Wireless Networks 10, 711–721, 2004


# Trust for Ubiquitous, Transparent Collaboration

BRIAN SHAND, NATHAN DIMMOCK * and JEAN BACON
*University of Cambridge Computer Laboratory, J.J. Thomson Avenue, Cambridge CB3 0FD, UK*

**Abstract.** In this paper, trust-based recommendations control the exchange of personal information between handheld computers. Combined with explicit risk analysis, this enables unobtrusive information exchange, while limiting access to confidential information. The same model can be applied to a wide range of mobile computing tasks, such as managing personal address books and electronic diaries, to automatically provide an appropriate level of security. Recommendations add structure to the information, by associating categories with data and with each other, with degrees of trust belief and disbelief. Since categories also in turn confer privileges and restrict actions, they are analogous to rôles in a Rôle-Based Access Control system, while principals represent their trust policies in recommendations. Participants first compute their trust in information, by combining their own trust assumptions with others' policies. Recommendations are thus linked together to compute a considered, local trust assessment. Actions are then moderated by a risk assessment, which weighs up costs and benefits, including the cost of the user's time, before deciding whether to allow or forbid the information exchange, or ask for help. By unifying trust assessments and access control, participants can take calculated risks to automatically yet safely share their personal information.

**Keywords:** trust and risk, ad-hoc collaboration, unobtrusive security, ubiquitous computing

## 1. Introduction

In this paper, we present a trust and risk framework, to facilitate secure collaboration in ubiquitous and pervasive computer systems, while minimising the need for human intervention.

Ubiquitous computing needs trust between participants in order to support collaborative activities, such as arranging meetings, while protecting sensitive information used in the collaboration. At the same time, security measures must be proportional to the risk involved to allow the interaction between devices to be as automated as possible.

For example, consider a business meeting with representatives from two companies. To schedule a follow-up meeting, the attendees would like to find a time that suits everyone, with the help of electronic diaries and calendars. However, depending on the trust between the companies, they might not want to disclose their detailed movements to each other.

Instead, the members of each company might decide to find potential meeting times among themselves, then share only this aggregate information between the companies. This paper proposes trust and risk models to help automate interactions of this sort, making the computations as unobtrusive as possible while still respecting participants' trust beliefs.

## 2. Trust infrastructure

Mutual trust is crucial for ubiquitous devices, which must share information and work together to present an unobtrusive interface [7] to their users.

Our trust framework uses a homogeneous recommendation system, to allow users to share and exchange privileged information. This information can include conventional data such as personal contacts and calendar entries, and also trust beliefs about principals.

For example, Alice might give a telephone number to Bob, together with recommendations that it is her telephone number and that it be considered privileged business information. This is illustrated in figure 1(a). Alice signs $i$ to certify that she is the origin of the information and also signs her recommendations to allow the recipient to evaluate their relevance using Alice's trust-rating. $t$ represents Alice's *trust* in her recommendation, that is, how much confidence she has in it. Later, Bob forwards Alice's number to Charlie (figure 1(b)), along with her recommendation that it is her number ($R$) and Bob's recommendation that it is her "work" number ($R''$) in which he has trust $t''$. He could also forward Alice's original recommendation $R'$ that it her "business" number if he wished to, but he has chosen not to in this case as the transmission link is expensive and he thinks Charlie will find his recommendation more useful.

Existing trust models for pervasive computing typically represent trust using a security policy which explicitly permits or prohibits actions [6]. These policies are not well suited to dynamic environments, in which participants have only partial trustworthiness, and trust assessments must constantly change. To avoid this, Abdul-Rahman and Hailes [2] have also proposed explicit recommendation systems, but with only very simple trust values. In our work, we use recommendations to control the flow of information, as well as for access control; we are also able to combine our more complex recommendations consistently, by formally ordering recommendations according to information content [5]. This gives us a well-founded approach to trust management decisions, which is suitable for distributed computing applications.

