Probabilistic Computational Trust

Karl Krukow[1], **Mogens Nielsen**[2], **Vladimiro Sassone**[3]

[1] *Trifork*
Denmark
kkr@trifork.com

[2] *University of Aarhus*
Denmark
mn@science.au.dk

[3] *University of Southampton*
UK
vs@ecs.soton.ac.uk

Abstract

We argue briefly for the role of computational trust in ubiquitous computing, and in particular for the need of a formal foundation for computational trust. We provide two examples towards such a foundation: a formal foundation for a some probabilistic approaches from the literature, and a formal framework for comparing probabilistic trust models.

Keywords. Computational Trust, Ubiquitous Computing.

1 Introduction

This paper surveys the notion of computational trust and illustrates some initial progress towards developing its theoretical underpinning. Computational trust refers to *decision-making* in computing applications where *uncertainty* is a dominant aspect. This may manifest itself in several different ways, including unpredictability in the execution environment, as typical of e.g. mobile and sensor networks; in the emergent behaviour of dynamic agglomerates of computational entities, as e.g. in self-configuring, highly-distributed systems; and in applications where the optimal strategy is itself uncertain, as typical of adaptation and situational awareness in autonomous agent systems. The tract common to these examples is the move away from the 'Eden' of certain and abundant information to a place where the only 'truths' available are those that can be experienced directly. This is analogous to higher living organisms, like we humans, which continuously have to assess the evidence around them and decide whether or not to rely on the information they determine from it. To provide a purposely naive example

in realm of computing, on the Internet it is ultimately a matter of trust whether or not I expect that following a given URL will actually take me to my bank's website. Such is in fact the origin of the term 'trust' in our context: in the absence of complete and completely reliable information, computational entities must weigh the relative importance of different factors —that is, determine their level of trust in them— in their decision-making. Indeed, 'decision-making' remains the central keyword here, and one can think of computational trust as a computer abstraction underpinning it.

To develop such an abstraction is not as immediate as it might appear at first. It is common experience to enter a room filled with strangers, sit around a table and soon establish a level of trust sufficient to function cooperatively (well, at least in most cases). This does not come as naturally to machines, and it is in fact from our point of view a superbly sophisticated collective behaviour: to achieve a similar level of adaptability, awareness and responsiveness from computing agents requires computational structures, algorithms, middleware infrastructures, models and theoretical foundations, and is exactly what the work on computational trust aspires to.

An application field central to our interests is ubiquitous computing. That is a comprehensive term which indicates information processing through a computational infrastructure embedded seamlessly and pervasively in the surroundings. It encompasses very many issues of a quite different nature, ranging from the design and deployment of low-power, self-sustaining electronic devices in pervasive wireless networks, to the investigation of innovative user interfaces, from the development of semantic models of agent mobility and distribution, to the architecture of communication networks and the corresponding programming language primitives. Computational trust is intrinsic to ubiquitous computing in that entities on the 'global' network have intrinsically unverifiable identities, origins, and past histories.

The area of computational trust has produced several applications within ubiquitous computing with truly impressive experimental performance. However, we are not yet in a position where we understand why, when, and how a particular approach is applicable, as expressed e.g. in [20]. Such questions are typically formulated in terms of underlying models. In this paper, we focus on just one type of models considered within computational trust based on well known concepts from probability theory.

Probabilistic approaches have proved useful in science to formulate and test hypotheses over quantities not exactly known, as illustrated by e.g. Thomas Bayes, who developed his eponymous method by solving the so-called problem of 'inverse probability:' given that an outcome has been observed (e.g., a red ball has been extracted from the urn), what can be inferred about the model (e.g., number of red and yellow

balls in the urn)? This adapts well to our 'decision-making' problem; e.g.: given that the URL led to my bank's website, what can be said about the rest of the information on this page to inform my future decisions? We shall illustrate in this paper how such an observation led to a powerful computational idea, which has been exploited in several computational trust algorithms. Indeed, as the field grows under the thrust of suggestive analogies like the one above, and experimental successes pile up, the need arises to understand at a deeper level and distil the essence of the methods.

Our aim in this paper is primarily to illustrate that it is possible to ask and to answer formally questions on the behaviour of systems expressed in underlying formal probabilistic trust models. We focus here on behavioural notions like *correctness*. Traditionally in software development, the notion of correctness is formulated in terms of a yes/no question: does a piece of software satisfy its specification? It is our position that in the setting of computational trust, we need to develop new formal frameworks for a more general notion of correctness, which allows us e.g. (i) to express and to argue *how well* a particular system behaves under various assumptions about the environments (i.e. in which application scenarios does the system do well?), and (ii) to express and argue *how robust* a particular system is with respect to changes in the environment. And in this paper we present ideas towards new frameworks for addressing both these issues. However, these are just examples of new types of formal frameworks to be developed within computational trust. We hope this paper can serve to illustrate our results on 'formal' computational trust and as a 'call-to-arms' to tackle the open ones.

Contents of paper: We present a formal foundation for two of the well known probabilistic approaches from the literature. We then illustrate how a new approach to correctness and robustness can be applied to answer questions on the relative performance of the two approaches. We first summarise some of the arguments for trust playing a role in ubiquitous computing, giving a brief historical account on the development from trust management systems. We then focus on a few probabilistic approaches to computational trust, and we illustrate how they can be understood and explained formally in terms of standard concepts from probability theory. Finally, we sketch ideas towards a theoretically well-founded technique for comparing probabilistic systems in various different environments. The paper is mainly based on selected parts of the PhD dissertation of the first author [13]. Also, some of the results have appeared in [21, 14].

2 The Role of Trust in Ubiquitous Computing

Many researchers have argued convincingly for the relevance of trust management in distributed systems security (cf., Blaze, Feigenbaum et al. [2]). We recapture some of these arguments, and try to highlight

a number of properties of ubiquitous computing which make the trust management approach even more appealing, even if the limitations of the traditional technology and models must first be overcome.

