# An Intensional Functional Model of Trust*

Kaiyu Wan and Vasu Alagar

**Abstract** Computers have been in use for many years to build high confidence systems in safety-critical domains such as aircraft control, space transportation and exploration, and nuclear power plant management. However, due to the recent rush in developing ubiquitous and pervasive computing applications and a demand from across the world to access information from shared sources, the mosaic of computing ecosystem has undergone a radical change. It is in this context that computing has to be made *trustworthy*. To build and manage a trustworthy system it is necessary to blend and harmonize socially acceptable and technically feasible norms of trust. This in turn requires a generic formal model in which trust categories can be specified, trusted communication can be enabled, and trustworthy transactions can be specified and reasoned about. In this paper we introduce a formal *intensional* model of trust and suggest how it can be integrated into a trust management system.

## 1 Introduction

Trust is a broad social concept, which defies a precise definition. We generally understand what trust is not when the outcome of an event or a transaction or an action is not something that we expected. In almost all societies trust is related to honesty, truthfulness, reliability, and competence. But none of these attributes can be precisely stated for use in automatic computing systems. The Oxford Reference

Kaiyu Wan
Department of Computer Science, East China Normal University, e-mail: `kywan@cs.ecnu.edu.cn`

V. Alagar
Department of Computer Science and Software Engineering, Concordia University, e-mail: `alagar@cs.concordia.ca`

* The research is supported by a grant from Natural Sciences and Engineering Research Council, Canada.

*Please use the following format when citing this chapter:*

Wan, K. and Alagar, V., 2008, in IFIP International Federation for Information Processing, Volume 263; *Trust Management II*; Yücel Karabulut, John Mitchell, Peter Herrmann, Christian Damsgaard Jensen; (Boston: Springer), pp. 69–85.

Dictionary (ORD) states that trust is "the firm *belief* in the *reliability* or *truth* or *strength* of an *entity*." The European Commission Joint Research Center (ECJRC) [11] defines trust as the "*property* of a business *relationship*, such that *reliance* can be placed on the business *partners* and the business *transactions* developed with them." Putting together both definitions and abstracting it we see that trust notion is a fundamental requirement for reliance. The definitions also suggest that trust is a *relation* between (groups of) entities, and the attributes that associates a *value* to it are reliability, truth, integrity (strength), (reliance) security. In this paper we refine this definition and provide a formal of it in order that trustworthy computing systems can be rigorously developed and trust management systems (TMS) can administer trust policies uniformly across different applications.

Grandison and Sloman [9] have provided a survey of trust in internet applications. Trust is thereby classified according to the different *contexts* in which services are demanded. Without defining what a context is, they have considered *access to a trustor's resources*, *certification of trustees*, *delegation*, and *infrastructure trust* as four possible contexts. As a consequence the classification may be regarded only as ad hoc. In this paper we use context in its full formality, as was first introduced by Alagar [3], and subsequently refined by Wan [16]. In the formal trust model that we define context is a parameter to a higher order function that defines trust. We call the function *intensional* for reasons explained below. The term "intension" has a different meaning from the term "intention" (an aim or plan), as explained below.

Intensional logic is a branch of mathematical logic used to precisely describe context-dependent entities. According to Carnap, the real meaning of a natural language expression whose truth-value depends on the context in which it is uttered is its *intension*. The *extension* of that expression is its actual truth-value in the different possible contexts of utterance. For an instance, the statement *"The capital of China is Beijing"* is intensional because its valuation depends on the context (here is the time) in which it is uttered. If this statement is uttered before 1949, the extensions of this statement are *False* (before 1949 the capital was Nanjing). However, if it is uttered after 1949, the extensions of this statement are *True*. Thus the real meaning of an expression is a function from contexts to values, and the value of the intension at any particular context is obtained by applying context operators to the intension. Basically, intensional paradigm provides intension on the representation level, and extensions on the evaluation level. Hence, intensional paradigm allows for a more declarative way of specifying trust and programming it without loss of accuracy.

McKnight and Chervany [13] have given a typology for classifying trusting behavior in domains such as sociology, psychology, political sciences, and management. A trusting behavior, when expressed in natural language and semantically interpreted through Carnap's intensional logic, becomes an intensional statement. The six-fold classification of trusting behavior of an entity *a*, called *trustor*, on another entity *b*, called *trustee*, given by McKnight and Chervany [13] are (1) *disposition* (an entity *a* is naturally inclined to trust, (2) *situation* (an entity *a* trusts *b* in a particular scenario, (3) *structure* (entity *a* trusts the structure (institution) of which the entity *b* is a member), (4) *belief* (entity *a* believes that *b* is trustworthy), (5) *behavior* (entity *a* voluntarily depends on entity *b*), and (6) *intention* (entity *a* is willing to depend on entity *b*). McKnight and Chervany [13] have also classified the behavior of trustees in categories. The most relevant reasons for trusting *b* are *competence*, *benevolence*, *integrity*, or *predictability*. The natural language statements corresponding to these trusting behaviors have a natural intensional logic interpretation.

In our formalism, context is multi-dimensional and is much more general and expressive than the *situational* notion. The intensional functional definition of trust has context as a parameter, and has the expressive power to express the six-fold trusting behavior categories. Hence, trust contexts and trust categories that we define in this paper are not to be confused with the trusting behavior categories. However, our trust model can be integrated into the application environments of the domains considered by McKnight and Chervany [13].

With respect to our formalism itself, the two most relevant related works are the seminal work of Weeks [19] and Carbone [5]. Weeks first recognized *least fixed point* as a way to compute global trust, assuming a complete partial order of the trust domain. Carbone strengthened the work of Weeks by adding *information ordering* to *trust value ordering*. According to Carbone, trust value ordering is to measure the degree of trustworthiness, while information ordering is to measure the degree of uncertainty present in the information content on which trust is based. Information ordering enables the refinement of trust values. In [8] the notion of context is recognized as important for trust management, however no formal treatment of it has been attempted. Context formalization has found applications in different application domains, such as language design, multi-agent systems, and system security [3, 18, 17]. The distinct properties of our formalism are the following: (1)Trust definition is intensional. The benefits of an intensional model are explained earlier; (2) Trust values are restricted to a domain which is a complete partial order (cpo) and a lattice and a metric space. This allows formalizing the degree of trust and trust measure; (3) Trust calculation formulas are automatically developed based on formal context calculus. Thus subjectivity in calculation is eliminated; (4) Global trust computation is both context dependent and dynamic. Thus, during *delegation* the principals should adhere to the specific context in which the cooperation is sought and the global trusting rules that are relevant to that context; and (5) Homomorphism between trust domains (chosen by participants) must exist, otherwise global trust can not be defined. This property is particularly important for distributed systems.