* Corresponding author.
E-mail: nathan.dimmock@cl.cam.ac.uk

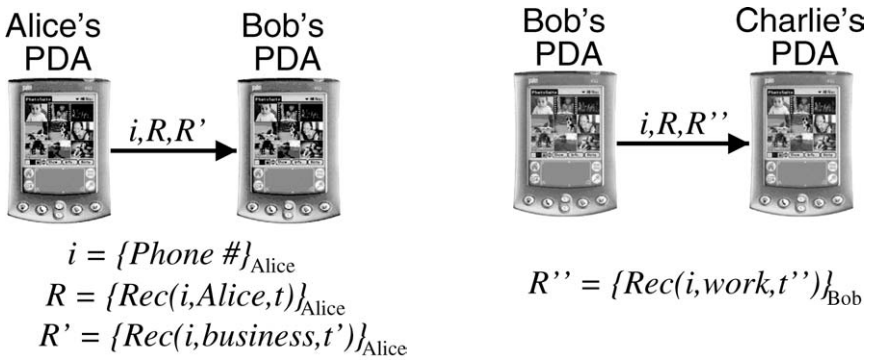


Figure 1. Recommendations in action. (a) Alice sends Bob her work phone number. (b) Some time later, Bob forwards Alice's number to Charlie.

Principals in our framework can also be associated with categories using the recommendation system, and being a member of a category may confer certain access control capabilities. Bob might send a recommendation $\{Alice, work, t_1\}_{Bob}$ to himself, that recommends Alice as a member of category "work" with trust $t_1$.

Principals and information are thus associated with categories using the recommendation system. Each item might have more than one recommendation, whether from different principals or for different categories. The trust model assesses the importance of an item (with respect to a category) by combining all the pertinent recommendations.

In the example above, the importance of displaying Alice's telephone number in Charlie's work category would depend on the degree to which Alice recommended the number, Bob recommended the number as a "work" number and Charlie's trust in Bob as a business acquaintance. Furthermore, Bob would not pass on the number automatically to Charlie; his PDA only sends the number because it knows Charlie is a trusted business acquaintance.

In order for a PDA to make these decisions automatically on behalf of its owner, it must understand the dynamics of how the owner themselves would make the decision – usually by assessing the trust and risk in the current context. In [4] we describe a more general framework for trust and risk driven decision-making; here we shall concentrate on a prototype addressbook application for a PDA.

The use of categories to assign access privileges is discussed in more detail in section 3. First, the following section extends the example to show how recommendations give structure to data.

### 2.1. Phone book example

A phone book exchange service illustrates the need for and advantages of trust-based information exchange for ubiquitous computing. Users of handheld computers currently exchange contact details laboriously on a one-to-one basis. Furthermore, there is no associated trust information, so users cannot recommend to whom the information should be redistributed – for example, private and business numbers are usually redistributed together.

In this section, we show how our trust and risk framework can make this service more transparent for users and increase automation while preserving the privacy of personal information.

The phone book database consists of many items, each with associated recommendations. These may be signed to prove their authenticity, using a public key infrastructure.

*Accessing and displaying information*

Each information item has a unique identity, depending on the author and a secure hash of the contents; any reference to an item uses this identity. As a result, recommendations about an item will cease to apply if the contents are changed. In the case of a phone book, these contents might be a name, a phone number or an address.

Figure 2 illustrates how Charlie uses the trust model to display Alice's phone book information in the example above. When Charlie searches his phone book for "Alice", he finds the entry for Alice's name, ranked according to the strength of its recommendations. If he views that entry, he is presented with the linked information too, again weighted by importance. Very unimportant entries might not be displayed at all, according to a threshold set by Charlie.

In contrast, consider David, Charlie's colleague who is allowed to view business information in Charlie's phone book, but nothing personal. If David views Alice's information there, he is presented with the restricted view shown in figure 3. Furthermore, the importance of links might be different, if David had other knowledge of Alice, such as an old work number that is now out of date, as illustrated here.

### 2.2. User privacy

When Alice gives her phone number to Bob, she trusts him not to redistribute it to people she would not want to know her telephone number. However, Bob must then realise that Alice has given him her direct line number instead of the switchboard and not pass it on indiscriminately. In our example in figure 1, Alice's recommendation $R$ states that she recommends that Bob treat this as business information and not a public number.