2.1 The Access Control List

The unique dynamic properties of the Internet and, more generally, those envisioned for ubiquitous computing, imply that traditional theories and mechanisms for security and resource access-control are often inappropriate as they are of too static a nature. For example, traditional access control consists of a policy specifying which subjects (user identities) may access which objects (resources), e.g., a user accessing a file in a UNIX file system. Apart from inflexibility and lack of expressive power, this approach assumes that resources are only accessed by a static set of known subjects (and that resources themselves are fixed); an assumption incompatible with open dynamic systems.

Many modern distributed systems use a combination of access control lists and user authentication, usually implemented via some public key authentication protocol, i.e., deciding a request to perform an action is done by authenticating the public key, effectively linking the key to a user identity, and then looking up in the access control list to see whether that identity is authorised to perform the action [2]. Furthermore, the security of current systems is often not verified, i.e., proofs of soundness of the security mechanism are lacking (e.g., statements of the form "if the system authorises an action requested by a public key, then the key is controlled by user U, and U is authorised to perform that action according to the access control list").

In Internet applications there is an extremely large set of entities making requests, and this set is in constant change as entities join and leave networks. Furthermore, even if we could reliably decide *who* signed a given request, the problem of deciding whether or not access should be granted is not obvious: Should all requests from unknown entities be denied?

Blaze et al. [2] present a number of reasons why the traditional approach to authorisation is inadequate:

- *Authorisation = Authentication + Access Control List.* Authentication deals with establishing identity. In traditional static environments, e.g., operating systems, the identity of an entity is well-known. In ubiquitous computing applications this is often not the case. This means that if an access control list (ACL) is to be used, some form of authentication must first be performed. In distributed systems, often public-key based authentication protocol are used, which usually relies on centralised and global certification authorities.

- *Delegation.* Since the global scale of ubiquitous computing implies that each entity's security policy must encompass billions of entities, delegation is necessary to obtain scalable solutions. Delegation implies that entities may rely on other (semi) trusted entities for deciding how to respond to requests. In traditional approaches either delegation is not supported, or it is supported in an inflexible manner where security policy is only specified at the last step of a delegation chain.
- *Expressive power and Flexibility.* The traditional ACL-based approach to authorisation has proved not to be sufficiently expressive (with respect to desired security policies) or extensible. The result has been that security policies are often hard-coded into the application code, i.e., using the general purpose programming language that the application is written in. This has a number of implications: changes in security policy often means rewriting and recompilation, and security reasoning becomes hard as security code is intertwined with application code (i.e., violating the principle of 'separation of concerns').
- *Locality.* The autonomy of ubiquitous computing entities means that different entities have different trust requirements and relationships with other entities. Hence, ubiquitous computing entities should be able to specify local security and trust policies, and security mechanisms should not enforce uniform and implicit policies or trusting relations.

2.2 The Traditional Trust Management Approach

In contrast to the 'access control list' approach to authorisation, trust management is naturally distributed, and consists of a unified and general approach to specifying security policies, credentials and trusting relationships, backed up by general (application-independent) algorithms to implement these policies. The trust management approach is based on programmable security policies that specify access-control restrictions and requirements in a domain-specific programming language, leading to increased flexibility and expressive power.

Given a request r signed by a key k, the question we really want to answer is the following: "Is the knowledge about key k such that the request r should be granted?" In principle, we do not care about *who* signed r, only whether or not sufficient information can be inferred to grant the request. The trust management approach does not need to resolve the actual identity (e.g., the human-being believed to be performing the request) but, instead, deals with the following question, known as the compliance-checking problem: "Does the set C of credentials prove that the request r complies with the local security policy σ?" [2, 3]. Let us elaborate: a request r can now be accompanied by a set C of *credentials*. Credentials are signed policy statements, e.g., of

the form "public key k is authorised to perform action a." Each entity that receives requests has its own security policy σ that specifies the requirements for granting and denying requests. Policies can directly authorise requests, or they can delegate to credential issuers that the entity expects have more detailed information about the requester.

Policy is separated from mechanism: a trust management *engine* takes as input a request r, a set of credentials C and a local policy σ; it outputs an authorisation decision (this could be 'yes'/'no', but also more general statements about, say, *why* a request is denied, or *what would be further needed* to grant the request). This separation of concerns supports security reasoning needed in distributed applications, which we believe to be an important feature for the ubiquitous computing challenge.

2.3 Security in Ubiquitous Computing

The above reasoning argues that traditional authorisation mechanisms are inadequate in modern distributed systems. When one considers ubiquitous computing applications, we can add further to this list. Most of the dynamic properties of ubiquitous computing entities (e.g., mobility, autonomy, ubiquity, global connectivity, ...) affect their security requirements. For example, mobility implies that an entity might find itself in a hostile environment, disconnected from its preferred security infrastructure, e.g., certification authorities. Further, the autonomy requirement means that even in this scenario, it must be able to assign privileges to other entities, privileges that are meaningful based on the usually incomplete information that the assigning entity has about the assigned entity.

- *Active decisions.* Trust management systems focus on deciding how to respond to *requests*. However, ubiquitous computing entities do not only need to respond meaningfully to requests, i.e., taking *passive* security decisions, but often need to *actively* and autonomously select among equivalent services provided by a number of apparently similar providers. Such decisions may also affect security: interaction often entails exposing personal data, as well as requiring resources like time, computation, battery and storage. When taking active decisions, there are usually no credentials available; hence, other information, e.g., reputation information, must be taken into account to make meaningful decisions.
- *Information vs. credentials.* Traditional trust management systems focus on *credentials* as the main source of proving compliance of a request with a policy. However, even when no delegation chain may establish sufficient information about a requesting entity, sometimes, collaboration may still be the most beneficial action. Notions of *risk* of an interaction, and *cost/benefit* of the

outcome of an interaction are relevant concepts that are not considered in traditional trust management policies. For example, histories, that is, memory of past interactions with an entity, may contain enough information to risk interaction. This entails that trusting relationships change dynamically, based on information about the history of an entity.

- *Probability.* Incomplete information leads naturally to probabilistic decision-making. Trust management systems that focus on information could consider probabilities explicitly (as an alternative to, or, to complement the establishing of credentials in the traditional sense). Ideally, security policies would be amenable to probabilistic yet rigorous reasoning which leads to more generality and flexibility. Additionally, as factors such as cost and benefit of interactions enters the equation, a notion of *risk* emerges as a product of cost/benefit and probability.

