# A Formal Model for Trust in Dynamic Networks

Marco Carbone[1] Mogens Nielsen[1] Vladimiro Sassone[2]

[1] BRICS*, University of Aarhus [2] COGS, University of Sussex

## Abstract

*We propose a formal model of trust informed by the Global Computing scenario and focusing on the aspects of trust formation, evolution, and propagation. The model is based on a novel notion of trust structures which, building on concepts from trust management and domain theory, feature at the same time a trust and an information partial order.*

## Introduction

Global Computing (GC) is an emerging aspect of computer science and technology. A GC system is composed of entities which are autonomous, decentralised, mobile, dynamically configurable, and capable of operating under partial information. Such systems, as e.g. the Internet, become easily very complex, and bring forward once again the need to guarantee security properties. Traditional security mechanisms, however, have severe limitations in this setting, as they are often either too weak to safeguard against the actual risks, or so stringent to impose unacceptable burdens on the effectiveness and flexibility of the infrastructure. *Trust management systems*, whereby safety critical decision are made based on trust policies and their deployment in the presence of partial knowledge, have an important role to play in GC.

This paper focuses on the foundations of formal models for trust in GC-like environments, capable of underpinning the use of trust-based security mechanisms as an alternative to the traditional ones.

Trust is a fundamental concept in human behaviour, and has enabled collaboration between humans and organisations for millennia. The ultimate aim of our research on trust-based systems is to transfer such forms of collaboration to modern computing scenarios. There will clearly be differences between the informal notion of trust explored in the social sciences and the kind of formality needed for computing. Mainly, our models need in the end to be operational, so as to be implementable as part of GC systems. Equally important is their role in providing a formal understanding of how trust is formed from complex interactions between individuals, so as to support reasoning about properties of trust-based systems.

Although our notion of trusted entity intends to cover only computing entities – even though of variable nature, spanning from soft to hard devices of all sorts – familiarity with trust models from the social sciences is a good starting point for our search of a foundational, comprehensive formal model of trust. One of our main sources has been the work by McKnight and Chervany [15], who provide a typology of trust used to classify existing research on trust in domains like sociology, psychology, management, economics, and political sciences. Trust is thereby classified conceptually in six categories: *disposition*, when entity *a* is naturally inclined to trust; *situation*, when *a* trusts a particular scenario; *structure*, when *a* trusts impersonally the structure *b* is part of; *belief*, when *a* believes *b* is trustworthy; *intention*, when *a* is willing to depend on *b*; *behaviour*, when *a* voluntarily depends on *b*. Orthogonally, the notion of *trustee* is classified in categories, the most relevant of which decree that *b* is trusted because of its *competence*, *benevolence*, *integrity*, or *predictability*. We believe that a good mathematical model of computational trust should be capable of expressing all such aspects, as well as further notions of primary relevance in computing, e.g. that trust information is time dependent and, in general, varies very rapidly. Also, it should be sufficiently general to allow complex structures representing combinations of different types of trust.

We think of the standard deployment of a trust management system as consisting of a "*trust engine*" and a "*risk engine*" coupled together as part of a "*principal*." The trust engine is responsible for updating trust information based on direct and indirect observations or evidence, and to provide trust information to the risk engine as input to its procedures for handling requests. The risk engine will feed back information on principals' behaviours as updating input to the trust engine. Abstracting over this point of view, we

MC and MN supported by '**SECURE**: Secure Environments for Collaboration among Ubiquitous Roaming Entities', EU FET-GC IST-2001-32486. MC supported by '**DisCo**: Semantic Foundations of Distributed Computation', EU IHP 'Marie Curie' HPMT-CT-2001-00290. VS supported by '**MyThS**: Models and Types for Security in Mobile Distributed Systems', EU FET-GC IST-2001-32617. *Basic Research In Computer Science funded by the Danish National Research Foundation.

single out as central issues for our trust model the aspects of trust *formation*, *evolution*, and *propagation*. The latter is particularly important in our intended application domain, where the set of active principals is large and open-ended, and centralised trust and ad-hoc methods of propagation of its variations make little sense. An important propagation mechanism is *delegation*, whereby principals cooperate to implement complex, intertwined "global" trusting schemes. Just to pin down the idea, bank $b$ may be willing to trust client $c$ to an overdraft limit $x$ only if bank $b'$ trusts it at least up to $2x/3$, and $c$ itself does not trust $d$, a crook known to $b$. Delegation has important consequences for trust representation, because it brings forward the idea of *trust policy*, i.e. algorithmic rules – such as bank $b$'s above – to evaluate trust requests. In principle, trust among principals can be represented straightforwardly, as a function from pairs of principals to trust levels,

$$\mathtt{GTrust} : \mathtt{Principal} \longrightarrow \mathtt{Principal} \longrightarrow \mathtt{TrustDegree}$$

where $\mathtt{GTrust}(a)$ is a function which associates to each principal $b$ the value of $a$'s trust in $b$. Delegation leads to model local policies, say $b$'s, as functions

$$\mathtt{TrustPolicy} : \mathtt{GTrust} \longrightarrow \mathtt{Principal} \longrightarrow \mathtt{TrustDegree}$$

where the first argument is (a representation of) a universal trust function that $b$ needs, to know $b'$'s level of trust in $c$ and whether or not $c$ trusts $d$.

The domain of `TrustPolicy` makes the core of the issue clear: we are now entangled in a "*web of trust*," whereby each local policy makes reference to other principals' local policies. Technically, this means that policies are defined by mutual recursion, and global trust is the function determined collectively by the web of policies, the function that stitches them all together. This amounts to say that `GTrust` is the least *fixpoint* of the universal set of local policies, a fact first noticed in [21] which leads straight to *domain theory* [19]. Domains are kinds of partially ordered sets which underpin the semantic theory of programming languages and have therefore been studied extensively. Working with domains allows us to use a rich and well-established theory of fixpoints to develop a theory of security policies, as well as flexible constructions to build structured trust domains out of basic ones. This is precisely context and the specific contribution of this paper, which introduces a novel domain-like structure, the *trust structures*, to assign meaning and compute trust functions in a GC scenario. We anticipate that, in due time, techniques based on such theories will find their way as part of trust engines.

As domains are (complete) partial orders and trust degrees naturally come equipped with an ordering relation (actually a lattice structure), a possible way forward is to apply the fixpoint theory to `TrustDegree` viewed as a domain. This is indeed the way of [21] and, as we motivate below, it is not a viable route for GC. There are very many reasons in a dynamic "web of trust" why a principal $a$ trying to query $b$ about $c$ may not get the information it needs: $b$ may be temporarily offline, or in the process of updating its policy, or experiencing a network delay, or perhaps unwilling to talk to $a$. Unfortunately, the fixpoint approach would in such cases evaluate the degree of trust of $a$ in $c$ to be the lowest trust level, and this decision would be wrong. It would yield the wrong semantics. Principal $a$ should not distrust $c$, but accept that it has not yet had enough information to make a decision about $c$. What is worse with this confusion of "trust" with "knowledge," is that the information from $b$ could then become available a few milliseconds after $a$'s possibly wrong decision.