## 2 Formal Trust Model

We start with a set of requirements for trust. Trust is relation between a trustor, the subject that trusts a target entity, and a trustee, the entity that is trusted. It is possible to measure the level of trust, called trust value, of a trustor on a trustee in any given context. Trust values must be comparable. It must be possible to manipulate trust values across different contexts, and aggregation of contexts. It should be possible to refine trust values, going from a general (specific) to a specific (general) context. It must be possible to express indirect trust relations and compute indirect trust values between a trustor and a trustee. It must be possible to ground the theoretical results in practice. A formal model that fulfills the above requirements must be comprehensive enough to cover most of the trust categories [13]. This is an important requirement because in a practical application more than one trust category will arise.

A number of choices exist in choosing a relation model of trust. The model depends on the inclusion/exclusion of the properties *reflexive*, *symmetric*, and *transitive*. In addition, it is also possible to allow or disallow one of the properties *one-to-many*, *many-to-one* and *many-to-many*. The choice is essentially dictated by the application domain. We get around this web of choices by choosing an intensional function definition for assigning trust value. We require the domain of trust values, hereafter called trust domain, to be a lattice with a minimum element. We turn the trust domain into a metric space. This will enable us to compare trust values and compute trust values from trust expressions. We define trust categories, motivate how contexts may be constructed within a trust category and arrive at extensions to the intensional trust definition within each category.

## 2.1 Intensional Function Definition

Trust involves *entities* ($\mathscr{E}$), their *interactions* ($\mathscr{I}$) and *policies* ($\mathscr{P}$). An *active* entity, called a *principal* (or an *agent*), can initiate a transaction, negotiate a deal, and communicate its intentions. A *passive* entity is a resource which can only be acted upon by an active entity. Data stores, files, printers, and programs are passive entities. An interaction is always initiated by an active entity. A human being accessing the internet, a program that reads data from a data base, and a component requesting a service from another component are examples of typical interactions. When an entity $a$ requests service from another entity $b$, there is an implicit trust involved in that $a$ trusts in the ability of $b$ to provide the requested service. The credentials of the service provider, in this case $b$, and the *contract* announced by $b$ that binds the service are convincing enough for $a$ to trust $b$. Credentials and contracts, similar to utterances as explained earlier, are declarative statements. Trust is usually not absolute, it is always relative and contextual. Consequently declarative trust statements are regarded as intensions. That is, a trust statement (policy) must be evaluated at different contexts to extract their extensions. Hence, the hypothesis that a policy, although will be stated declaratively, will always be applicable to one or more specific contexts, is both fair and sound. This is the rationale for formally introducing *context* ($\mathscr{C}$) as another modeling element. Context and contextual information provide the qualitative aspect of trust, and trust domain ($\mathscr{D}$) provides the quantitative aspect to the model. That is, the information in the context justifies the trust value assigned. A formal treatment of context appears in [17], and a logic of reasoning with context for multi-agent systems appears in [16]. We briefly summarize the most relevant results in section 3.

### 2.1.1 Properties of Trust Domain

We require that the elements of a trust domain, also called trust values, are chosen as the most appropriate ordinal type to indicate the level (degree) of trust of $a$ on $b$ in a context $c$. Trust values can be either simple values, such as numbers or symbols, or vectors. The numeric values themselves may be either discrete, such as natural numbers or rational, or continuous such as real values. If symbols are used then we regard it as an *enumerated* type, the order being implicit in the enumeration. As an example, in the enumeration $\{hold, sell\}$, the degree of trust in *sell* recommendation is higher than the degree of trust in *hold* recommendation. Real numbers, natural numbers, and rational numbers are totally ordered sets, and hence degree of trusts are comparable. In the case of vector values, it is necessary to define an ordering. In general, it is sufficient to require that the trust values are partially ordered, which is the weakest relation necessary to compare an element with at least one other element in the set.

### 2.1.2 Partial Order

Assume that $\simeq$ is a reflexive partial order defined on $\mathscr{D}$. Consequently trust relationship is transitive, which goes against the opinion expressed in [15]. Since context was not a factor in previously published trust definitions the transitivity property promoted a wide open trust delegation policy, which was not always desirable. Because of the introduction of context, transitivity is constrained to contexts that are *compatible* and delegation can be enabled only in such contexts. As an example, if Alice delegates her trust decisions to Bob in the context of *installing a virus protection software in her computer*, and Bob trusts Cathy in the context of *Linux installation*, then the installation of virus protection cannot be delegated to Cathy. We also remark that computing transitivity is part of global trust computation and it should not change the local trust policies of any entity. With these safeguards, the partially ordered domain $(\mathscr{D}, \simeq)$ is quite safe. We also require $(\mathscr{D}, \simeq)$ to have a least element $\perp$, in the sense that $\forall d \in \mathscr{D}$, $\perp \simeq d$. An $\omega$-chain on $(\mathscr{D}, \simeq)$ is a monotone function $\iota : \omega \rightarrow \mathscr{D}$, where $\omega$ is the set of natural numbers. That is in the sequence $\iota = (\iota_n)_{n \in \omega}$, $\iota_1 \simeq \iota_2 \simeq \ldots \iota_n$. An element $\iota_u \in \mathscr{D}$ such that $\iota_k \simeq \iota_u$, $k = 1, \ldots, n$ is called a *least upper bound (lub)* of $\omega$. In order that $(\mathscr{D}, \simeq)$ be a cpo every $\omega$-chain in it must have a least upper bound. Interpreting $\perp$ to mean "undefined trust value" the function defined below becomes a total function.

**Definition 1** *The function, $\tau : \mathscr{E} \times \mathscr{E} \times \mathscr{C} \rightarrow \mathscr{D}$ associates for $a, b \in \mathscr{E}$, and $c \in \mathscr{C}$ a unique element $d \in \mathscr{D}$, called the trust that $a$ has on $b$ in context $c$. That is, $\tau(a, b, c) = d$. The function $\tau$ is context-dependent and expresses the trust* intension *in different contexts. So we call $\tau$ an intensional function.* □

## 2.2 Further Properties of Trust Domain

By imposing a metric, in addition to a partial order, we can measure the disparity between two trust levels. Finally, we also define a homomorphism between trust domains in order that trust value from one domain may be carried into the other trust domain. In a distributed system, where each site may have a different trust domain, without a homomorphic mapping it is not possible to calculate and compare trust globally.