2.4 Computational Trust

In the arguments above, we have considered only a single notion of trust in ubiquitous computing, namely the concept of 'trust management' coined by Blaze, Feigenbaum and Lacy [3]. In fact, there are many different strands of research on trust addressing the challenges of ubiquitous computing. A whole range of trust based alternatives to existing technologies have appeared, collectively referred to as *computational trust*. For comprehensive surveys on computational trust, the reader is referred to e.g. [8, 12, 20, 19, 14].

In this paper we deal with just one particular approach within computational trust based on probability theory. Before focusing on that, we would like to comment briefly on some of the other approaches aiming specifically for a computational formalisation of the *human notion of trust*, i.e., trust as a sociological, psychological and philosophical concept (for a good survey of these, see [1]). However, the human concept of trust is elusive and its many facets make it hard to define formally [17, 5]. We believe that to live up to the ubiquitous computing challenge, it is necessary that the two concepts be merged in a 'unified' theory of trust which combines the strengths of both notions. To be more precise, our ideal would be to combine the *rigour* of traditional trust management with the *dynamics* and *flexibility* of the human notion. Let us elaborate: traditional trust management deals with credentials, policies, requests and the compliance-checking problem. Rigourous security *reasoning* is possible: the intended meaning of a trust management engine is formally specified, correctness proofs are feasible, and many security questions are effectively decidable [16]. In contrast, we have yet to see a system based on the human notion of trust which, with realistic assumptions, guarantees any sort of rigourous security property. On the other hand, such systems are capable of making *intuitively* reasonable

decisions based on information such as evidence of past behaviour, reputation, recommendations and probabilistic models. A combination of these two approaches would lead to powerful frameworks for decision-making which incorporates more general information than credentials, yet which remains tractable to rigorous reasoning.

3 Probabilistic Computational Trust

In the survey [14], computational trust is characterised as being either *credential-based* (following the ideas of Blaze et al. above) or *experience-based*. The latter term refers to approaches to trust, where an entity's trust in another is based on past behaviour, covering many so-called *reputation-based* trust management systems, which are often used in peer-to-peer (P2P) and eCommerce applications.

3.1 Experience-based Trust

Consider a set $\mathcal{P}$ of principal identities. From time to time, principals will interact in a pair-wise manner, and such interactions result in each principal observing a set of time-stamped events. In the following we make a number of simplifications, but stay general enough to capture most of the principles of existing experience-based systems: we assume that each time p interacts with another principal, say $q \in \mathcal{P}$, the interaction generates only a single event e, drawn from some set E of events (left unspecified here).

Let $(T, \leq)$ be a totally ordered set of time-stamps, e.g., $T = \{0, 1, \ldots\}$ for discrete time. Principal p records its interactions with other principals so that at each point in time, $t_0 \in T$, there is a set $\mathit{Hist}^p(t_0)$ consisting of triples (q, t, e) where $q \in \mathcal{P}$, $e \in E$; and $t \in T$ satisfies $t \leq t_0$. To be clear, a triple $(q, t, e) \in \mathit{Hist}^p(t_0)$ represents that: "In an interaction between p and q, principal p has observed event e at time t." We write $\mathit{Hist}^p_q(t)$ for the q-projection, i.e., the set of pairs (t', e) such that $(q, t', e) \in \mathit{Hist}^p(t)$.

At any point in time, $t \in T$, the sets $\mathit{Hist}^p(t)$, for $p \in \mathcal{P}$, constitute the *basic* or *direct* data of an experience-based system at time t. When p needs to make a decision at time t, e.g., about a principal q, it does so based on information from the direct data of the system at time t. Usually, such information is incomplete: while p typically knows $\mathit{Hist}^p(t)$, the sets $\mathit{Hist}^r(t)$ for $r \neq p$ may not be known exactly. This may be due to several reasons, e.g.: p may only have $\mathit{Hist}^r(t')$ for some $t' < t$; when asked about $\mathit{Hist}^r(t')$, r may lie; principal p may not be able to obtain any information about $\mathit{Hist}^r(t')$; principal p may only see some abstracted version of r's direct data; and any combinations of the above.

Most experience-based systems work on some abstracted version of the direct data, denoted $\mathit{AbsHist}^p(t)$. Some systems are centralised,

so that (abstract versions of) the direct data are stored on a global server, whereas other system are distributed. In the following we focus on models, not architectures (e.g., centralised vs distributed). Given our general model, an experience-based system is designed by *(i)* choosing if and how to abstract (or aggregate) the sets $Hist^p(t)$ to obtain the 'abstract' sets $AbsHist^p(t)$; *(ii)* choosing if and how each principal p will obtain information about $Hist^q(t)$ for $q \neq p$; *(iii)* optionally choosing how principals combine personal data with the data of others; and *(iv)* designing an architecture and algorithms (possibly distributed) to implement the system. The optional step *(iii)* often works in the following way: principal p computes for each other principal, say q, a 'score' or 'rating', $T_{pq} \in D$, for some set D of possible scores. The score T_{pq} is usually computed from some of the the abstracted versions of the direct data, i.e. $(AbsHist^r_q(t) \mid r \in I)$ for some $I \subseteq \mathcal{P}$, and represents q's trustworthiness (or reputation), seen from the point-of-view of p. Some systems have a uniform mechanism where $T_{pq} = T_{p'q}$ for all $p, p' \in \mathcal{P}$, i.e., q has a unique 'global' score.

A common example of an abstraction is the following. At time t_0, principal p is interested in information about principal q. Each record (q, t, e) is evaluated as either 'positive' or 'negative', and time is ignored; hence, $Hist^p_q(t_0)$ is abstracted to a pair consisting of the number of 'positive' interactions and the number of 'negative' interactions. Principals then obtain information about $AbsHist^r_q(t)$ by asking a central repository. Sets of records $(AbsHist^r_q(t) \mid r \in I \subseteq \mathcal{P})$ are combined into a single pair by adding-up the total number of 'negative' interactions, and similarly adding-up total number of 'positive' interactions. This example system is much like the eBay system.