We believe that many security systems fail because of their high administrative overhead – passwords on post-it notes at-

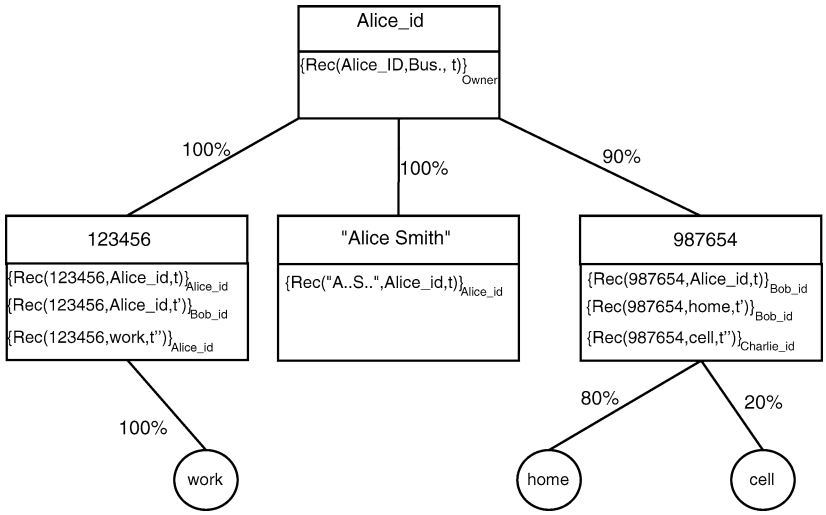


Figure 2. Each piece of information is stored separately and links between them are determined by the trust model.

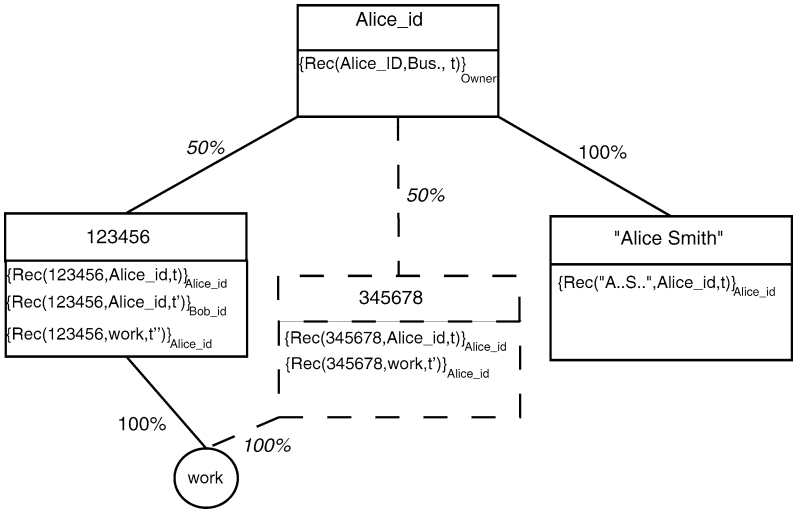


Figure 3. David receives a different view on Charlie's information about Alice, plus additional information from his own database (shown as dashed lines).

tached to monitors, for example, because proper user-account administration is considered to be too much work. We aim to create a security mechanism suitable for use in the pervasive computing environment where human intervention is a valuable resource [7], yet due to the nature of the data involved, security remains important.

The following section shows how rôles and categories can be structured to preserve the *meaning* of recommendations. This ensures that user privacy is better protected in automated information transfers, by unifying trust assessments with access control.

## 3. Categories and rôles

Information exchange is restricted with the help of categories, arranged in a partial order. These categories restrict the distribution of information, and the actions of principals, and are analogous to rôles in a Rôle-Based Access Control system [3].

We extend traditional RBAC rôles by associating a trust assessment with each category assignment. Users of the system can then combine a risk assessment, together with their trust in the information, to decide whether or not it should be used or displayed. For example, the risk of displaying an incorrect telephone number might depend on the cost of the user's time when attempting to use it. Conversely, if the number is not displayed (or is shown as less important), the risk is that the user might not find it even though it is correct.

Each category has a list of privileges associated with it; these are action and category pairs which can be used by principals associated with the category. The overall trust assessment of an entity is thus a mapping from action and category pairs to primitive trust values. These trust assessments and the contribution of other recommendations are formally expressed as a local policy function [5], defined in section 4.3, analogously to Weeks' proposal for formalising access control system policies [11].

Categories are arranged in a natural privilege hierarchy: when category $c_0$ extends the privileges of another $c_1$, we write $c_0 \supset c_1$. Figure 4 illustrates a typical hierarchy, where the top category $\top$ contains the owner of the PDA, $c_n$ represent user defined categories such as immediate family, business colleagues, business contacts, friends and relatives. *A* are acquaintances, people known to the owner, but not categorised, and *S* are strangers.