We counter this problem by maintaining two distinct order structures on trust values: a *trust ordering* and an *information ordering*. The former represents the degree of trustworthiness, with a least element representing, say, absolute distrust, and a greatest element representing absolute trust; the latter the degree of precision of trust information, with a least element representing no knowledge and a greatest element representing certainty. The domain-theoretic order used to compute the global trust function is the information order. Its key conceptual contribution is to introduce a notion of "*uncertainty*" in the trust value principals obtain by evaluating their policies. Its technical contribution is to provide for the "semantically right" fixpoint to be computed.

Following this lead, we introduce and study trust structures of the kind $(D, \preceq, \sqsubseteq)$, where the two order relations over the set $D$, carry the meaning illustrated above. We then provide constructions on trust structures – including an "interval" construction which endows complete lattices with a natural notion of uncertainty and lifts them to trust structures – and use the results to interpret a toy, yet significant policy language. We believe that introducing the information ordering alongside the trust ordering is a significant step towards a model of trust feasible in a GC scenario; it is a major point of departure from the work of Weeks [21], and the central contribution of this paper.

**Plan of the document.** In §1 we define our trust model along the lines illustrated above, whilst §2 focuses on trust structures, providing methods for constructing useful structures as well as a general method to add uncertainty to the model. In §3 we introduce a policy language and use our trust structure to give it a denotational semantics.

**Related Work.** Trust is a pervasive notion, thoroughly studied in a variety of different fields, including social sciences, economics and philosophy. Here we only survey recent work on trust as a subject in computing; the reader is referred to [15] for a broader interpretation. A detailed survey can be found in Grandison and Sloman's [9].

Most of the existing relevant work concerns system building. In [18], Rivest *et al.* describe SDSI, a public key infrastructure featuring a decentralised name space which allows principals to create their own local names to refer to other principals' keys and in general, names. Ellison *et al.* [8] proposed a variation of the model which contributes flexible means to specify authorisation policies. The proposals are now merged in a single approach, dubbed SPKI/SDSI. Other systems of practical relevance include PGP [24], based on keys signed by trusted certificating authorities; KeyNote [2], which provides a single, unified language for both local policies and credential containing predicates to describe the trusted actions granted by (the holders of) specific public keys; and REFEREE [5], which uses a tri-valued logic which enriches the booleans with a value `unknown`. Trust in the framework of mobile agents is discussed e.g. in [22]. Delegation plays a relevant rôle in trust-based distributed systems. A classification of delegation schemes is proposed by Ding *et al.* [7], where they discuss implementation and analyse appropriate protocols. The ideas expressed in [7] lie at a level different from ours, as their focus is exclusively on access control.

The theoretical work can be broadly divided in two main streams: logics, where the trust engine is responsible for constructing [4, 3, 11, 12, 13] or checking [1] a proof that the desired request is valid; and computational models [21, 6], like our approach.

Burrows *et al.* propose the BAN logic [4], a language for expressing properties of and reasoning about the authentication process between two entities. The language is founded on cryptographic reasoning with logical operators dealing with notions of shared keys, public keys, encrypted statements, secrets, nonce freshness and statement jurisdiction. In [3], Abadi *et al.* enhance the language by introducing delegation and groups of principals: each principal can have a particular role in particular actions. The Authorisation Specification Language (ASL) by Jajodia *et al.* [11] separates explicitly policies and basic mechanisms, so as to allow a more flexible approach to the specification and implementation of trust systems. ASL supports also role-based access control.

Modal logics have a relevant place in specifying trust models, and have been used to express possibility, necessity, belief, knowledge, temporal progression, and more. Jones and Firozabadi [12] address the issue of reliability of agents' transmissions using a modal logic of actions [16] to model agents. Rangan [17] views a distributed system as a collection of communicating agents in which an agent's state is the history of its messages. Rangan's model builds on simple trust statements to define simple properties, which are then used to specify systems and analyse them with respect to properties of interest. Recently, Jøsang [13] proposed a logic of uncertain probabilities, a work which is related to our interval construction and can be recast as an instance of it in our framework. Specifically, Jøsang considers intervals of belief and disbelief over real numbers between 0 and 1.

Concerning computational models, Weeks [21] provides a model based on fixpoint computations which is of great relevance to our work. Winsborough and Li [23] study automated trust negotiation, an approach to regulate the exchange of sensitive credentials in untrusted environments. Clarke *et al.* [6] provide an algorithm for "certificate chain discovery" in SPKI/SDSI whereby principals build coherent chains of certificates to request and grant trust-based access to resources.

## 1 A Model for Trust

The introduction has singled out the traits of trust most relevant to our computational scenario: trust involves *entities*, has a *degree*, is based on *observations* and ultimately determines the *interaction* among entities. Our model will target these aspects primarily.

Entities will be referred to as *principals*. They form a set $\mathcal{P}$ ranged over by $a, b, c, \ldots$ and $p$. We assume a set $\mathcal{T}$ of *trust values* whose elements represent degrees of trust. These can be simple values, such as $\{$`trusted`, `distrusted`$\}$, or also structured values, e.g. pairs where the first element represents an action, say access a file, and the second a trust level associated to that action; or perhaps vectors whose elements represent benevolence in different situations.

As trust varies with experience, a model should be capable of dealing with observations resulting from the principal's interaction with the environment. For clarity, let us isolate the principal's trust management from the rest of its behaviour, and think of each principal as having a "trust box," that is an "object" module containing all of its trust management operations and data. In this paper, we only focus on the trust box and assume, without loss of generality, that the remaining parts of the principal interact with it via appropriately exported methods.

### Modelling the Trust Box

Principals' mutual trust can be modelled as a function which associates to each pair of principals a trust value $t$ in $\mathcal{T}$:

$$m : \mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}$$

Function $m$ applied to $a$ and then to $b$ returns the trust value $m(a)(b) \in \mathcal{T}$ expressing $a$'s trust in $b$. This however does not mean that a single principal's trust can be modelled as a function from $\mathcal{P}$ to $\mathcal{T}$, since $a$'s trust values may depend on other principals' values. For instance, $a$ may wish to enforce that its trust in $c$ is $b$'s trust in $c$. Similarly, we may be willing to receive a message from unknown sources, provided somebody we know trusts the sender. This mechanism of relying on third-party assessments, known as *delegation*, is fundamental in all scenarios involving cooperation, including computational paradigms such as GC.