Probabilistic computational trust refers to a particular kind of an experience-based approach, which assumes a probabilistic model, say λ, for the behaviour of principals. The goal is then to predict the behaviour of principals in future interactions, given the model λ and their behaviour in past interactions. The abstractions, i.e., $AbsHist(t)$, are then chosen to be as efficient as possible while preserving as much information as is relevant with respect to the model. For example, λ may specify that each principal (intrinsically) is either 'good' or 'bad', and that interaction with 'good' principals always results in event e, whereas interaction with 'bad' principals always results in event f. In this model, one only needs to interact with a principal once to know if he is a 'good' type or 'bad' type. Hence, the sets $AbsHist^p_q(t)$ need only have three values to preserve sufficient information: 'good', 'bad' or 'unknown.'

3.2 Two Probabilistic Models

Despotovic et al. [6, 7] propose a probabilistic system and an estimation algorithm based on *maximum likelihood*. It is assumed that peers inter-

act with each other in a binary way: in each interaction they can either be 'honest' or 'cheat.' Furthermore, peers can report to other peers on past behaviour (and they are allowed to lie in their reports).

The probabilistic model of Despotovic et al., λ_{D}, assumes that each principal $j \in \mathcal{P}$ is 'probabilistic' in the sense that there is a fixed probability $\theta_j \in [0,1]$ of peer j acting honestly in any interaction. Note, this assumes that j is always honest with probability θ_j, independently of any other information we might have (e.g., the time, the past, etc.). The parameters, θ_j, are unknown and the goal is to estimate them. Furthermore, each principal $k \in \mathcal{P}$ can report on its past interactions with j; it is assumed that k's report is also probabilistic so that the probability of observing a report $y_k \in \{0,1\}$ ('0' means cheated, '1' means honest) from principal k is given by

$$P(Y_k = y_k \mid \theta_j, l_k) = \begin{cases} l_k(1-\theta_j) + (1-l_k)\theta_j & \text{if } y_k = 1; \\ l_k\theta_j + (1-l_k)(1-\theta_j) & \text{if } y_k = 0 \end{cases}$$

where l_k (like θ_j) are fixed parameters specifying the probability of k submitting a false report. Hence for each principal j, there are two parameters that probabilistically decides its behaviour: θ_j and l_j.

Let us write $\mathit{AbsHist}_j^{\times}(t)$ for the collection of information that a particular principal has about j. In the system, this collection consists of a number of reports $((y_1,p_1),(y_2,p_2),\ldots,(y_m,p_m))$ where $y_i \in \{0,1\}$, $p_i \in \mathcal{P}$, and (y_i,p_i) means that principal p_i has filed report y_i (we do not consider here how reports are obtained). Hence, time is abstracted away and events are 'rated' in a binary fashion.

Now given $\mathit{AbsHist}_j^{\times}(t) = \mathbf{Y} = ((y_1,p_1),(y_2,p_2),\ldots,(y_m,p_m))$ of independent reports, the so-called likelihood function is:

$$L(\theta_j, l) = P(\mathbf{Y} \mid \theta_j, l, \lambda_{\mathrm{D}}) = \prod_{i=1}^{m} P(Y_i = y_i \mid \theta_j, l_{p_i})$$

(note this expression depends also on l, which is not clear from the authors' presentation [6]). Given current estimates for l and the data $\mathbf{Y}$ the goal is to estimate the behaviour of principal j, i.e., to estimate θ_j.

The system uses a maximum likelihood procedure which seeks to find a θ_j which maximises the likelihood expression. In the computation of likelihood function, estimates for the l_k are based on past interactions, but it is unspecified exactly how these are computed. The authors also present an approach based on normal distributions instead of the fixed θ_j's. Similarly, the maximum likelihood techniques are used to estimate the parameters of the normal distribution.

Jøsang et al. [11] and Mui et al. [18] were among to first to (independently) develop reputation systems based on a Bayesian probabilistic

approach with *beta priors*. In the following we recall the beta distribution and explain the underlying theoretical model for the beta-based reputation systems.

The beta family $Beta(\cdot, \times)$ is a parameterised collection of continuous probability density functions (pdfs) defined on the interval $[0, 1]$. There are two parameters $\alpha > 0$ and $\beta > 0$ that select a specific beta distribution from the family. The pdf $Beta(\alpha, \beta)$ is given by

$$f(\theta \mid \alpha, \beta) = \frac{1}{\mathbf{B}(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1} = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\int_0^1 \mathrm{d}t\, t^{\alpha-1}(1-t)^{\beta-1}}$$

where $\mathbf{B}$ is the beta function, and $\mathbf{B}(\alpha, \beta)^{-1}$ is a normalising constant. The expected value and variance are given by

$$\mathbf{E}_{f(\theta|\alpha,\beta)}(\theta) = \frac{\alpha}{\alpha+\beta}, \qquad \sigma^2_{f(\theta|\alpha,\beta)}(\theta) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

The beta distributions provide a so-called *family of conjugate prior distributions* for the family of distributions for Bernoulli trials. To explain the notion of conjugate priors, consider the general problem of estimating a parameter θ given some data x and background information I. Let H_θ be some hypothesis about parameter θ. The Bayesian approach (see the excellent book of Jaynes [9]), is to compute the posterior $P(H_\theta \mid xI)$ (i.e., the probability after seeing the data) from the prior $P(H_\theta \mid I)$ (the *a priori* probability, given only information I) and the likelihood function $H_\theta \mapsto P(x \mid H_\theta I)$, using Bayes' Theorem:

$$P(H_\theta \mid xI) = P(H_\theta \mid I) \frac{P(x \mid H_\theta I)}{P(x \mid I)}$$

Different priors $P(H_\theta \mid I)$ may make this probability more or less difficult to calculate, but certain choices of the prior lead to the the posterior $P(H_\theta \mid xI)$ having the same algebraic form as the prior. Now, a family of conjugate prior distributions for the family of distributions $H_\theta \mapsto P(x \mid H_\theta I)$ is a collection of distributions such that when the prior $P(H_\theta \mid I)$ belongs to the family, the posterior $P(H_\theta \mid xI)$ is also in that family (one might say that the family is Bayes-closed, i.e., is closed under the application of Bayes' Theorem).