### 2.2.1 Metric Space

Let $(\mathscr{D}, \simeq)$ be a cpo. We define a total monotone function $\rho :: \mathscr{D} \rightarrow \omega$ which assigns a non-negative integer value to each trust value such that for $d_1, d_2 \in \mathscr{D}$

- if $d_1 \simeq d_2$ then $\rho(d_1) \leq \rho(d_2)$, and
- for every $\omega$-chain $\iota : \omega \rightarrow \mathscr{D}$, $\rho(\iota_1) \leq \rho(\iota_2) \leq \ldots \rho(\iota_n)$.

The function $\delta : \mathscr{D} \times \mathscr{D} \rightarrow \omega$ defined by $\delta(d_1, d_2) = \mid \rho(d_1) - \rho(d_2) \mid$ satisfies the properties

- $\delta(d_1, d_2) \geq 0$
- $\delta(d_1, d_2) = \delta(d_2, d_1)$, and
- for $d_1, d_2, d_3 \in \mathscr{D}$, $\delta(d_1, d_3) \leq \delta(d_1, d_2) + \delta(d_2, d_3)$

Hence $(\mathscr{D}, \rho, \delta)$ is a metric space. The usefulness of this exercise is clear when we want to quantify the disparity between $d_1 = \tau(a, b, c)$ and $d_2 = \tau(a, b, c')$, which is precisely $\delta(d_1, d_2)$. If $d_1 < d_2$, then by going from context $c$ to context $c'$, $a$'s trust in $b$ has increased by the amount $\delta(d_1, d_2)$. That is, metrication helps to reason about trust refinement and trust evolution.

### 2.2.2 Homomorphism

The trust domain may be chosen differently by different developers for the same application domain. In a distributed network, each local site may have a different trust domain. In order that trust values across sites can be compared and global trust computed it is necessary that the homomorphism $f$, defined below, is a continuous function on the $\omega$ chains.

$f : (\mathscr{D}, \simeq, \rho) \rightarrow (\mathscr{D}', \simeq', \rho')$

- for $d_1, d_2 \in \mathscr{D}, f(d_1 \simeq d_2) = f(d_1) \simeq' f(d_2)$
- $f(\perp) = \perp'$,
- $f(\rho(d_1, d_2)) = \rho'(f(d_1), f(d_2))$

**Theorem 1** *For every $\omega$-chain $\iota$ in $(\mathscr{D}, \simeq)$, $\iota' = f(\iota)$ is a $\omega$-chain of $(\mathscr{D}', \simeq')$, and $f(\iota_u) = \iota'_u$,* *Proof: Since $\iota : \omega \rightarrow \mathscr{D}$ is monotonic and $f$ is continuous the composition $f \circ \iota$ is monotonic. But, $\iota' : \omega \rightarrow \mathscr{D}'$ satisfies the equation $\iota' = f \circ \iota$. Hence $\iota'$ is monotonic.*

**Theorem 2** *Let $\tau : \mathscr{E} \times \mathscr{E} \times \mathscr{C} \rightarrow \mathscr{D}$ and $\tau' : \mathscr{E} \times \mathscr{E} \times \mathscr{C} \rightarrow \mathscr{D}'$ be two intensional functions that give the trust values on the trust domains $\mathscr{D}$ and $\mathscr{D}'$. Then $f \circ \tau = \tau'$*

*Proof: The proof follows from the two facts $f \circ \rho = \rho'$ and $f \circ \iota = \iota'$.*

## 2.3 Trust Expressions

By requiring that every subset $\mathscr{S} \subset \mathscr{D}$ has a least upper bound (lub) and a greatest lower bound (glb) in $\mathscr{D}$ we turn the trust structure to a complete lattice. For $d_1, d_2 \in \mathscr{D}$, we write $lub(d_1,d_2) = d_1 \vee d_2$ and $glb(d_1,d_2) = d_1 \wedge d_2$. In general $lub(\mathscr{S}) = \bigvee_{d \in \mathscr{S}} d$ and is written $\bigvee \mathscr{S}$. Similarly, $glb(\mathscr{S}) = \bigwedge_{d \in \mathscr{S}} d$, and is written $\bigwedge \mathscr{S}$. We also define $\rho(d_1 \wedge d_2) = min\{\rho(d_1), \rho(d_2)\}$, and $\rho(d_1 \vee d_2) = max\{\rho(d_1), \rho(d_2)\}$. In general, $\rho(glb(\mathscr{S})) = min_{d \in \mathscr{S}}\{\rho(d)\}$ and $\rho(lub(\mathscr{S})) = max_{d \in \mathscr{S}}\{\rho(d)\}$. A trust expression over $(\mathscr{D}, \simeq, \rho)$ is defined by the syntax $\tau_e = d \mid \tau_e \vee \tau_e \mid \tau_e \wedge \tau_e$. An expression $\tau_e$ is evaluated from left to right. In an evaluation we let $d \vee \perp = d$, and $d \wedge \perp = \perp$. Thus every expression uniquely evaluates to a trust value $d \in \mathscr{D}$, and hence a unique $\rho(d)$, although not every expression can be assigned a meaning. As an example, the expression $\tau(a_1,b,c) \wedge \tau(a_2,b,c)$ represents the trust level that both $a_1$ and $a_2$ can agree upon the entity $b$ in context $c$. However, the expression $\tau(a_1,b_1,c_1) \vee \tau(a_2,b_2,c_2)$, where $a_1 \neq a_2$, $b_1 \neq b_2$, and $c_1 \neq c_2$ has a unique value in $\mathscr{D}$, but the value is hard to interpret in a meaningful way. With this extension we can write many other trust expressions of the form $\tau_e \vee (\bigwedge \mathscr{S})$, $\tau_e \wedge (\bigvee \mathscr{S})$.

# 3 Trust Contexts - a formalization

In this section we discuss the notion of *trust contexts* (TC), motivate the need to formalize this notion in a discussion of trust modeling, and their importance in trust management. We follow the formal definition of context, the syntax for context representation, and review the basics of context calculus given by Wan in [17].