This leads us to a refined view of a principal's trust as being defined by a *policy*. According to such a view, each principal has a local policy $\pi$ which contributes by way of delegation to form the global trust $m$. A policy expresses how the principal computes trust information given not just his own beliefs, but also other principals' beliefs. It follows that, as anticipated in the introduction, $a$'s policy $\pi_a$ has the type below, whose first argument represents the knowledge of third principals' policies that $a$ needs to evaluate $\pi_a$.

$$\pi_a : (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}) \longrightarrow (\mathcal{P} \longrightarrow \mathcal{T})$$

In this paper we leave unspecified the way a policy is actually defined, as this definitely depends on the application. We study a relevant example of policy language in §3.

By collecting together the individual policies, we obtain a function $\Pi \triangleq \lambda p : \mathcal{P}.\pi_p$ whose type is (isomorphic to)

$$\Pi : (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}) \longrightarrow (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}).$$

To interpret this collection of mutually recursive local policies as a global trust function $m$, we apply some basic domain theory, namely fixpoints and complete partial orders. We recall below the main notions involved; in general we assume the reader to be acquainted with partial orders (cf. [10] for a thorough introduction). Given a partial order $(T, \sqsubseteq)$, an $\omega$-chain $\mathsf{c}$ is a monotone function from the set of natural numbers $\omega$ to $T$; that is $\mathsf{c} = (c_n)_{n \in \omega}$ such that $c_0 \sqsubseteq c_1 \sqsubseteq c_2 \sqsubseteq \ldots$

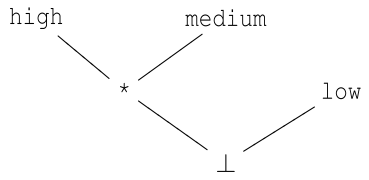
**Definition 1 (CPOs and Continous functions).** A partial order $(T, \sqsubseteq)$ is a *complete partial order* (CPO) if it has a least element $\perp$ and each $\omega$-chain $\mathsf{c}$ in $T$ has a least upper bound $\bigsqcup \mathsf{c}$. A function $f$ between CPOs is continuous if for each $\omega$-chain $\mathsf{c}$, it holds that $\bigsqcup f(\mathsf{c}) = f(\bigsqcup \mathsf{c})$.

The importance of CPOs here is that every continuous function $f : (T, \sqsubseteq) \to (T, \sqsubseteq)$ on a CPO has a least fixpoint $\mathsf{fix}(f) \in T$, that is a least $x$ such that $f(x) = x$ (cf. [19]). So, requiring $\mathcal{T}$ to be a CPO, which implies that $\mathcal{P} \to \mathcal{P} \to \mathcal{T}$ is a CPO too, and taking $\Pi$ to be continuous, we can define the global trust as $m \triangleq \mathsf{fix}(\Pi)$, the *least fixpoint* of $\Pi$.

The question arises as to what order to take for $\sqsubseteq$. We maintain that it *cannot* be the order which measures the degree of trust. An example is worth many words. Let $\mathcal{T}$ be the CPO $\{\mathtt{low} \leq \mathtt{medium} \leq \mathtt{high}\}$, and consider a policy $\pi_a$ which delegates to $b$ the degree of trust to assign to $c$. In this setup, $a$ will assign `low` trust to $c$ when it is not able to gather information about $c$ from $b$. This however would be an erroneous conclusion, as the interruption in the flow of information does not bear any final meaning about trust, its most likely cause being a transient network delay that will soon be resolved. The right conclusion for $a$ to draw is not to distrust $c$, but to acknowledge that it does not know (yet) whether or not to trust $c$. In other words, if we want to model dynamic networks, we cannot allow confusion between "`don't trust`" and "`I don't know`:" the latter only means lack of evidence for trust or distrust, the former implies a trust-based, possibly irreversible decision.

In order to make sense of our framework in a GC scenario, we need to acknowledge that principals only have a partial knowledge of their surroundings and, therefore, of *their own* trust values. We thus consider *approximate* trust values which embody a level of *uncertainty* as to which value we are actually presented with. Specifically, beside the usual *trust value ordering*, we equip trust values with a *trust information ordering*. While the former measures the degree of trustworthiness, the latter measures the degree of uncertainty present in our trust information, that is its information content. We will assume that the set $\mathcal{T}$ of (approximations of) trust values is a CPO with an ordering relation $\sqsubseteq$. Then $t \sqsubseteq t'$ means that $t'$ "refines" $t$, by providing more information than $t$ about what trust value is being approximated. With this understanding the continuity of $\Pi$ is a very intuitive assumption: it asserts that the better determined the information from the other principals, the better determined is value returned by the policy. An example will help to fix these ideas.

**Example 1.** Let us refine the set of trust values $\mathcal{T}$ discussed previously by adding some new intermediate values $\{\perp, *, \mathtt{low}, \mathtt{medium}, \mathtt{high}\}$, and consider the information ordering $\sqsubseteq$ specified by the following Hasse diagram.



Note that this ordering says nothing about what is more trust. It focus only on the quantity of information a principal has. The limit of any chain reflects the finest information. The element $*$ represents the uncertainty as to whether `high` or `medium` holds, while $\perp$ gives no hint at all about the actual trust value. Suppose we have a set of principals $\mathcal{P} = \{a, b, c\}$ with the following policies.

| | $a$ | $b$ | $c$ |
|---|---|---|---|
| $a$ | `high` | $\perp$ | `ask` $b$ |
| $b$ | $*$ | `high` | `low` |
| $c$ | `ask` $b$ | `high` | `high` |

where each row is a principal's policy. For instance the third row gives $c$'s policy: $c$'s trust in $a$ is $b$'s trust in $a$; $c$'s trust in $b$ is `high`. After a few interactions in which the principals' exchange their current values, following fixpoint is reached.

| | $a$ | $b$ | $c$ |
|---|---|---|---|
| $a$ | `high` | $\perp$ | `low` |
| $b$ | $*$ | `high` | `low` |
| $c$ | $*$ | `high` | `high` |

We reiterate that, importantly, the ordering $\sqsubseteq$ is not to be identified with the equally essential ordering "more trust."

## 2 Trust Structures

Having pointed out the need for order structures equipped at the same time with an information and a trust ordering, in this section we focus on the triples $(\mathcal{T}, \preceq, \sqsubseteq)$, which we call *trust structures*, and study their basic properties. The notion of complete lattice, recalled below, will play a relevant role.

**Definition 2 (Complete lattice).** A partial order $(D, \leq)$ is a *complete lattice* if every $X \subseteq D$ has a least upper bound (lub) and, as a consequence, a greatest lower bound (glb). We use $\vee$ and $\wedge$ to denote, respectively, lubs and glbs in lattices.