The diagram also shows another important feature of our framework, from a human interaction perspective. We have divided the categories into bands (see table 1); these bands dictate the extra privileges granted to categories within them. This makes it far more convenient for users to manage their trust policies, simply by moving categories within their trust lattice. For example, categories in the "Group" band can conventionally recommend that members may write data to their own categories and those below them, and read data below them.

The banding of categories allows user privileges to be easily and intuitively assigned. However, if necessary the banding may be overridden, by explicitly associating extra permissions with categories or users.

Formally, these bands are a partition of the privileges of the system. For example, if $p(c)$ represents the privileges of category or band $c$, then $p(\text{Group}) = (p(c_2) \cup p(c_3) \cup p(c_4)) \setminus p(S)$, the marginal privileges accrued by categories within the band.

The category bands also have a second function: they facilitate information exchange, by providing a common framework for expressing category meaning between devices, even devices with otherwise different categories.

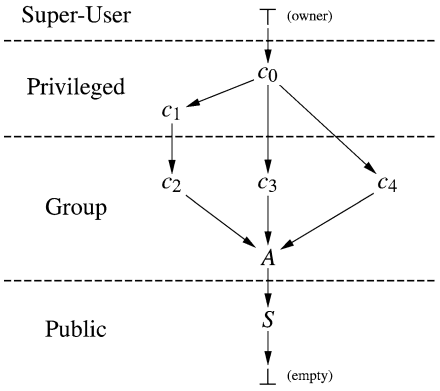


Figure 4. Example of a category hierarchy.

Table 1
Privileges are conferred on a category by membership of a *Privilege Band*.

| Band | |
|---|---|
| Super-User | As Privileged, but may also delete information. |
| ↑ | |
| Privileged | As Group, but can also read own category. |
| ↑ | |
| Group | As Public, but can also write to own category and those below. |
| ↑ | |
| Public | Read: Categories lower in graph. Write: None. |

This allows recommendations between devices to be made in terms of category bands. As long as users attribute similar meanings to these bands – encouraged by band privileges – then information transferred between devices will automatically be restricted to the appropriate band, unless there is an explicit user override. This is particularly useful in avoiding sensitive information being leaked by different collaborating users assigning different meaning to categories of the same name or placing them in different bands. For example, suppose Alice has two categories, *business* and *business contacts*, the former in the *Privileged* band and the latter in *Group* and she gives her *business* number to colleague Bob, who's *business* category is in *Group*. However, if Alice sends an additional recommendation that the number is *Privileged* then Bob's addressbook should respect this when calculating access rights.

### *3.1. Categories in calendars*

The same framework can also be used for calendar information. In the phone book we had read and write capabilities for viewing, inserting and updating phone numbers. When a principal attempts to read a particular time-slot, the information returned will depend on their location in the hierarchy of categories in relation to the category of the appointment. All appointments have a projection into categories lower in the hierarchy, although not into $\bot$. This has the effect that a principal who does not have read permission for an appointment sees the lower category projection that the time is busy, tentative or free but not the details of any appointments. Because appointments are not projected into $\bot$, principals in sufficiently low categories (such as *S* in figure 4) do not see anything at all and can learn no information about the owner's schedule.

Write permission to a category is the ability to make an appointment in that category, and categories could be used to determine the default response to an appointment made in a free slot (for example: "automatically accept all appointments made by principals who are members of category PhD-Supervisor").

## 4. Trust computation

Participants compute their trust in information, by combining their own trust assumptions with others' recommendations. This section outlines the structure of these recommendations, and the formulae which compose them together.

Although we present our framework for a particular ubiquitous computing application, we believe that its use extends to all recommendation-based systems, particularly those operating in mobile environments, where communication is limited and unreliable.

### *4.1. Recommendations*

Recommendations associate one permission with another in our trust framework. By treating the identities of actors, cat-

egories and data entries as permissions, we can use a homogeneous recommendation structure for privilege assignment and for restricting the flow of information. In practice, these permissions are associated with the public and private keys of each actor and category.