3.3 Probabilistic Models and Trust

The reader may wonder what this has to do with trust and reputation. In the following we give our personal explanation of the Bayesian beta-based approach in reputation systems. The explanation is not explicitly presented in such detail in the papers describing beta-systems [11, 18, 4, 22], and, hence, the authors may have different perspectives.

Consider again sequences of independent experiments with binary outcomes, each yielding one of the outcomes with some fixed probability (i.e., Bernoulli trials). In systems where principal-interactions consists of binary outcomes (or where interactions are rated on a binary scale, e.g., 'cooperate' or 'defect'; 'success' or 'failure'), one can model repeated interaction (or repeated ratings) as Bernoulli trials. Let us be more precise: let $p, q \in \mathcal{P}$ be principals, and assume that p and q have interacted n times; that in each interaction q takes an action; and that the whole interaction is given a binary rating by p (which depends only on q's action). Let $X_i^{pq} \in \{0, 1\}$, for $i = 1, 2, \ldots, n$, be p's rating (i.e., subjective evaluation) of the ith interaction with q. Let us *assume* that principal q's behaviour is so that there is a fixed parameter such that at each interaction we have, *independently of anything we know about other interactions*, the probability θ for a 'success' and therefore probability $1 - \theta$ for 'failure.' This gives us a probabilistic model, and let us call it the *beta model*. Note, this is like the model λ_{D} of Despotovic et al., except for the parameters l_k for $k \in \mathcal{P}$. Let $\lambda_{\mathbf{B}}$ denote a formal proposition representing the beta model, i.e., the assumptions about the behaviour of q; also, let $\theta \in [0, 1]$ be the parameter determining success in the ith trial. Finally, let $\mathbf{X}$ be the conjunction of statements Z_i of the form

$$Z_i \equiv (X_i^{pq} = 0) \text{ or } Z_i \equiv (X_i^{pq} = 1),$$

so $\mathbf{X} = \wedge_{i=1}^{n} Z_i$, and let there be f statements of the first form and s statements of the second form (there is one statement for each i, so $s + f = n$). Then, by definition of our model $\lambda_{\mathbf{B}}$, we have the following likelihood.

$$P(\mathbf{X} \mid \theta\lambda_{\mathbf{B}}) = \prod_{i=1}^{n} P(Z_i \mid \theta\lambda_{\mathbf{B}}) = \theta^s(1-\theta)^f$$

Hence, we can obtain the posterior pdf as

$$\begin{aligned} g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}}) &= g(\theta \mid \lambda_{\mathbf{B}}) \frac{P(\mathbf{X} \mid \theta\lambda_{\mathbf{B}})}{P(\mathbf{X} \mid \lambda_{\mathbf{B}})} \\ &= g(\theta \mid \lambda_{\mathbf{B}}) \frac{\theta^s(1-\theta)^f}{\int_0^1 \mathrm{d}\theta\, P(\mathbf{X} \mid \lambda_{\mathbf{B}}\theta) g(\theta \mid \lambda_{\mathbf{B}})} \\ &= g(\theta \mid \lambda_{\mathbf{B}}) \frac{\theta^s(1-\theta)^f}{\int_0^1 \mathrm{d}\theta\, \theta^s(1-\theta)^f g(\theta \mid \lambda_{\mathbf{B}})} \end{aligned}$$

(where $g(\theta \mid \lambda_{\mathbf{B}})$ is the prior pdf for θ —cf. Jaynes [9]). If we postulate the prior pdf $g(\theta \mid \lambda_{\mathbf{B}})$ to be $Beta(\theta \mid \alpha_0, \beta_0)$ (which in particular is the uniform distribution when $\alpha_0 = \beta_0 = 1$), then we can compute the

posterior:

$$g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}}) = g(\theta \mid \lambda_{\mathbf{B}}) \frac{\theta^s (1-\theta)^f}{\int_0^1 \mathrm{d}\theta\, \theta^s (1-\theta)^f g(\theta \mid \lambda_{\mathbf{B}})}$$

Since the normalizing constant in $g(\theta \mid \lambda_{\mathbf{B}})$ cancels out, we obtain

$$\begin{aligned} g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}}) &= \frac{\theta^{\alpha_0+s-1}(1-\theta)^{\beta_0+f-1}}{\int_0^1 \mathrm{d}\theta\, \theta^{s+\alpha_0-1}(1-\theta)^{f+\beta_0-1}} \\ &= \frac{1}{\mathbf{B}(\alpha_0+s, \beta_0+f)} \theta^{\alpha_0+s-1}(1-\theta)^{\beta_0+f-1} \end{aligned}$$

which means that $g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}})$ is $Beta(\theta \mid \alpha_0+s, \beta_0+f)$.

Now let $Z_{n+1} \equiv (X_{n+1}^{pq} = 1)$, i.e., the statement that the $(n+1)$st interaction is rated as a 'success', then $P(Z_{n+1} \mid \mathbf{X}\lambda_{\mathbf{B}})$ is a predictive probability: given no direct knowledge of θ, but only past evidence ($\mathbf{X}$) and the model ($\lambda_{\mathbf{B}}$), then $P(Z_{n+1} \mid \mathbf{X}\lambda_{\mathbf{B}})$ is the probability that the next interaction will be a 'good' one. We can compute it as follows.

$$\begin{aligned} P(Z_{n+1} \mid \mathbf{X}\lambda_{\mathbf{B}}) &= \int_0^1 \mathrm{d}\theta\, P(Z_{n+1} \mid \mathbf{X}\lambda_{\mathbf{B}}\theta) g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}}) \\ &= \int_0^1 \mathrm{d}\theta\, \theta g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}}) \\ &= \mathbf{E}_{g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}})}(\theta) \end{aligned}$$

Now recall the expectation of beta distributions; then

$$P(Z_{n+1} \mid \mathbf{X}\lambda_{\mathbf{B}}) = \mathbf{E}_{g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}})}(\theta) = \frac{\alpha_0+s}{\alpha_0+s+\beta_0+f} \tag{1}$$

since $g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}})$ is $Beta(\theta \mid \alpha_0+s, \beta_0+f)$.