When defining a trust management system, it is natural to start off with a set $D$ of trust values, or degrees. On top of that, we are likely to need ways to compare and combine elements of $D$ so as to form, say, a degree which comprehends a given set of trust values, or represents the trust level common to several principals. This amounts to start with a complete lattice $(D, \leq)$, where those combinators can be considered as taking lubs or glbs of sets of values. To account for uncertainty, we define an operator $I$ to extend a lattice $(D, \leq)$ to a trust structure $(\mathcal{T}, \preceq, \sqsubseteq)$. The set $\mathcal{T}$ consists of the set of intervals over $D$ which, besides containing a precise image of $D$ – viz. the singletons – represent naturally the notion of approximation, or uncertainty about elements of $D$.

### Interval Construction

We define now the ordering $\preceq$ which has been already considered in [14].

**Definition 3.** Given a complete lattice $(D, \leq)$ and $X, Y \subseteq D$ nonempty subsets we say that $X \preceq Y$ if and only if

$$\wedge X \leq \wedge Y \qquad \text{and} \qquad \vee X \leq \vee Y$$

Clearly, $\preceq$ is not a partial order on the subsets of $D$, as the antisymmetry law fails. We get a partial order by considering as usual the equivalence classes of $\sim \,=\, \preceq \cap \succeq$. It turns out that the intervals over $D$ are a set of representatives of such classes.

**Definition 4.** For $(D, \leq)$ a complete lattice, the set $I(D) = \{[d_0, d_1] \mid d_0, d_1 \in D,\ d_0 \leq d_1\}$, where $[d_0, d_1] = \{d \mid d_0 \leq d \leq d_1\}$ is the interval of $D$ determined by $d_0$ and $d_1$.

**Proposition 1.** *Let $X = [d_0, d_1]$ be an interval in $D$. Then, $\wedge X$ is $d_0$ and $\vee X$ is $d_1$.*

As a consequence of the proposition above we have that $X \sim [\wedge X, \vee X]$, for all $X \subseteq D$. Furthermore, $[d_0, d_1] \sim [d'_0, d'_1]$ implies that $d_0 = d'_0$ and $d_1 = d'_1$. The following lemma characterises $\preceq$ in terms of $\leq$.

**Lemma 1.** *For $[d_0, d_1]$ and $[d'_0, d'_1]$ intervals of $D$, we have $[d_0, d_1] \preceq [d'_0, d'_1]$ if and only if $d_0 \leq d'_0$ and $d_1 \leq d'_1$.*

We can now show that the lattice structure on $(D, \leq)$ is lifted to a lattice structure $(I(D), \preceq)$ on intervals.

**Theorem 1.** *$(I(D), \preceq)$ is a complete lattice.*

*Proof.* Let $S$ be a subset $\{[d_0^i, d_1^i] \mid i \in J\}$ of $I(D)$, for some $J \subseteq \omega$. Then $\bigvee S = [\vee d_0^i, \vee d_1^i]$. □

We now define an ordering on intervals which reflects their information contents. Such an ordering will be a CPO where we base fixpoint computations on. The task is quite easy: as the interval $[d_0, d_1]$ expresses a value between $d_0$ and $d_1$, the narrower the interval, the lesser the uncertainty. This leads directly to the following definition.

**Definition 5.** For $(D, \leq)$ a complete lattice and $X, Y \in I(D)$, define $X \sqsubseteq Y$ if $Y \subseteq X$.

Analogously to $\preceq$, we can characterise $\sqsubseteq$ in terms of $\leq$.

**Lemma 2.** *For $[d_0, d_1]$ and $[d'_0, d'_1]$ intervals of $D$, we have that $[d_0, d_1] \sqsubseteq [d'_0, d'_1]$ if and only if $d_0 \leq d'_0$ and $d'_1 \leq d_1$.*

Finally, as for the previous ordering, we have the following result.

**Theorem 2.** *$(I(D), \sqsubseteq)$ is a CPO.*

*Proof.* The least element of $(I(D), \sqsubseteq)$ is $D = [\wedge D, \vee D]$. The lub of an $\omega$-chain $[d_0^n, d_1^n]_n$ is $\bigsqcup [d_0^n, d_1^n]_n = [\vee d_0^n, \wedge d_1^n]$. □

The trust structures above give a method to model trust based systems. We remark that intervals are a natural way to express partial information: trust in a principal is $[d_0, d_1]$ when it could be any value between $d_0$ and $d_1$.

**Example 2 (Intervals in [0,1]).** Let $R$ stand for the set of reals between 0 and 1, which is a complete lattice with the usual ordering $\leq$, and let us consider the set $I(R)$ of intervals in $R$. It follows from the previous results that $(I(R), \preceq)$ is a complete lattice and $(I(R), \sqsubseteq)$ is a complete partial order. The trust domain so obtained is particularly interesting, as it allows us to express complex policies. In particular, it is related to the uncertainty logic [13], where an interval $[d_0, d_1]$ in $I(R)$ is seen as a pair of numbers where $d_0$ is called belief and $1 - d_1$ disbelief. Although a formal comparison with Jøsang's logic is beyond the scope of our presentation, in the following we shall rework a few simple examples from [13] in the present framework.

An important property of $(I(D), \preceq, \sqsubseteq)$ is stated below.

**Theorem 3.** *Relation $\preceq$ is continuous with respect to $\sqsubseteq$ and, conversely, relation $\sqsubseteq$ is continuous with respect to $\preceq$.*

## Lifting Operators

The continuity of the function $\Pi$ is an important requirement. This property depends on the operators used with the policies. In the sequel we give a useful result, with respect to our interval construction, which allows us to lift continuous operators in the original lattice $(D, \leq)$ to continuous operators in $(I(D), \sqsubseteq)$ and $(I(D), \preceq)$.

**Definition 6.** For $(D, \leq)$ and $(D', \leq')$ complete lattices and $f : D \longrightarrow D'$ a continuous function, let $I(f) : I(D) \longrightarrow I(D')$ be the *pointwise extension* of $f$ defined as

$$I(f)([d_0, d_1]) = [f(d_0), f(d_1)].$$

Note that in this definition the continuity of $f$ ensures that $I(f)$ is well defined.

An $\omega$-cochain in a complete lattice $(D, \leq)$, is an anti-monotone function $\mathsf{c} : \omega \to T$, that is a function such that $i \leq j$ implies $c_j \leq c_i$. A function $f : (D, \leq) \longrightarrow (D', \leq')$ is co-continuous iff for each $\omega$-cochain $\mathsf{c}$ in $D$, it holds that $\bigwedge f(\mathsf{c}) = f(\bigwedge \mathsf{c})$; $f$ is bi-continuous if it is continuous and co-continuous.