The permissions $\mathcal{P}$ linked by recommendations are the following:

**Actor permissions** $\mathcal{A}$ are used to identify people (and their aliases), such as "Alice" or "asa21".

**Category permissions** $\mathcal{C}$ represent membership of a category such as "business".

**Data entry permissions** $\mathcal{D}$ refer to address book entries, including telephone numbers and names in our first application.

**Action permissions** $\mathcal{P}_A = \{Read, Write\} \times \mathcal{C}$ allow the holder to read or write data in a particular category.

**Link permissions** $\mathcal{P}_L = \{Link\} \times (\mathcal{A} \cup \mathcal{C})$ are used when data is written, to recommend that it be associated with a category or an actor.

We limit which permissions may be linked, allowing only the combinations shown in table 2. Recommendations are then combined transitively to determine effective trust values, discounted according to the permissions the recommenders hold.

Taking the example shown in figure 5, suppose that Alice recommends that Bob should be a member of category "business" with trust $t_1$, that is $\{Rec(Bob, business, t_1)\}_{Alice}$. If the PDA's owner trusts Alice as a member of "business", then Bob will be considered a member too. Furthermore, if the owner also recommends that "business" acquaintances can read data from category "strangers", then the trust will be transferred and Bob will be allowed to read data associated with strangers too.

Each recommendation thus links one permission to another, with a certain degree of trust according to the recommender. Next, we present the structure of these trust values, before showing how they are combined to make decisions.

Table 2
Acceptable recommendations.

| From\To | $\mathcal{A}$ | $\mathcal{C}$ | $\mathcal{D}$ | $\mathcal{P}_A$ | $\mathcal{P}_L$ |
|---|---|---|---|---|---|
| $\mathcal{A}$ | ✓ | ✓ | | ✓ | |
| $\mathcal{C}$ | | | | ✓ | |
| $\mathcal{D}$ | | | | | ✓ |
| $\mathcal{P}_A$ | | | | | |
| $\mathcal{P}_L$ | | | | | |

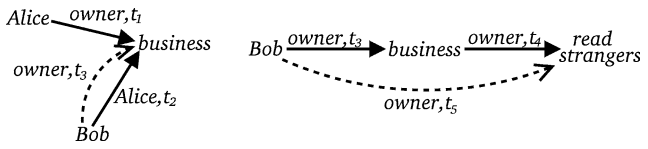


Figure 5. Recommendations in action – the dashed lines represent the derived beliefs of the PDA owner.

### 4.2. Trust values

In our framework, each trust value consists of a (*belief*, *disbelief*) pair, representing the weight of evidence for and against a particular trust assignment, with *belief* + *disbelief* $\leqslant 1$. This can be compared to Jøsang's logic of uncertain probabilities, based on the Dempster–Shafer theory of evidence [8].

No information is represented by $(0, 0)$, while $(1, 0)$ and $(0, 1)$ represent certain belief and disbelief, respectively. We order these trust values according to trustworthiness by defining $(b_1, d_1) \preceq (b_2, d_2)$ iff $b_1 \leqslant b_2$ and $d_2 \leqslant d_1$, which forms a lattice on our trust domain $T_b$.

However, there is also a second natural ordering, according to information, where $(b_1, d_1) \sqsubseteq (b_2, d_2)$ iff $b_1 \leqslant b_2$ and $d_1 \leqslant d_2$, which we will use in combining recommendations below [5].

### 4.3. Policy functions

Users must combine their own recommendations with others' to assess trust. This is achieved by forming a *policy function* for each principal's recommendations; these policy functions are then combined to reach the appropriate trust conclusions.

Each policy function denotes the trust each principal places in others' trust information; $Pol_x(T, y, z)$ is the degree to which principal $x$ believes $y$ should hold permission $z$, if everyone else's trust assignments are given in $T$.

This combines $x$'s own recommendations with recommendations by others $x$ trusts. Let $d_x(y, z)$ summarise $x$'s recommendations, with $d_x(y, z) = t$ if there is a recommendation $\{Rec(y, z, t)\}_x$, and $(0, 0)$, otherwise. (Newer recommendations are assumed to supersede older ones.)

Two sorts of recommendations are transitively combined, generalising those in figure 5:

- those where $x$ associates $y$ with $p$, and $p$ with $z$, and
- those where $x$ gives $p$ permission $z$, and $p$ recommends $y$ for $z$.