In [9] trust is defined as “the firm belief in the competence of an entity to act independently, securely, and reliably within a *specified context*”. It is observed in [8, 9] that “the notion of *context* is important in the SECURE trust model”. In the former, context is not defined. In the later, trust is regarded as a *multi-dimensional* quantity and the different dimensions are called trust contexts. However, no formal treatment of context is attempted. The definition of context in [17] includes dimensions, and *tags* (indexes) along the dimensions. The term “dimension” used by Dimmock ectal; [8] essentially refers to a knowledge domain, as suggested by the statement “...by analogy, a person who is trusted to *drive a car* may not be trusted to *fly a plane*.” In the above statement “driving a car” and “flying a plane” are defined as dimensions by Dimmock [8]. However in the work of Wan [17] “driving a car” and “flying a plane" are regarded as "different worlds of knowledge (information)" and the term "dimension" is used to systematically break down the structural information content in each world. For instance, in the world "driving a car" knowledge on the drivers, cars, and their associations are available. That is, information on the name ($N$) of driver, address ($A$) of the driver, date of issue ($DI$) of the driving permit, termination date ($DT$) of the driving permit, driving status ($DS$), car type ($CT$) and restrictions on the driver ($RD$) are available. In Wan's formalization these information categories are regarded as dimensions. The tag sets and their types for these dimensions are shown in Table 3.

| Dimension Name | Tag Set | Type |
|---|---|---|
| $N$ | a set of driver names | *alpha* (string) |
| $A$ | a set of addresses | record type (string, string, string) |
| $DI, DT$ | a set of dates | record type (int,int,int) |
| $DS$ | a set of discrete values | enumerated type<br>Example:(`learner`, `good`, `dangerous`) |
| $CT$ | a set of car models | enumerated type<br>Example:(`mini`, `compact`, `sport`) |
| $RD$ | a set of restrictions | enumerated type<br>Example: (day-time only, must wear glasses) |

**Table 1** Dimensions and Tag Sets for "Driving a Car"

We assume that a TMS exists whose major goal is *compliance checking*. It provides a systematic, and application-independent framework to managing security policies, credentials, and trust relationships. When a principal submits a request to perform an action, the access control manager (ACM) in TMS should dynamically determine dimensions, construct contexts and provide context-sensitive services that are correct with respect to the trust policy in that context. A policy usually specifies *who is trusted to do what*. In our model we attach a context condition along with a rule, as explained later.

### 3.1 Review of Context Formalization

We have motivated in [16] that five dimensions are essential for dynamically constructing contexts. These are [1.] *perception- who* (which entity) provides the service or requires the service?, [2.] *interaction* - *what* type of service (resource allocation, algorithms for problem solving) is required?, [3.] *locality* - *where* to provide the service?, [4.] *timeliness- when* to provide the service (what time bounds, or delay, or duration should be satisfied)?, and [5.] *reasoning* - *why* a certain action is required in providing the service (due to obligation, adaption, or context-aware requirements)? Example 1 illustrates the construction of contexts and compliance checking by a TMS.

**Example 1** *Consider the policy* Policy1: Either the surgeon *a* on-duty in a hospital or any surgeon *b* whom *a* trusts can be trusted to perform surgery of patients either admitted by *a* or treated by *a*. *This policy is in the "health care" trust category. The policy refers to physician name, her current status (on-duty or off-duty), patients admitted by her, and patients under her care. The context for enforcing the policy is suggested by the above information. The dimensions and their tag sets are are PN (a finite set of physician names), PS (a finite set of statuses of physicians), WS (a finite collection of work schedules), PA (a finite set of patients admitted) and PC (a finite set of patients cared). An example of a context c with these dimensions is represented in the syntax* $[PN : Bob, PS : on-duty, WS : 1, PA : Alice, PC : Tom]$. *This context describes the setting in which "physician Bob is on duty on the first day of the week, admitted Alice and cared for Tom". To check the compliance of "Policy1", the TMS constructs the current context* $c_1$*, and retrieves from its database the context* $c_2$ *in which the patient was admitted. The physician* $a_1$ *in context* $c_1$ *and the physician* $a_2$ *in context* $c_2$ *are both trusted to perform the surgery. Assume that the degree of trust demanded by* $a_1$ *on another surgeon is at least* $\varepsilon_1$*, and the degree of trust demanded by* $a_2$ *on another surgeon is at least* $\varepsilon_2$*. The TMS allows the* $b_1$ *or* $b_2$ *as stand-by surgeon for performing the surgery if* $\rho(\tau(a_1, b_1, c_1)) \geq \varepsilon_1$ *and* $\rho(\tau(a_2, b_2, c_2)) \geq \varepsilon_2$.

What is important is to make clear that "Alice" and "Tom" are patients and not hospital personnel. That is, context definition requires a unique dimension name for each entity type, because a hospital patient may also be an employee in the hospital. The set of dimensions and the tag sets for dimensions are usually suggested by the world knowledge associated with the trust category. In many applications, such as pervasive computing, contexts must be constructed dynamically in real-time. As an example, once again consider a hospital environment. Assume every nurse possesses a hand-held device equipped with sensory and communication capabilities. As the nurse goes around hospital wards the sensory information gathered by the device at any instant is transformed into a context. The dimensions for such contexts may be chosen as *CLOC* (the ward where the nurse is), *NLOC* (the next ward to be visited), *TIME* (current local time), and *DUR* (time taken to walk the distance between the wards). In general, once the dimensions and tags are determined context is formalized as a *typed relation*.