The following proposition states that all $\omega$-cochains in $(I(D), \sqsubseteq)$ have glbs.

**Proposition 2.** *Let $[d_0^n, d_1^n]$ be an $\omega$-cochain in $(I(D), \sqsubseteq)$. Then $\sqcap [d_0^n, d_1^n] = [\wedge d_0^n, \vee d_1^n]$.*

*Proof.* Symmetric to that of Theorem 2. □

We can now give the following result about lifted functions in trust structures.

**Theorem 4.** *For $(D, \leq)$ and $(D', \leq')$ complete lattices and $f : D \longrightarrow D'$ a bi-continuous function, the pointwise extension $I(f)$ is bi-continuous with respect to both the information and the trust orderings.*

*Proof.* Easy, from the definition of $I(f)$, together with Theorems 1 and 2 and with the bi-continuity of $f$. □

In the following examples we show how to apply the previous theorem to some interesting operators.

**Example 3 (Lub and glb operators).** The most natural operators, regarding lattices, are lub and glb. It is easy to see that they are bi-continuous in a complete lattice $(D, \leq)$. Exploiting Theorem 4 we can now state that lub and glb with respect to $\preceq$ are bi-continuous over $(I(D), \sqsubseteq)$.

**Example 4 (Multiplication and Sum).** When considering the interval construction over $R$, as in Example 2, we can extend the operators of sum (weighted) and multiplication over the intervals. In fact, given two intervals $[d_0, d_1]$ and $[d_0', d_1']$, the product is defined as

$$[d_0, d_1] \cdot [d_0', d_1'] = [d_0 \cdot d_0', d_1 \cdot d_1'],$$

which is exactly the extension of multiplication over reals. Similarly we can define sum as

$$[d_0, d_1] + [d_0', d_1'] = [d_0 + d_0' - d_0 \cdot d_0', d_1 + d_1' - d_1 \cdot d_1'].$$

These operations appear in [13] under the names of conjunction and disjunction.

**Example 5 (A non-lifted operator: Discounting).** Discounting, as defined in [13], is an operator which weighs the trust value received from a delegation according to the trust in the delegated principal.

$$[d_0, d_1] \rhd [d_0', d_1'] = [d_0 \cdot d_0', 1 - d_0 \cdot (1 - d_1')]$$

### 2.1 Product and Function Constructors

Our model should satisfy "context dependent" trust. By this we mean that trusting a principal $a$ to obtain information about restaurants does not mean that we trust $a$ about, say, sailing. We can accommodate this kind of situation using a simple property of lattices and CPO's. Namely, we can form products of trust structures where each component accounts for a particular context. For instance, using a domain of the form *Restaurants* × *Sailing* will allow us to distinguish about $a$'s dependability on the two issues of our example. The next theorem shows that extending the orders pointwise to products and function spaces gives the result we need.

**Theorem 5.** *Given two complete lattices $(D, \leq)$, $(D', \leq')$ and a generic set $X$ then*

1. *$I(D \times D')$ is isomorphic to $I(D) \times I(D')$;*
2. *$X \longrightarrow I(D)$ is isomorphic to $I(X \longrightarrow D)$.*

*Proof.* In both cases we have to show that there exists a bijective correspondence which preserves the orderings. For (1) the bijection is

$$[(d_0, d_0'), (d_1, d_1')] \overset{\sim}{\longmapsto} ([d_0, d_1], [d_0', d_1']).$$

As for (2), the bijection $I(X \longrightarrow D) \cong X \longrightarrow I(D)$ is witnessed realised by the mutually inverse mappings below.

$$[f_0, f_1] \overset{\sim}{\longmapsto} \lambda x.[f_0(x), f_1(x)]$$
$$g \overset{\sim}{\longmapsto} [\lambda x. \wedge g(x), \lambda x. \vee g(x)]$$

*Remark* 1. Theorem 4 holds for any bi-continuous function $f : D_0 \times \ldots \times D_n \longrightarrow D$. The pointwise lifting of $f$ gives a function $I(f) : I(D_0 \times \ldots \times D_n) \longrightarrow I(D)$ and from the result above we have that $I(f)$ is (isomorphic to) a function $F : I(D_0) \times \ldots \times I(D_n) \longrightarrow I(D)$.

## 3 A Policy Language

Following our discussion we propose to operate with a language for trust policies capable of expressing intervals, delegation, and a set of function constructions. We exemplify the approach by studying the simple policy language below.

### Syntax

The language consists of the following syntactic categories, parametric over a fixed trust lattice $(D, \leq)$.

$$\begin{array}{rll}
\pi ::= & \ulcorner p \urcorner & \text{(delegation)} \\
 & \mid \lambda x : \mathcal{P}.\ \tau & \text{(abstraction)} \\
 & & \\
p ::= & a \in \mathcal{P} & \text{(principal)} \\
 & \mid x : \mathcal{P} & \text{(vars)} \\
 & & \\
\tau ::= & [d, d] \in I(D) & \text{(value/var)} \\
 & \mid \pi(p) & \text{(policy value)} \\
 & \mid e \mapsto \tau; \tau & \text{(choice)} \\
 & \mid \mathtt{op}(\tau_1 \ldots \tau_n) & \text{(lattice op)} \\
 & & \\
e ::= & p = p & \text{(equality)} \\
 & \mid e\ \mathtt{bop}\ e & \text{(boolean op)}
\end{array}$$

Here $\mathtt{op}$ is a continuous function over $(I(D), \sqsubseteq)$, and $\mathtt{bop}$ is a standard boolean operator. The elements of the category $p$ are either principals or variables. The main syntactic category is $\pi$: it can be either delegation to another principal or a $\lambda$-abstraction. An element of $\tau$ can be an interval, the application of a policy, a conditional or the application of a continuous operator $\mathtt{op}$. The elements of $e$ are boolean functions applied to equalities between elements of $\mathcal{P}$.

It is worth noticing that such a simple language goes beyond delegation interpreted strictly. In fact, rather that allowing principals to merely delegate somebody to decide on their behalf, it allows them to consults with each other to form complex, informed trust judgements. The examples to follow will clarify this concept.