To summarise, given the assumptions of the beta model, one can compute the probability of a success in the next interaction as the expectation of the beta pdf $g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}})$ which results via Bayesian updating given the past history $\mathbf{X}$. Hence, the beta based systems (that deploy the technique we have described here) are mathematically well-founded on probability theory.

Jøsang et al. [11], Mui et al. [18], Buchegger et al. [4], Jennings et al. [22] all present systems based on the beta model. Buchegger et al. and Jennings et al. also propose mechanisms for dealing with lying reputation sources. Technically, all the systems work by maintaining the two parameters (α, β) of the current pdf $g(\theta \mid \mathbf{X}\lambda_{\mathbf{B}})$. However, the systems (except for [18] and [22]) deviate from the model $\lambda_{\mathbf{B}}$ in the following sense: the parameters (α, β) are adjusted as time passes; for

instance, Jøsang uses exponential decay where α and β are multiplied by a constant $0 < u < 1$ each time parameters are updated (or a fixed time limit is exceeded). The intuition is that somehow information about more recent interactions should we considered more important than information about older interactions.

Several models are based on a notion of 'belief theory' which is related to probability theory: Yu et al. developed a distributed reputation system [23], and Jøsang developed the subjective logic of opinions [10]. Indeed, the subjective logic is closely linked to the probabilistic beta model [10].

4 Towards Formal Computational Trust

Sabater and Sierra argue in [20] that the field of computational trust is lacking a more formal foundation, including a way of comparing the qualities of the many proposed trust-based systems. Sabater and Sierra propose that our field develop "(...) test-beds and frameworks to evaluate and compare the models under a set of representative and common conditions" [20].

We fully agree with these views. Also, as mentioned earlier, we believe that computational trust needs new notions of the correctness of systems, as well as frameworks for talking about the robustness of systems relative to their environments.

As an example, consider the issue of correctness of the maximum likelihood algorithm of Despotovic et al. introduced above. The traditional notion of correctness of algorithms would require a proof that the algorithm is correct with respect to its specification, i.e., that it indeed computes the maximum likelihood as specified above. However, in the setting of ubiquitous computing, we are more interested in the question of how well the algorithm approximates θ in the chosen probabilistic model. In the following, we introduce a particular framework for formalising such questions: a generic measure of how well an algorithm approximates the behaviour of entities in a given probabilistic model. And this framework is then applied in order to compare the performance of the two probabilistic algorithms from above: the maximum likelihood algorithm of Despotovic et al. and the the beta-based algorithm of Mui et al.

Our generic measure is intended to 'score' specific probabilistic trust-based systems in a particular environment (i.e., "a set of representative and common conditions"). The score, which is based on the so-called Kullback-Leibler divergence, is a measure of how well an algorithm approximates the 'true' probabilistic behavior of principals.

Consider a probabilistic model of principal behavior, say λ. We consider only the behavior of a single fixed principal p, and we consider only algorithms that attempt to solve the following problem: suppose we are given an interaction history $\mathbf{X} = [(x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)]$

obtained by interacting n times with principal p, observing outcome x_i at time t_i. Suppose also that there are m possible outcomes $(y_1, \ldots, y_m)$ for each interaction. The goal of a probabilistic trust-based algorithm, say $\mathcal{A}$, is to approximate a distribution on the outcomes $(y_1, \ldots, y_m)$ given this history $\mathbf{X}$. That is, $\mathcal{A}$ satisfies:

$$\mathcal{A}(y_i \mid \mathbf{X}) \in [0,1] \text{ (for all } i), \qquad \sum_{i=1}^{m} \mathcal{A}(y_i \mid \mathbf{X}) = 1.$$

We assume that the probabilistic model, λ, defines the following probabilities: $P(y_i \mid \mathbf{X}\lambda)$, i.e., the probability of "y_i in the next interaction given a past history of $\mathbf{X}$" and $P(\mathbf{X} \mid \lambda)$, i.e., the "*a priori* probability of observing sequence $\mathbf{X}$ in the model."

Therefore $(P(y_i \mid \mathbf{X}\lambda) \mid i = 1, 2, \ldots, m)$ defines the 'true' distribution on outcomes for the next interaction (according to the model); in contrast, $(\mathcal{A}(y_i \mid \mathbf{X}) \mid i = 1, 2, \ldots, m)$ attempts to approximate this distribution. The Kullback-Leibler divergence [15], which is closely related to Shannon entropy, is a measure of the distance from a true distribution to an approximation of that distribution. The Kullback-Leibler divergence from distribution $\hat{p} = (p_1, p_2, \ldots, p_m)$ to distribution $\hat{q} = (q_1, q_2, \ldots, q_m)$ on a finite set of m outcomes, is given by

$$\mathrm{D}_{\mathrm{KL}}(\hat{p} \mid\mid \hat{q}) = \sum_{i=1}^{m} p_i \log_2 \left(\frac{p_i}{q_i} \right)$$

(any log-base could be used). The Kullback-Leibler divergence is almost a distance (in the mathematical sense), but the symmetry property fails. That is D_{KL} satisfies $\mathrm{D}_{\mathrm{KL}}(\hat{p} \mid\mid \hat{q}) \geq 0$ and $\mathrm{D}_{\mathrm{KL}}(\hat{p} \mid\mid \hat{q}) = 0$ only if $\hat{p} = \hat{q}$. The asymmetry comes from considering one distribution as 'true' and the other as approximating.