**Definition 2** *Let DIM denote the set of all possible dimensions and TAG denote a set of* totally ordered *sets. Let* $\bigcup TAG$ *be the set of tags for all dimensions in DIM. The set of all un-typed contexts is*

$$CO = \mathbb{P}(DIM \times \bigcup TAG)$$

*By letting each set in TAG to assume all possible tag values for some dimension in DIM we get typed contexts. Let* $CO^{\star}$ *denote the set of non-empty un-typed contexts, and* $CT = DIM \rightarrow TAG$ *be the type of all total surjective functions with DIM as domain and TAG as range. Let* $\kappa \in CT$*. The set*

$$G(\kappa) = \{c \mid c \in CO^{\star} \wedge dom\ c \subseteq dom\ \kappa \wedge \forall (X, x) \in c \bullet X \in DIM, x \in \kappa(X)\}$$

*denotes the set of typed contexts for the given type* $\tau$.□

If $DIM = \{X_1, \ldots, X_n\}$ and $\kappa(X_i)$, $i = 1, \ldots, n$ are respectively the tag sets for $X_i$, $i = 1, \ldots, n$, the syntax for a context is $[X_1 : x_1, \ldots, X_n : x_n]$, where $x_i \in X_i$. In the rest of the paper by context we mean a finite non-empty context, and omit explicit reference to $\kappa$ unless our discussion demands it. A context in which the dimensions are distinct is called *simple context*. It is shown in [17] that a non-simple context is equivalent to a set of simple contexts. In this paper we encounter only simple contexts.

| operator name | symbol | meaning | precedence |
|---|---|---|---|
| Union | $\sqcup$ | Set Union | 3 |
| Intersection | $\sqcap$ | Set Intersection | 3 |
| Difference | $\ominus$ | Set Difference | 4 |
| Subcontext | $\subseteq$ | Subset | 6 |
| Supcontext | $\supseteq$ | Superset | 6 |
| Override | $\oplus$ | Function overwrite | 4 |
| Projection | $\downarrow$ | Domain Restriction | 1 |
| Hiding | $\uparrow$ | Range Restriction | 1 |
| Undirected Range | $\rightleftharpoons$ | Range of simple contexts with same domain | 5 |
| Directed Range | $\rightharpoonup$ | Range of simple contexts with same domain | 5 |

**Table 2** Context Operators and Precedence

### 3.1.1 Context Calculus

A set of context operators are formally defined by Wan [17]. We review them here, give examples, and illustrate the evaluation of context expressions. For the sake of simplicity we assume that tag sets are the set of natural numbers.

Table 2 shows the context operators, their functionalities, and precedences. Example 2 illustrates the application of a few context operators.

**Example 2** *:*

*Let* $c_1 = [\,d:1,\ e:4,\ f:3\,]$, $D = \{\,d,\ e\,\}$

$c_1 \downarrow D = [\,d:1,\ e:4\,]$; $c_1 \uparrow D = [\,f:3\,]$

*Let* $c_2 = [\,e:3,\ d:1\ ,f:4]$, $c_3 = [\,e:1,\ d:3\,]$

$c_2 \oplus c_1 = [\,e:4,\ d:1,\ f:3]$; $c_1 \oplus c_2 = [\,e:3,\ d:1,\ f:4]$

$c_1 \ominus c_2 = [\,e:4,\ f:3]$; $c_1 \sqcap c_2 = [\,d:1]$

$c_3 = c_1 \sqcup c_2 = [\,d:1,\ e:3,\ e:4,\ f:3,\ f:4]$.

*Using the result in [17] $c_3$ can be written as a set of simple contexts:*

$c_3 = \{[\,d:1,\ e:3,\ f:3],[\,d:1,\ e:3,\ f:4],[\,d:1,\ e:4,\ f:3],[\,d:1,\ e:4,\ f:4]\}$

$$\begin{aligned} c_2 \rightleftharpoons c_3 = \{&[\,e:1,d:1,f:4],[\,e:1,d:2\ ,f:4],[\,e:1,d:3\ ,f:4],[\,e:2,d:1\ ,f:4],\\ &[\,e:2,d:2\ ,f:4],[\,e:2,d:3\ ,f:4],[\,e:3,d:1\ ,f:4],[\,e:3,d:2\ ,f:4],\\ &[\,e:3,d:3\ ,f:4]\} \end{aligned}$$

$c_2 \rightharpoonup c_3\ =\ \{[\,d:1,\ f:4],[\,d:2,\ f:4],[\,d:3,\ f:4]\}$

### 3.1.2 Context Expression

A context expression is a well-formed expression in which only contexts, context variables, and context operators occur. A well-formed context expression will evaluate to a context. As an example, $c_3 \uparrow D \oplus c_1 \mid c_2$, where $c_1 = [x:3, y:4, z:5]$, $c_2 = [y:5]$, and $c_3 = [x:5, y:6, w:5]$, $D = \{w\}$ is a well-formed expression. The steps for evaluating the expression according to the precedence rules shown in Table 2 are shown below:

[Step1]. $c_3 \uparrow D = [x:5, y:6]$ *[↑ Definition]*
[Step2]. $c_1 \mid c_2 = c_1$ *or* $c_2$ *[| Definition]*
[Step3]. *Suppose in Step2,* $c_1$ *is chosen,*
$c_3 \uparrow D \oplus c_1 = [x:3, y:4, z:5]$ *[⊕ Definition ]*
*else if* $c_2$ *is chosen,*
$c_3 \uparrow D \oplus c_2 = [x:5, y:5]$ *[⊕ Definition]*

### 3.1.3 Trust Context Categories

A *trust context category* is a set of trust contexts relevant to capture the knowledge in that category. As an example, the set of contexts $\{[N:n, DT:t, DS:s] \mid ct \leq t+6 \wedge s = \text{learner}\}$ belongs to the category "driving a car". These contexts are relevant to the action "determine the drivers who were granted `learner's permit` within the last six months (here *ct* stands for current time)". A trust category has its own trust domain which has the structural properties discussed in Section 2.

Every principal will choose its trust categories and submit it to the TMS, who in conjunction with ACS, has the knowledge and resources to determine the contexts for each trust category. For instance, if principal $a$ submits the categories $\pi_a = \{\pi_{a_1}, \ldots, \pi_{a_{k_a}}\}$ to the TMS then for each category $\pi_{a_i} \in \pi_a$ the TMS will determine a set $DIM_{a_i} = \{X^1_{a_i}, \ldots, X^{k_i}_{a_i}\}$ of dimensions, and a tag set $\kappa(X^j_{a_i})$ for each dimension $X^j_{a_i} \in DIM_{a_i}$. Whenever principal $a$ submits a request for action belonging to that category, with the help of the context tool kit, the TMS will construct contexts that are relevant for the action. In this scheme, the trust categories and contexts in those categories of a principal $a$ are known only TMS, ACS, and itself. The context toolkit is an implementation package of the context calculus. As an example, using the toolkit the TMS can calculate the set $s$ of simple contexts corresponding to a non-simple context $c$. It uses the rule

$\{s \mid \forall c' \in s \bullet dim(c') = dim(c) \wedge (X,x) \in c' \rightarrow (X,x) \in c\}$