### Semantics

We provide a formal semantics for the language described above. As pointed out before, $\pi$ is a policy. Hence the semantic domain, as described in §1, will be the codomain of the function

$$\llbracket \pi \rrbracket_\sigma :\ (\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}) \longrightarrow (\mathcal{P} \longrightarrow \mathcal{T}),$$

where $\sigma$ is an assignment of values in $\mathcal{P}$ to variables. The semantic function $\llbracket \cdot \rrbracket_\sigma$ is defined by structural induction on the syntax of $\pi$ as follows.

$$\llbracket \ulcorner p \urcorner \rrbracket_{\sigma m} = m (\!| p |\!)_{\sigma m};$$
$$\llbracket \lambda x : P.\ \tau \rrbracket_{\sigma m} = \lambda p : \mathcal{P}. (\!| \tau |\!)_{\sigma\{p/x\}m}.$$

Here $(\!| \cdot |\!)_{\sigma m}$ is a(n overloaded) function which given an assignment $\sigma$ and a global trust function $m : \mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{T}$ maps elements of $p$, $\tau$, and $e$ respectively to the semantic domains $\mathcal{P}$, $I(D)$, and Bool as follows.

$$\begin{aligned}
(\!|\ [d_0, d_1]\ |\!)_{\sigma m} &= [d_0, d_1] \\
(\!| \pi(p) |\!)_{\sigma m} &= \llbracket \pi \rrbracket_{\sigma m} (\!| p |\!)_{\sigma m} \\
(\!| e \mapsto \tau_1; \tau_2 |\!)_{\sigma m} &= \mathsf{if}\ (\!| e |\!)_{\sigma m}\ \mathsf{then}\ (\!| \tau_1 |\!)_{\sigma m}\ \mathsf{else}\ (\!| \tau_2 |\!)_{\sigma m} \\
(\!| \mathtt{op}(\tau_1 \ldots \tau_n) |\!)_{\sigma m} &= \mathsf{op}\,((\!| \tau_1 |\!)_{\sigma m}, \ldots, (\!| \tau_n |\!)_{\sigma m}) \\
(\!| a |\!)_{\sigma m} = a \qquad (\!| x |\!)_{\sigma m} &= \sigma(x) \\
(\!| p_1 = p_2 |\!)_{\sigma m} &= (\!| p_1 |\!)_{\sigma m} = (\!| p_2 |\!)_{\sigma m} \\
(\!| e_1\ \mathtt{bop}\ e_2 |\!)_{\sigma m} &= (\!| e_1 |\!)_{\sigma m}\ \mathsf{bop}\ (\!| e_2 |\!)_{\sigma m}
\end{aligned}$$

Let $\{\pi_p\}_{p \in \mathcal{P}}$ be a an arbitrary collection of all policies, where $\pi_p = \lambda x : \mathcal{P}.\ \bot$ for all but a finite number of principals. The fixpoint semantics of $\{\pi_p\}_{p \in \mathcal{P}}$ is the global trust function determined by the collection of individual policies, and it is readily expressed in terms of $\llbracket \cdot \rrbracket_\sigma$:

$$\{\!|\ \{\pi_p\}_{p \in \mathcal{P}}\ |\!\}_\sigma = \mathsf{fix}(\lambda m. \lambda p. \llbracket \pi_p \rrbracket_{\sigma m}).$$

We believe that this policy language is sufficiently expressive for most application scenarios in GC, as supported by the following examples. Note however that our approach generalises to any choice of underlying trust structure $(\mathcal{T}, \preceq, \sqsubseteq)$, provided the operators used in the policy language are continuous with respect to the information ordering.

**Example 6 (Read and Write access).** Let $D = \{\mathtt{N}, \mathtt{W}, \mathtt{R}, \mathtt{RW}\}$ represent the access rights to principal's CVs. The set $D$ is ordered by the relation $\leq$

$$\forall d \in D. \mathtt{N} \leq d \qquad \text{and} \qquad \forall d \in D. d \leq \mathtt{RW}.$$

Let us consider how to express some simple policies in our language. The following policy says that LIZ's trust in BOB is at least $[\mathtt{W},\mathtt{RW}]$ and depends on what she thinks of $\mathtt{CARL}$. Instead, $\mathtt{LIZ}$'s trust in $\mathtt{CARL}$ will depend on her trust in $\mathtt{BOB}$: if it is above $[\mathtt{W},\mathtt{W}]$ then $[\mathtt{R},\mathtt{RW}]$ otherwise $[\mathtt{N},\mathtt{RW}]$.

$$\begin{aligned}\pi_{\mathtt{LIZ}} = \lambda x : \mathcal{P}.&\\ &x = \mathtt{BOB} \mapsto [\mathtt{W},\mathtt{RW}] \vee \ulcorner\mathtt{LIZ}\urcorner(\mathtt{CARL});\\ &x = \mathtt{CARL} \mapsto\\ &\quad([\mathtt{W},\mathtt{W}] \preceq \ulcorner\mathtt{LIZ}\urcorner(\mathtt{BOB}) \mapsto [\mathtt{R},\mathtt{RW}]; [\mathtt{N},\mathtt{RW}]);\\ &[\mathtt{N},\mathtt{RW}]\end{aligned}$$

This policy can be made dependent on someone else's belief. For instance, the above judgement about $\mathtt{BOB}$ is merged below with $\mathtt{PAUL}$'s belief (weighed by discounting).

$$\begin{aligned}\pi_{\mathtt{LIZ}} = \lambda x : \mathcal{P}.&\\ &x = \mathtt{BOB} \mapsto [\mathtt{N},\mathtt{W}] \vee \ulcorner\mathtt{LIZ}\urcorner(\mathtt{CARL})\\ &\quad \vee \ulcorner\mathtt{LIZ}\urcorner(\mathtt{PAUL}) \rhd \ulcorner\mathtt{PAUL}\urcorner(x);\\ &x = \mathtt{CARL} \mapsto\\ &\quad([\mathtt{W},\mathtt{W}] \preceq \ulcorner\mathtt{LIZ}\urcorner(\mathtt{BOB}) \mapsto [\mathtt{R},\mathtt{RW}]; [\mathtt{N},\mathtt{RW}]);\\ &[\mathtt{N},\mathtt{RW}]\end{aligned}$$

In this case $\mathtt{LIZ}$'s trust in $\mathtt{PAUL}$ is the bottom value $[\mathtt{N},\mathtt{RW}]$ which is going to be the left argument of the discounting operator $\rhd$.

