \Pi_\lambda)_p$  depends on  $\pi_q$  for all  $q \in \mathcal{P}$ .

# Operational Aspects

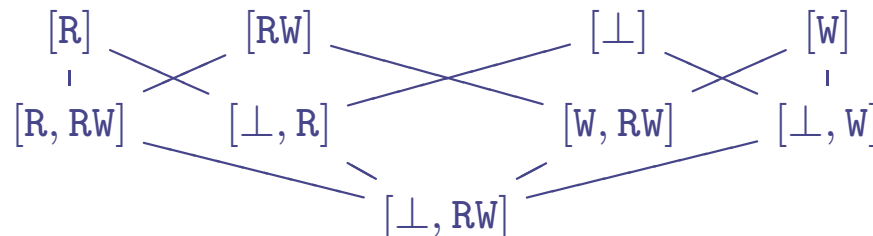
- Trust-structures  $TS = (T, \preceq, \sqsubseteq)$  give a framework for denotational semantics for collections of mutually referring trust policies.
- No good if principals are unable to reason about their own trust in others.
- $p \in \mathcal{P}$  wants to compute  $(\text{lfp } \Pi_\lambda)_p : \mathcal{P} \rightarrow T$ 
  - Problem: function  $\Pi_\lambda$  is distributed as  $\pi_q, q \in \mathcal{P}$ .
  - Problem: in principle  $(\text{lfp } \Pi_\lambda)_p$  depends on  $\pi_q$  for all  $q \in \mathcal{P}$ .
  - In practice, perhaps  $\pi_p$  depends on a significantly smaller subset.
  - Dynamically compute dependency, and then run a distributed least-fixed-point algorithm.

# Constructing trust-structures.

- Suppose I have a structure  $(D, \leq)$  of 'trust values' without uncertainty.
- If  $(D, \leq)$  is a complete lattice, then  $(ID, \preceq, \sqsubseteq)$  is a trust-structure.
  - Access-rights often form a complete lattice.



- Intervals introduces uncertainty in a canonical way.



- Starting point: Define a formal model for trust to be deployed in the SECURE project.

# PhD Studies: Present

- Starting point: Define a formal model for trust to be deployed in the SECURE project. ✓

# PhD Studies: Present

- Starting point: Define a formal model for trust to be deployed in the SECURE project. ✓
- Spin-off problems:
  - Consider trust models more generally, study notion of trust structures  $(T, \sqsubseteq, \preceq)$ .

# PhD Studies: Present

- Starting point: Define a formal model for trust to be deployed in the SECURE project. ✓
- Spin-off problems:
  - Consider trust models more generally, study notion of trust structures  $(T, \sqsubseteq, \preceq)$ .
  - Operational aspects of a denotational framework: algorithms and approximation protocols.

# PhD Studies: Present

- Starting point: Define a formal model for trust to be deployed in the SECURE project. ✓
- Spin-off problems:
  - Consider trust models more generally, study notion of trust structures  $(T, \sqsubseteq, \preceq)$ .
  - Operational aspects of a denotational framework: algorithms and approximation protocols.
  - Dynamics: Formalisation of *behaviour*.

# The future?

- Reality-check: ✓?
  - Assess the usefulness of trust-structures, the SECURE model, transfer-functions. . .
  - Practical aspects of algorithms.

# The future?

- Reality-check: ✓?
  - Assess the usefulness of trust-structures, the SECURE model, transfer-functions. . .
  - Practical aspects of algorithms.
- Develop further the notion of trust-structures  
 $TS = (T, \preceq, \sqsubseteq)$ .
  - Natural axioms.
  - Specification and proof of trust-based security properties.
    - e.g., if  $\pi$  assigns value  $t$  to  $p$ , then  $p$ 's behaviour  $b$  satisfies  $\phi(b)$ .
  - Development of protocols valid in every trust-structure.

# Publications

- Nielsen, M., Krukow, K.: Towards a formal notion of trust. In: Proceedings of the 5th ACM SIGPLAN international conference on Principles and Practice of Declarative Programming, ACM Press (2003) 4–7
- Cahill, V., Krukow, K., et al.: Using trust for secure collaboration in uncertain environments. IEEE Pervasive Computing **2** (2003) 52–61
- Nielsen, M., Krukow, K.: On the formal modelling of trust in reputation-based systems. To be published in Springer Lecture Notes in Computer Science. (2004)