As another example, the TMS can determine the set of contexts such that a given logical expression is true at every context in that set. To check the compliance with respect to the policy "A `learner` should pass road test examination within `six months` of the *date of expiry* of the learner's permit" the TMS should determine the contexts where the logical expression $(ct \leq t+6 \wedge s = \text{learner})$ is true. An application of this policy is warranted when the action " determine the set $L$ of drivers with

| $a$-Dimensions | $b$-Dimensions | $c$-Dimensions |
|---|---|---|
| LOC | WHERE | PLACE |
|  | WHEN | DATE |
| WHO | NAME | USER |
| WHAT |  | ACTION |
| WHY | PURPOSE |  |

**Table 3** Mapping Between Dimensions

learner permit issued within the last six months" is initiated. We call $(ct \leq t+6 \wedge s = \text{learner})$ a context condition. It is clear that a context condition, in general, may satisfy a set of contexts. We write $\mathbf{ist}(\alpha, c)$ to denote that expression $\alpha$ is true (valid) in context $c$. The context condition $\alpha$ for a context $c$ is an assertion on the *evidence* relevant to the context $c$. Example 3 illustrates the usefulness of context conditions.

**Example 3** *Let Bob trust a driver a if a can be certified to comply with the learner policy stated above. The degree to which Bob trusts the driving of Alice is at least as much as his trust on any learner who has complied with learner policy. We can formally express this trust policy of Bob as*

$\rho(\tau(Bob, Alice, c)) \geq minimum\{\rho(\tau(Bob, a, c))\}$,

*where* $a \in L \wedge \mathbf{ist}(\alpha, c)$*, and* $\alpha = ct \leq t+6 \wedge a = learner$

After constructing the trust contexts for all categories of a principal the TMS must resolve *conflict* among dimension names and *construct ontology* for "equivalent" dimensions.

- *resolve conflicts:* A conflict arises when a dimension name is common for two different categories of a principal and the tag sets of the dimensions are of different types. If these two types are sub types of a super type then the conflict is resolved by replacing the tag types with the super type. If a super type does not exist then the conflict is resolved by renaming the dimension in one trust category.
- *ontology:* It is possible that some principals have a common trust category, but the dimension names (with their tag names) chosen by the principals may be different. The TMS maintains an ontology table in which *equivalent* dimension names are listed. An example of the mapping between dimensions chosen by three principals is shown in Table 3. Informally, dimensions $X$ and $Y$ are equivalent if either they have the same tag set or the types of tag sets can be lifted to a unique super type. As an example, the dimensions *LOC*, *WHERE*, and *PLACE* are equivalent under the assumption that they have the "set of city names" as their tag.

# 4 Trust Calculation

Without loss of generality assume that all trust categories in $\pi_a$ have a common trust domain, say $\mathscr{D}_a$. For a given evidence $\alpha_i$ in the context category $\pi_{a_i}$, the trust of $a$ on $b$ over all contexts in the context category $\pi_{a_i}$ that satisfy the evidence $\alpha_i$ can be calculated in more than one way. Let $\mathscr{C}_{a_i}$ be the set of contexts in the trust category $\pi_{a_i}$.

- *minimum trust:* $\tau_i^l(a,b \mid \alpha_i) = \bigwedge_{\mathbf{ist}(\alpha_i,c) \wedge c \in \mathscr{C}_{a_i}} \tau(a,b,c)$
- *maximum trust:* $\tau_i^h(a,b \mid \alpha_i) = \bigvee_{\mathbf{ist}(\alpha_i,c) \wedge c \in \mathscr{C}_{a_i}} \tau(a,b,c)$
- *uncertainty:* Since trust is not absolute in any context there is an element of uncertainty associated with the assignment of trust values. The two potential approaches to deal with uncertainty in trust calculation are based on *trust intervals* and *trust probabilities*.
  - *trust interval* A domain of *trust intervals* $\mathscr{I}(\mathscr{D}_a)$ is constructed from $\mathscr{D}_a$ which is a cpo and a function $\mu$
    $\mu : \mathscr{E} \times \mathscr{E} \times \mathscr{C}_{a_i} \rightarrow \mathscr{I}(\mathscr{D}_a)$
    that assigns trust values to intervals is defined. This method is followed in [5]. We refine this approach, emphasizing that the interval $\mathscr{I}(\mathscr{D})$ must be a *sub interval* of the interval $[\tau_i^l(a,b \mid \alpha_i), \tau_i^h(a,b \mid \alpha_i)]$
  - *Probabilistic Model*: Trust values are considered as random variables over the domain $\mathscr{D}_a$. With respect to a probabilistic distribution in the trust domain $\mathscr{D}_a$ a probability $P(\tau(a,b,c) = d)$ is computed. That is, with probability $P(\tau(a,b,c) = d)$ the degree of trust of $a$ on $b$ is $d$ in context $c \in \mathscr{C}_{a_i}$. The principal $a$ may choose $d_{max} \in \mathscr{D}_a$ such that $d_{max} = max_{d \in \mathscr{D}_a}\{P(\tau(a,b,c) = d\}$ is a maximum, and assign it as the trust of $a$ on $b$ in context $c$. Further, $a$ can choose a threshold $\beta_a$, and may decide not to trust $b$ in context $c$ if the maximum probability is less than $\beta_a$.

Choosing any one of the above methods to calculate trust, the principal $a$ will have a trust vector
$\tau'(a,b \mid \alpha_a) = \langle \tau_1'(a,b \mid \alpha_{a_1}, \ldots, \tau_k'(a,b \mid \alpha_{a_k} \rangle$,
where $\alpha_a = \alpha_{a_1} \wedge \alpha_{a_2} \wedge \ldots \alpha_{a_k}$, $\alpha_{a_i}$ is the evidence for context category $\pi_{a_i}$. The set $\{\tau'(a,b \mid \alpha_a) \mid a \in \mathscr{E}_b\}$ is the set of trust vectors of principals in $\mathscr{E}$ who have calculated their trust for the participant $b$. At any instant the collection of all such vectors is part of the global trust in the system.

## 4.1 Trust Delegation and Refinement

Assume that $\alpha' \rightarrow \alpha$, where $\alpha$ and $\alpha'$ are context conditions. If $\mathbf{ist}(\alpha', c')$, and $\mathbf{ist}(\alpha, c)$, then a trust policy enforced in a context $c$ is quite relevant in context $c'$ defined by $\alpha'$ and should be enforced there. Hence for a principal $a$ and entity $b$ the trust degree $\tau_i(a,b \mid \alpha')$ is known then we conclude that $\tau_i(a,b \mid \alpha) = \tau_i(a,b \mid \alpha')$

for trust category $\pi_i$. In other words, logical implication propagates trust values from one context to another.