**Example 7 (Spam Filter).** Let $R$ be as in Example 2. We illustrate some policies modelling filters for blocking spam emails. The set of principals $\mathcal{P}$ is the set of Internet domains from which we could receive emails, e.g. daimi.au.dk. A starting policy, where we suppose that our server spam.filter.edu knows no one, could be

$$\pi_1 = \lambda x : \mathcal{P}.\ x = \mathtt{spam.filter.edu} \mapsto [1,1]; [0,1],$$

meaning that only internal emails are trusted. It could happen that spam.filter.edu starts interacting with other principals. A likely event is that it receives a list of other universities' Internet domains, and decides to trust them to a large extent, and actually use their beliefs. We could have

$$\pi_2 = \lambda x : \mathcal{P}.\ x \in \mathit{UniList} \mapsto [.75,1];\ \bigvee_{y \in \mathit{UniList}} \ulcorner y \urcorner(x) \vee \pi_1(x),$$

where we suppose that "$\in$" stands for a chain of nested conditionals for all the elements of *UniList*. Let us suppose now that the filter receives emails from a certain number of suspicious addresses, and would like to single them out and enforce a special treatment for them. The policy could be updated as

$$\pi_3 = \lambda x : \mathcal{P}.\ x \in \mathit{BadList} \mapsto [0,.5];\ \pi_2(x).$$

The spam-filter could then decide to add a new level of badness and create the new list *VeryBadList*. At the same time, it would like to change the policy for *BadList* putting certain restrictions on the intervals returned as other universities' opinions.

$$\begin{aligned}\pi_4 = \lambda x : \mathcal{P}.&\\ &x \in \mathit{VeryBadList} \mapsto [0,.2];\\ &x \in \mathit{BadList} \mapsto \pi_2(x) \wedge [0,.5];\\ &\pi_2(x).\end{aligned}$$

As illustrated in the *Spam Filter* example, we see trust evolution as being modelled by suitable updates of policies, as response to, e.g., observations of principal behaviour. However, it is still not clear exactly what update primitives are required in practice. We are currently working on developing a calculus of of trust and principal behaviour, with features for trust policy updates We will return on this in the concluding section.

**Example 8 (Reputation Based Systems).** The work [20] presents a reputation-based model of trust, where each principal $a$ has an associated history $H_a$ of observations, or *events*. A history $(e_1,\ldots,e_n)$ indicates that event $e_i$ has happened after events $e_1,\ldots,e_{i-1}$, for all $i$. A principal can provide information to the others (a.k.a. 'recommending') based on its past history. This means that it is not trust being propagated between principals, but observations. Reputation is then defined to be (as a formula satisfied) when a principal has never been observed to ignore certain conditions, i.e., if it never misused a resource.

Our approach is flexible enough to express some of this. (A full treatment requires the integration of policy updates in the policy language.) The idea is to make history part of a policy $\pi$, so that a principal's trust decision process can be defined in terms of its own and other principals' past observations. Let us consider the example of a peer-to-peer file distribution system discussed in [20]. In such scenario, users are allowed to download provided that they allows at least one upload every three downloads. Let $\mathbb{B}$ be the set of ordered boolean values, with $\mathtt{ff} \le \mathtt{tt}$, and let $\mathbb{N}_\bullet$ be the set of natural numbers completed with a top element $\infty$. Histories are elements of $\mathbb{H} = \mathcal{P} \to \mathbb{N}_\bullet \times \mathbb{N}_\bullet$, i.e. functions which assign to principals the numbers of uploads and downloads they performed in the past. Then, $a$'s trust function $\pi_a$ is of the kind

$$\mathcal{P} \to \mathbb{H} \to \mathbb{B}$$

where we understand that $a$ after a history $h$ trusts $x$ to download if $\pi_a(x)(h)$ yields $\mathtt{tt}$. The $\mathtt{SERV}$er's policy can be written as follows in a suitable "sugared" version of our language:

$$\pi_{\mathtt{SERV}} = \lambda p : \mathcal{P}.\lambda h : \mathbb{H}.\ \mathsf{let}\ (u,d) = h(p)\ \mathsf{in}\ d \le 3u.$$

If access is granted, $h$ is updated in view of the next invocation by increasing $p$'s count of downloads and, correspondingly, its peer's count of uploads.

## Conclusion and Further Work

We presented a novel model for trust in distributed dynamic networks, such as those considered in Global Computing. The model builds on basic ideas from trust management systems and relies on domain theory to provide a semantic model for the interpretation of trust policies in trust-based security systems. Our technical contribution is based bi-ordered structures $(\mathcal{T}, \preceq, \sqsubseteq)$, where the information ordering $\sqsubseteq$ measures the information contents of data, and is needed to compute the fixpoint of mutually recursive policies, while the trust ordering $\preceq$ measures trust degrees and is used to make trust-informed decisions. Trust and information orderings, as relations, are continuous with respect to each other. Following this lead, we presented an interval construction as a canonical way to add uncertainty to trust lattices, and used the theory to guide the design and underpin the semantics of a simple, yet realistic policy language. We believe that the model can be used to explain existing trust-based systems, as well as help the design of new ones.

We based our investigation on the notion of (complete) lattice, since it is the standard in the literature. However, there are reasons to believe that *upper semilattices* – that is ordered structures in which only bounded sets have least upper bounds – provide a better starting model. From a modelling perspective, it is easy to think of situations in which it should not be possible to form the join of two trust level. For instance, in a starship's auto-destruction system, the capabilities "*possess key A*" and "*possess key B*" to ignite cannot be joined, as the capability of possessing *both* the keys is not contemplated in the system. From a theoretical point of view, the absence of a top element simplifies the development of trust structures and enriches their theory.

We remark that the constructions illustrated here can be understood in abstract (categorical) terms. We have chosen to spell them out in set theoretical details to reach a wider audience. In particular, looking at the partial order $(D, \leq)$ as a category, our interval construction $I$ can be seen as the free construction of a *double category* with all $\omega$-filtered colimits. Specifically, $\preceq$ and $\sqsubseteq$ are respectively the horizontal and vertical arrows, while least upper bounds and their (mutual) commutation laws are expressed by as colimits. Furthermore, the $(I(D), \preceq)$ component of the interval construction is exactly the *functor category* $Arr \to D$, where $Arr = \bullet \to \bullet$ is the category with due objects and one non-identity arrow between them. More generally, the construction is related to the Yoneda embedding, as the image of the *hom-functor* $\mathsf{Hom}_D : D^{op} \times D \to Set$ is $(I(D), \sqsubseteq)$. Starting from these observations, we are currently investigating abstract characterisations of the trust structures arising from the present work.

We are clearly still at the first steps of development, where we need to assess the generality of our approach by applying it to various scenarios. Regarding semantics, we aim at a theory to account for the dynamic modification of the "web of trust," as for instance occurs when a principal updates its trust policy. Such modifications introduce an element of non-monotonicity that we plan to investigate by extending our model with a "possible-world" semantics, where updating a policy marks the transition to a "new world" and triggers a (partial) re-computation of the global trust function.