For each n let $\mathbf{O}^n$ denote the set of interaction histories of length n. Let us define, for each n, the nth *expected Kullback-Leibler divergence from λ to $\mathcal{A}$*:

$$\mathrm{D}^n_{\mathrm{KL}}(\lambda \mid\mid \mathcal{A}) \stackrel{(\mathrm{def})}{=} \sum_{\mathbf{X} \in \mathbf{O}^n} P(\mathbf{X} \mid \lambda) \mathrm{D}_{\mathrm{KL}}(P(\cdot \mid \mathbf{X}\lambda) \mid\mid \mathcal{A}(\cdot \mid \mathbf{X})),$$

that is,

$$\mathrm{D}^n_{\mathrm{KL}}(\lambda \mid\mid \mathcal{A}) = \sum_{\mathbf{X} \in \mathbf{O}^n} P(\mathbf{X} \mid \lambda) \left(\sum_{i=1}^{m} P(y_i \mid \mathbf{X}\lambda) \log_2 \left(\frac{P(y_i \mid \mathbf{X}\lambda)}{\mathcal{A}(y_i \mid \mathbf{X})} \right) \right).$$

Note that, for each input sequence $\mathbf{X} \in \mathbf{O}^n$ to the algorithm, we evaluate its performance as $\mathrm{D}_{\mathrm{KL}}(P(\cdot \mid \mathbf{X}\lambda) \mid\mid \mathcal{A}(\cdot \mid \mathbf{X}))$; however, we accept that some algorithms may perform poorly on very unlikely

training sequences, $\mathbf{X}$. Hence, we weigh the penalty on input $\mathbf{X}$, i.e., $\mathrm{D}_{\mathrm{KL}}(P(\cdot \mid \mathbf{X}\lambda) \mid\mid \mathcal{A}(\cdot \mid \mathbf{X}))$, with the intrinsic probability of sequence $\mathbf{X}$; that is, we compute the *expected* Kullback-Leibler divergence.

Due to the relation to Shannon's Information Theory, one can interpret $\mathrm{D}^n_{\mathrm{KL}}(\lambda \mid\mid \mathcal{A})$ quantitatively as the expected number of bits of information one would gain if one would know the true distribution instead of $\mathcal{A}$'s approximation on n-length training sequences.

4.1 An example.

As an example of the applicability of our measure, we compare the beta-based algorithm of Mui et al. [18] with the maximum-likelihood algorithm of Despotovic et al. [6] introduced above. We can compare these because they both deploy the same fundamental assumptions:

> *Assume* that the behavior of each principal is so that there is a fixed parameter such that at each interaction we have, *independently of anything we know about other interactions*, the probability θ for a 'success' and therefore probability $1 - \theta$ for 'failure.'

This gives us the *beta model*, $\lambda_{\mathbf{B}}$. Let s stand for 'success' and f stand for 'failure,' and let $\mathbf{X} \in \{s, f\}^n$ for some $n > 0$.

We have the following likelihood for any $\mathbf{X} \in \{s, f\}^n$:

$$P(\mathbf{X} \mid \lambda_{\mathbf{B}}\theta) = \theta^{\mathrm{N}_s(\mathbf{X})}(1-\theta)^{\mathrm{N}_f(\mathbf{X})},$$

where $\mathrm{N}_x(\mathbf{X})$ denotes the number of x occurrences in $\mathbf{X}$.

Let $\mathcal{M}$ denote the algorithm of Mui et al., and let $\mathcal{D}$ denote the algorithm of Despotovic et al. Then,

$$\mathcal{M}(s \mid \mathbf{X}) = \frac{\mathrm{N}_s(\mathbf{X}) + 1}{n+2} \quad \text{and} \quad \mathcal{M}(f \mid \mathbf{X}) = \frac{\mathrm{N}_f(\mathbf{X}) + 1}{n+2},$$

and it is easy to show that:

$$\mathcal{D}(s \mid \mathbf{X}) = \frac{\mathrm{N}_s(\mathbf{X})}{n} \quad \text{and} \quad \mathcal{D}(f \mid \mathbf{X}) = \frac{\mathrm{N}_f(\mathbf{X})}{n}.$$

For each choice of $\theta \in [0, 1]$, and each choice of training-sequence length, we can compare the two algorithms by computing and comparing $\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{M})$ and $\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{D})$. For example:

Theorem 1. *If $\theta = 0$ or $\theta = 1$ then for all n*

$$\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{D}) \quad = \quad 0 \quad < \quad \mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{M}),$$

and if $0 < \theta < 1$ then for all n

$$\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{M}) < \mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{D}) = \infty.$$

PROOF. Assume that $\theta = 0$, and let $n > 0$. The only sequence of length n with non-zero probability is f^n, and we have $\mathcal{D}(f \mid f^n) = 1$; in contrast, $\mathcal{M}(f \mid f^n) = (n+1)/(n+2)$, and $\mathcal{M}(s \mid f^n) = 1/(n+2)$. Since $P(s \mid f^n \lambda_{\mathbf{B}}\theta) = \theta = 0 = \mathcal{D}(s \mid f^n)$ and $P(f \mid f^n \lambda_{\mathbf{B}}\theta) = 1 - \theta = 1 = \mathcal{D}(f \mid f^n)$, we have

$$\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{D}) = 0.$$

Since $\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{M}) > 0$ we are done. The argument for $\theta = 1$ is similar. For the case $0 < \theta < 1$ the result follows from similar arguments, and the fact that $\mathcal{D}$ assigns probability 0 to s on input f^k (for all $k \geq 1$), which results in $\mathrm{D}^n_{\mathrm{KL}}(\lambda \mid\mid \mathcal{D}) = \infty$. □

The degenerate property of algorithm $\mathcal{D}$ of Despotovic et al. stated in the Theorem above when $0 < \theta < 1$ follows from a general property of the Kullback-Leibler measure: given two distribution $\hat{p} = (p_1, \ldots, p_n)$ and $\hat{q} = (q_1, \ldots, q_n)$, if one of the 'real' probabilities, p_i is non-zero and the corresponding 'approximating' probability q_i is zero, then we have $\mathrm{D}_{\mathrm{KL}}(\hat{p} \mid\mid \hat{q}) = \infty$. To obtain stronger and more informative results, we shall consider a continuum of algorithms, denoted $\mathcal{A}_\epsilon$ for a real number $\epsilon > 0$, defined as

$$\mathcal{A}_\epsilon(s \mid \mathbf{X}) = \frac{\mathrm{N}_s(\mathbf{X}) + \epsilon}{n + 2\epsilon} \quad \text{and} \quad \mathcal{A}_\epsilon(f \mid \mathbf{X}) = \frac{\mathrm{N}_f(\mathbf{X}) + \epsilon}{n + 2\epsilon}.$$

Note that one can think of $\mathcal{A}_\epsilon$ as approximating $\mathcal{D}$ (which is $\mathcal{A}_0$) for small values of ϵ.