**Trust Delegation:** We can justify now that trust delegation is a form of trust propagation. Suppose principal $a_1$ has a trust policy "my trust on the entity $a_3$ is the same as the trust of $a_2$ on $a_3$. Assume that the policy is to be applied by $a_1$ in context $c_1$, and $\mathbf{ist}(\alpha_1, c_1)$. In context $c_1$, which is local to $a_1$, let $\tau(a_2, a_3, c_2) = d$, and $\alpha_2$ be the justification for computing this trust value. If $\alpha_2 \rightarrow \alpha_1$ the trust propagation happens, and $\tau(a_1, a_3, c_1) = d$. If $\alpha_2 \nrightarrow \alpha_1$ the trust delegation cannot happen.

**Refinement:** Refinement refers to calculating a more precise trust value than the one already available. Informally, the trust of $a$ on $b$ in context $c$ is *refined* when either some additional constraints are imposed on the information $W(c)$ or some additional information becomes available, thus creating a new context $c'$. Thus the trust value evolves as well as refined. Refinement need not be *monotonic*, meaning the addition of new information or the imposition of additional constraints on existing information may decrease the trust value.

Consider information content refinement by imposing constraints on existing information. For context $c \in \mathscr{C}$, let $W(c)$ denote the world of information referenced by $c$. Suppose for contexts $c_1, c_2 \in \mathscr{C}$, the world $W(c_2)$ referenced by $c_2$ is a restricted subset of the information in the world $W(c_1)$ referenced by $c_1$. The restriction is often imposed by constraints on the entity described in the world $W(c_1)$. Then the information in the world $W(c_2)$ is more precise or special than the information $W(c_1)$. This kind of sub-setting can be realized either semantically or syntactically. In the former case, a logical expression may be used. An example is $W(c_1)$ is the set of physicians and $W(c_2)$ is the set of surgeons. In the case of syntactic sub-setting, more dimensions are added to the set $dim(c_1)$ such that the world referenced by new contexts obtained by the addition of dimensions do not go outside $W(c_1)$. Formally a context $c_2$ is constructed such that $dim(c_2) = dim(c_1) \cup D$, where $D \subset DIM$ , $D \cap dim(c_1) = \varnothing$, such that $W(c_2) \subset W(c_1)$ . Thus, the additional dimensions in the set $dim(c_2) \setminus dim(c_1)$ have no extraneous influence on the information along the dimensions in the set $dim(c_1)$, instead it can only constrain it. An example is the context $c_2 \subset c_1$, where $c_2 = [GPS : Newyork, TIME : 10, NS : 12, EW : 3]$, and $c_1 = [GPS : Newyork, TIME : 10]$, with the interpretation that every event happening in context $c_2$ can be seen from context $c_1$. Hence, if $\alpha$ is a valid formula in $W(c_2)$ then it is possible to prove in context $c_1$ that $\alpha$ is true in context $c_2$. We define such a relationship between contexts as *visible*.

**Definition 3** *A context $c_2 \in S$ is said to be* visible *from context $c_1 \in S$, written $c_1 \succeq c_2$, if $c_1 \subset c_2$ and $W(c_2) \subset W(c_1)$.*□

From Definition 3 it follows that $\mathbf{ist}(c_2, \alpha) \Rightarrow \mathbf{ist}(c_1, \mathbf{ist}(c_2, \alpha))$. Consequently, the current context $c_1$ in which the value $\tau(a, b, c_2)$ is computed must satisfy the relation $c_1 \succeq c_2$. When information is added on to existing information context condition will change, and consequently contexts will change. For this case we offer a few axioms for simple situations. We write $d' \gtrsim d$ to mean that $d'$ is the refinement of $d$.

- *arbitrary subset:* $c_1 \subset c_2 \Rightarrow \tau(a, b, c_2) \gtrsim \tau(a, b, c_1)$

- *intersection:* $\tau(a,b,c_1) \gtrsim \tau(a,b,c_1 \sqcap c_2)$
  $\tau(a,b,c_2) \gtrsim \tau(a,b,c_1 \sqcap c_2)$
  $\tau(a,b,c_1) \wedge \tau(a,b,c_2) \gtrsim \tau(a,b,c_1 \sqcap c_2)$
- *disjoint union:* $\tau(a,b,c_1 \sqcup c_2) \gtrsim \tau(a,b,c_1)$
  $\tau(a,b,c_1 \sqcup c_2) \gtrsim \tau(a,b,c_2)$
  $\tau(a,b,c_1 \sqcup c_2) \gtrsim \tau(a,b,c_1) \vee \tau(ab,c_2)$

## 5 Trust Policy Framework

A trust model must include a formal framework for policy representation and policy application. A trust policy mentions entities (principals and objects), and suggests either directly or indirectly a sequence of actions to be done when the rule is followed. A Policy may mention a *role*, in which case it is applied to every entity playing that role. A policy in every trust category is a rule, an intensional statement. An example trust policy in health care category is *a physician can be trusted to access medical information on the patients under her care*. A policy, being a declarative statement, does not dictate how it should be represented in organizational databases and how it should be implemented. However we recommend that the policy representation include information on where it is applicable.

### 5.1 Policy Representation

A policy can be represented by a rule $H \Leftarrow B$, where $H$ is called the *head (consequent)* of the rule and $B$ is called the *body (antecedent)* of the rule. In general, the body of a rule is a conjunction of one or more conditions; no disjunction is allowed in the body. The head of a rule, expressed declaratively, is an action specification. We associate a context condition $U$ with each rule to suggest that the rule is applicable in any context that satisfies this condition. By separating the context condition from the rule we achieve rule generality, and flexibility in the application of the rule.