One of the main challenges ahead is to complement the denotational model introduced here with an operational model. In developing such a model we will need for instance to address the question of how to compute trust information efficiently over the global network. The highly dynamic nature of the kind of networks we are interested in, and their lack of any central control whatsoever pose serious challenges. In many applications it will not be feasible or necessary at all to compute exact values: we thus aim at techniques which allow to compute sufficient approximations to trust values. One issue is, as mentioned above, the update of computed trust elements; it would be interesting to investigate dynamic algorithms to update the least fixpoint computation yielding the global trust function. Another important issue is trust negotiation, whereby requester and granter engage in complex protocols aimed to convince each other of their reciprocal trustworthiness (for the specific purpose at hand), without disclosing more evidence than necessary. Similar ideas appear in the literature as "proof carrying authentication" [1] and "automated trust negotiation" [23].

In order to focus on the operational mechanisms of trust evolution and propagation in a distributed setting, we are currently working on a *calculus of trust* where principals' behaviour is accounted for. The approach is in the style of process algebras. Each principal is identified by a triple $a\{A\}_\pi$, where $a$ is the principal's name, $A$ the behaviour which models its actions, and $\pi$ its trust policy, described in a language such as the one presented in this paper. The dynamics of the calculus consists of interactions between principals, as for instance in:

$$\begin{aligned} & a\{b(x)A \mid A'\}_\pi \mid b\{a\langle e\rangle . B \mid B'\}_{\pi'} \\ & \qquad\qquad \searrow \quad a\{A\{e/x\} \mid A'\}_\pi \mid b\{B \mid B'\}_{\pi'}. \end{aligned}$$

Such interactions are granted according to the involved principals' policies. Furthermore, principals can take decisions based on their trust policies and – most importantly – update their policies, as e.g.

$$a\{[\zeta] . A \mid B\}_\pi \quad \searrow \quad a\{A \mid B\}_{\zeta(\pi)},$$

where $\zeta$ is a suitable "policy transformer." The overall idea here is that policy updates are informed during $a$'s evolution in time by its history of (un)successful interactions with other principals. The following example illustrates the matter further:

$$a\boldsymbol{\{}b(x)\,\mathbf{.}\,(x=k)\mathbf{?}[\zeta_1]\,\mathbf{:}\,[\zeta_2]\boldsymbol{\}}_{\pi} \mid b\boldsymbol{\{}[d_0,d_1] \preceq \pi'(a)\mathbf{?}a\langle k\rangle\,\mathbf{:}\,\mathbf{0}\boldsymbol{\}}_{\pi'}$$

Here $a$ and $b$ run in parallel. Principal $a$ is willing to receive a message from $b$ and, depending on whether or not the received value is the expected $k$, it will update its policy $\pi$ by $\zeta_1$ or by $\zeta_2$. On the other hand, $b$ will attempt to interact with $a$ (and send $k$) depending on whether its current trust in $a$ is above the threshold $[d_0,d_1]$.

Our current work on such extended framework attempts to capture the evolutionary aspects of trust in dynamic networks, together with the study of properties and related analysis techniques of systems based on trust in such networks. A particular locus of activity regards the formulation of type systems for the static control of trust. For instance, one may want to guarantee that at any moment in time the opportunity of an interaction between $a\{A\}_\pi$ and $b\{B\}_{\pi'}$ can only present itself if the interaction is granted by the policies, say e.g. if $\perp \prec \pi(b)$ and $\perp \prec \pi'(a)$.

Finally, we are also investigating ways for expressing and studying security properties of systems based on dynamic trust evolution and propagation, such as those above. Among the many approaches to checking of security properties, behavioural equivalences are particularly appealing. A valid alternative could be designing a logic for expressing properties of principals.

**Acknowledgements.** We would like to thank Karl Krukow and the Secure project consortium and in particular the Aarhus group and the Cambridge Opera group for the development of the research. Many thanks go to Maria Vigliotti who contributed to early developments.

## References

[1] A. W. Appel and E. W. Felten. Proof-carrying authentication. In *Proc. 6th ACM Conference on Computer and Communications Security*, 1999.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. KeyNote: Trust management for public-key infrastructure. *LNCS*, 1550:59–63, 1999.

[3] M. Burrows, M. Abadi, B. W. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *LNCS*, 576:1–23, 1991.

[4] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *In Proceedings of the Royal Society, Series A*, 426:18–36, 1991.

[5] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. REFEREE: Trust management for web applications. *Computer Networks and ISDN Systems*, 29(8-13):953–964, 1997.

[6] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. `http://theory.lcs.mit.edu/˜rivest`, 1999.

[7] Y. Ding, P. Horster, and H. Petersen. A new approach for delegation using hierarchical delegation tokens. In *Communications and Multimedia Security*, pages 128–143, 1996.

[8] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. SPKI certificate theory. *Internet RFC 2693*, 1999.

[9] T. Grandison and M. Sloman. A survey of trust in internet application. *IEEE Communications Surveys, Fourth Quarter*, 2000.

[10] G. Gräzer. *Lattice Theory: First Concepts and Distributive Lattices*. Freeman and Company, 1971.

[11] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *Proc. of the 1997 IEEE Symposium on Security and Privacy, Oakland, CA*, 1997.

[12] A. J. I. Jones and B. S. Firozabadi. On the characterisation of a trusting agent. In *Workshop on Deception, Trust and Fraud in Agent Societies*, 2000.

[13] A. Jøsang. A logic for uncertain probabilities. *Fuzziness and Knowledge-Based Systems*, 9(3), 2001.

[14] U. W. Kulish and W. L. Miranker. *Computer Arithmetic in Theory and Practice*. Academic Press, 1981.

[15] D. H. McKnight and N. L. Chervany. The meanings of trust. *Trust in Cyber-Societies - LNAI*, 2246:27–54, 2001.

[16] I. Pörn. Some basic concepts of action. In *S. Stenlund (ed.), Logical Theory and Semantic Analysis. Reidel, Dordrecht*, 1974.

[17] P. V. Rangan. An axiomatic basis of trust in distributed systems. In *Symposium on Security and Privacy*, 1998.

[18] R. L. Rivest and B. Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.

[19] D. S. Scott. Domains for denotational semantics. *ICALP '82 - LNCS*, 140, 1982.

[20] V. Shmatikov and C. Talcott. Reputation-based trust management. In *Workshop on Issues in the Theory of Security (WITS)*, 2003.

[21] S. Weeks. Understanding trust management systems. In *Proc. IEEE Symposium on Security and Privacy, Oakland*, 2001.

[22] U. G. Wilhelm, L. Buttyàn, and S. Staamann. On the problem of trust in mobile agent systems. In *Symposium on Network and Distributed System Security*. Internet Society, 1998.

[23] W. H. Winsborough and N. Li. Towards practical automated trust negotiation. In *IEEE 3rd Intl. Workshop on Policies for Distributed Systems and Networks*, 2002.

[24] P. Zimmermann. *PGP Source Code and Internals*. The MIT Press, 1995.