As an illustration of the type of comparison results one may obtain, we are going to show just two results formally addressing the question of how best to choose between this continuum of algorithms. The first theorem states that unless behaviour is completely random, for any fixed θ there exists one particular algorithm, which out-performes all other $\mathcal{A}_\epsilon$ algorithms.

Theorem 2. *For all $\theta \in [0,1], \theta \neq \frac{1}{2}, \mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{A}_\epsilon)$ assumes a minimal value compared to all $\mathcal{A}_\epsilon$ algorithms ($\epsilon > 0$) for*

$$\epsilon = \frac{2\theta(1-\theta)}{(2\theta - 1)^2}$$

PROOF. Follows form standard analysis of $\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{A}_\epsilon)$ as a function of ϵ. □

Correspondingly, the following theorem states that for a fixed $\epsilon_0 > 0$, algorithm $\mathcal{A}_{\epsilon_0}$ is the optimal choice amongst all $\mathcal{A}_\epsilon$ algorithms for precisely two values of θ.

Theorem 3. *For any fixed real number ϵ_0, for all $\epsilon > 0$,*

$$\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{A}_{\epsilon_0}) \leq \mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{A}_{\epsilon}) \quad \textit{iff} \quad \theta = \frac{1}{2} \pm \frac{1}{2\sqrt{2\epsilon_0 + 1}}$$

Proof. Follows from the analysis in the proof of Theorem 2. □

As an illustrative corollary of this theorem, we see that the algorithm $\mathcal{M}$ of Mui et al. is the optimal choice for precisely $\theta = \frac{1}{2} \pm \frac{1}{\sqrt{12}}$.

In fact, it is not so much the concrete comparison of algorithms $\mathcal{M}$ and $\mathcal{D}$ that interests us; rather, our message is that using probabilistic models *enables* such theoretical comparisons. We have focused here purely on the application of our framework in comparing two specific algorithms from the literature of computational trust, but it is clear to us that the framework in general opens up the *possibility* of formalising many other relevant questions. As an example, the question of robustness of say algorithm $\mathcal{M}$ can be asked in terms of how $\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{M})$ varies with θ, and investigate by a standard analysis techniques on $\mathrm{D}^n_{\mathrm{KL}}(\lambda_{\mathbf{B}}\theta \mid\mid \mathcal{M})$ as a function of θ.

5 Concluding remarks

In this paper we retraced some of the fundamental steps in the development of the concept of computational trust. We illustrated how the analogy with the common notion of trust leads to a new powerful computational paradigm for decision-making in the absence of complete and reliable information, and explained why this has a potentially major impact in several important application fields, and in particular on the emerging ubiquitous computing infrastructure. We focussed on the most prominent probabilistic approaches to computational trust, and showed how they can be expressed formally in terms of basic concepts from probability theory. Indeed, the experimental evidence in favour of such techniques is compelling; we argued however that as a community we do not yet possess a sufficient understanding of how, when and why a particular approach is superior, or at least fit for purpose, and we presented a formal framework in which such questions can be asked. We exercised our framework on comparing probabilistic trust algorithms 'quality-wise,' and obtained some new insights; yet, our main message is that a formal framework empowers us to formulate and investigate all sort of new questions, in particular those concerned with quantitative and relative measures of correctness and robustness for probabilistic computational trust systems.

References

[1] A. Abdul-Rahman. *A Framework for Decentralised Trust Reasoning.* PhD thesis, University of London, Department of Computer Science, University College London, England, 2005.

[2] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1603 of *Lecture Notes in Computer Science*, pages 185–210. Springer, 1999.

[3] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings from the 17th Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, 1996.

[4] S. Buchegger and J.-Y. Le Boudec. A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks. In *P2PEcon 2004*, 2004.

[5] V. Cahill, E. Gray, J.-M. Seigneur, C. D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C., English, W. Wagealla, S. Terzis, P. Nixon, G. M. Serugendo, C. Bryce, M. Carbone, K., Krukow, and M. Nielsen, Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3):52–61, 2003.

[6] Z. Despotovic and K. Aberer. A probabilistic approach to predict peers' performance in P2P networks. In *Proceedings from the Eighth International Workshop on Cooperative Information Agents (CIA 2004)*, volume 3191 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2004.

[7] Z. Despotovic and K. Aberer. P2P reputation management: Probabilistic estimation vs. social networks. *Computer Networks*, 60(4):485–500, 2006.

[8] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, 3(4), 2000.

[9] E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[10] A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, 2001.

[11] A. Jøsang and R. Ismail. The beta reputation system. In *Proceedings from the 15th Bled Conference on Electronic Commerce*, 2002.

[12] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. In *Decision Support Systems* **43**(2), 618–644. Elsevier Science, 2006.

[13] K. Krukow. *Towards a Theory of Trust for the Global Ubiquitous Computer*. PhD thesis, University of Aarhus, Denmark, 2006. Available online: `http://www.brics.dk/~krukow`.

[14] K. Krukow, M. Nielsen, and V. Sassone. Trust models in ubiquitous computing. Philosophical Transactions of the Royal Society A, 366(1881), 3781-93, 2008.

[15] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

[16] N. Li, J. C. Mitchell, and W. H. Winsborough. Beyond proof-of-compliance: Security analysis in trust management. *Journal of the ACM*, 52(3):474–514, 2005.

[17] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Computer Science and Mathematics, University of Stirling, 1994.

[18] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation (for ebusinesses). In *Proceedings from 5th Annual Hawaii International Conference on System Sciences (HICSS'02)*, 188. IEEE, 2002.

[19] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

[20] J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.

[21] V. Sassone, K. Krukow, and M. Nielsen. Towards a formal framework for computational trust. In *Proceedings from 5th International Symposium on Formal Methods for Components and Objects*, volume 2562 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 2007.

[22] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 997–1004, ACM Press, 2005.

[23] B. Yu and M. P. Singh. An evidential model of distributed reputation management. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 294–301, ACM Press, 2002.