**Example 4** *Consider the rule* $U : \iota(x,y,z) == \iota(x',y,z) \Leftarrow \rho(\iota(w,x',z)) \geq \rho(\iota(w,y,z))$*, where the context condition is* $U \equiv physician(x) \wedge nurse(y) \wedge head\_nurse(w) \wedge physician(x') \wedge c \in \mathscr{C}$*. The rule states that in context c that satisfies the context condition U, physician x trusts nurse y to the same degree that physician* $x'$ *trusts y provided nurse w trusts physician* $x'$ *more than the trust she has for nurse y".*

The TMS maintains the policy bases, one for each principal in the system. Let $\{PB_a \mid a \in \mathscr{E}\}$ denote the set of policy bases. The policy bases are secured in the sense that $PB_a$ can be accessed only by the principal $a$ and the ACS within the TMS. For an efficient processing of transactions, we propose two methods to organize the rules in each policy base.

- *partitioning:* The set $PB_a$ of rules is partitioned so that each subset has policies associated with a specific trust category. As an example, corresponding to the trust categories $\pi_a$ of the principal $a$, the policy base $PB_a$ is partitioned into $\{PB_{a_1}, \ldots, PB_{a_k}\}$.
- *Linking Partitions:* A rule $r :: U : H \Leftarrow B$ is relevant to the rule $r' :: U' : H' \Leftarrow B'$ if $U' \rightarrow U$. For a rule $r \in PB_{a_i}$ the rules in $PB_{a_i}$ that are relevant to $r$ are linked.

Following the links and applying the trust propagation rule the global trust can be computed at any instant. The global state of the TMS at any instant is given by the elements of the trust model, the organization of the policy base and the global trust computed from them.

## 6 Conclusion

The formal trust model proposed in this paper is both new and novel, yet it shares many properties of other trust models [5]. The basic function that computes trust is context-specific. This function is made use of in defining trust in trust categories. The explicit introduction of context in the computation of trust, annotating trust policies with context conditions, and defining delegation through related contexts are some of our new results given in this paper. The important benefits in the intensional definition of trust are (1) Context is independent of what it references. As a consequence, context-based trust definition captures different kinds information conveyed by events that happen in a context; (2)Context calculus enables the construction of new contexts from existing contexts, and the logic of contexts [16] enables one to reason about the information in the newly constructed context with respect to the information contents in the contexts from which the new one is constructed. As a consequence context-based trust definition is well-suited to handle trust in dynamic networks, in which contexts and their respective information contents may dynamically change independent of each other; (3) The model becomes more general and expressive because we incorporate what Grandison and Sloman [9] and [8] have strongly proposed, but failed to formalize.

The contribution made by us in this paper is a quick summary of what we have been doing during the last few months. With the full support of context theory and its logic of reasoning [16] we plan to add more formal arsenal to our trust model. These include devising rules for reasoning, and transporting our formalism to a practical stage in which trustworthy systems can be developed. We expect this work to follow very closely the ongoing work [1, 2].

## References

1. Vasu Alagar and Mubarak Mohammad. *A Component Model for Trustworthy Real-Time Reactive Systems Development.* In *Proceedings of the 4th International Workshop on Formal*

*Aspects of Component Systems (FACS'07)*, September 19 - 21, 2007. Sophia-Atipolis, France.

2. Vasu Alagar and Mubarak Mohammad. *Specification and Verification of Trustworthy Component-Based Real-Time Reactive Systems.* In *Proceedings of the Specification and Verification of Component-Based Systems Workshop (SAVCBS'07)*, September 03 - 04, 2007, Dubrovnik, Croatia. (ACM Portal http://portal.acm.org/citation.cfm?id=1292316.1292327 & coll=GUIDE & dl=)
3. V.S. Alagar, J. Paquet, K. Wan. *Intensional Programming for Agent Communication.* Proceedings of the 2nd International Workshop on Declarative Agent Languages and Technologies (DALT) 2004, New York, U.S.A., June 2004, LNAI Springer-Verlag, Vol. 3476, Page 239-255. ISBN: 3-540-26172-9.
4. R. Carnap. *Meaning and Necessity.* Chicago University Press, 1947. Enlarged Edition 1956.
5. Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. *A formal model for trust in dynamic networks.* Research Series RS-03-04, BRICS, Department of Computer Science, University of Aarhus, January 2003, EU Project SECURE IST-2001-32486 Deliverable 1.1.
6. N. Damianou, N. Dulay, E. Lupu, and M. Solomon. *The Ponder Policy Specification Language.* Proceedings Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29–31, Jan. 2001.
7. J. DeTreville. *Binder, a logic-based security language.* Proceedings of the 2002 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 2002, 105-113.
8. Nathan Dimmock, Andr*á*s Belokosztolszki, and David Eyers. *Using Trust and Risk in Role-Based Access Control Policies.*, In *Ninth ACM Symposium on Access Control Models and Technologies (SACMAT)*, Yorktown Heights, New York, June 2004.
9. Tyrone Grandison and Morris Sloman. *A Survey of Trust in Internet Applications.* IEEE Communications Surveys, Fourth Quarter 2000, 1-16.
10. Audun J$\phi$sang, Elizabeth Gray, and Michael Kinateder. *Analyzing topologies of transitive trust.* In *Proceedings of the Workshop of Formal Aspects of Security and Trust (FAST)*, September 2003.
11. S. Jones. *TRUST-EC: Requirements for Trust and Confidence in E-Commerce.* European Commission, Joint Research Center, 1999.
12. G. Klyne. *w*ww.ninebynine.org/iTrust/Intro.html, 2008.
13. D. Harrison McKnight and Norman L. Chervany. *Trust and Distrust Definitions: One Bite at a Time.* In *Trust in Cyber Societies - LNAI*, 2246:27-54, 2001.
14. Craig Mundie, Peter de Vries, Peter Haynes, and Matt Corwine. *Trustworthy Computing - Microsoft White Paper,* Microsoft Corporation, October 2002.
15. D. Povey. *Trust Management,* 1999, http://security.dstc.edu.au/presentations/trust.
16. KaiYu Wan, Vasu Alagar. *A Context Theory for Multi-agent Systems.* Revised Draft, January 2008.
17. Kaiyu Wan, *Lucx: Lucid Enriched with Context,* Ph.d Thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, January 2006.
18. K. Wan, V.S. Alagar. *An Intensional Programming Approach to Multi-agent Coordination in a Distributed Network of Agents.* Proceedings of the 3rd International Workshop on Declarative Agent Languages and Technologies (DALT) 2005, Utrecht, The Netherlands, July 25, 2005. Page 148-164, LNCS Springer-Verlag, Vol. 3904, pp. 205-222.
19. Stephen Weeks. *Understanding trust management systems.* In *Proceedings of IEEE Symposium on Security and Privacy.*, Oakland, 2001.