This is summed up in the policy function

$$Pol_x(T, y, z) = \bigoplus \Bigg\{ d_x(y, z) \cup \bigcup_{p \in \mathcal{P}} T(x, y, p) \otimes T(x, p, z) \cup \bigcup_{p \in \mathcal{P}} T(x, p, z) \otimes T(p, y, z) \Bigg\}, \quad (1)$$

where

$$Pol_x : \big(\mathcal{P} \to \big(\mathcal{P} \to (\mathcal{P} \to T_b)\big)\big) \to \big(\mathcal{P} \to (\mathcal{P} \to T_b)\big), \quad (2)$$

$$(b, d) \otimes (e, f) = \begin{cases} (0, 0) & \text{if } b \leqslant d, \\ \left(\dfrac{e}{k}, \dfrac{f}{k}\right) & \text{otherwise,} \end{cases}$$

$$\text{with } k = \max\left(\frac{e}{b-d}, \frac{f}{b-d}, 1\right). \quad (3)$$

We also define $\bigoplus X_i$ to combine a number of recommendations monotonically, by averaging their belief and disbelief

components, respectively. For example, given three recommendations $(0.2, 0)$, $(0.25, 0.5)$ and $(0.36, 0.4)$ as shown in figure 6, the compound recommendation deduced by $\oplus$ is $(0.27, 0.3)$.

Informally speaking, this considers the influence of the original recommendations $d$, together with the recommendation chains shown above. These are then combined to deduce an updated trust value. By repeating this process, the trust values converge to a final trust assessment.

To guarantee this convergence, each policy function $Pol_x$ must be monotone with respect to $T$, as shown in equation (4). If we then combine all the individual policy functions into a single function $Pol(T)$, this will also then be monotone, and have a least fixed point $T_f = Pol(T_f)$, which will be our considered trust assessment [5].

$$T' \geqslant T \;\Rightarrow\; Pol_x(T', y, z) \geqslant Pol_x(T, y, z), \quad \forall x, y, z \in \mathcal{P}, \tag{4}$$

$$Pol(T)(x, y, z) = Pol_x(T, y, z), \quad \forall x, y, z \in \mathcal{P}, \tag{5}$$

$$Pol : \big(\mathcal{P} \to \big(\mathcal{P} \to (\mathcal{P} \to T_b)\big)\big) \to \big(\mathcal{P} \to \big(\mathcal{P} \to (\mathcal{P} \to T_b)\big)\big). \tag{6}$$

The policy function $Pol$ given above always converges for non-cyclical recommendation sets, such as those used in our address book application. However, it is not always monotone as it stands – we need to augment it to ensure monotonicity, and hence convergence under recommendation cycles.

Therefore, in computing the trust policy, we augment each trust value $T_b$ with a list of "parent" recommendations that contributed to it. As the trust policy calculation iterates, the parent lists increase in terms of an extended information order $\sqsubseteq$, whose bottom element $\perp = (0, 0, [])$ represents no recommendations at all. The least fixed point will then correspond to the least specific trust assignments justified by the available recommendations.

We also extend the $\otimes$ and $\oplus$ operators to propagate parent lists on to the deduced recommendations, and to ignore any cyclical contributions from recommendations whose parents include the recommendation currently being computed. This ensures theoretical monotonicity, while still producing the same deduced trust values as before in the absence of cycles.

We have presented policy functions as a well-founded mechanism for deducing trust values by combining recommendations. Recommendations are combined transitively,

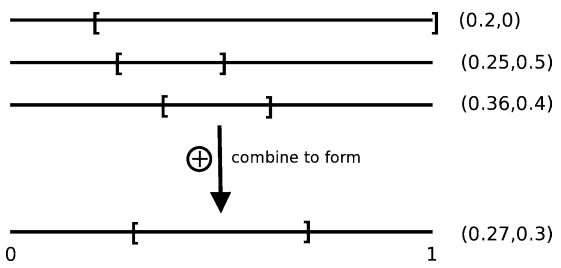


Figure 6. Combination of recommendations.

which allows certain permissions to entail others, and trusted acquaintances can delegate their permissions to others.

The resulting trust values guide decision making in our application, with the help of the risk assessments outlined in section 6. First, however, we consider implementation issues in resource-poor and vulnerably connected devices in ubiquitous computing environments.

## 5. Implementation issues

Trust management through recommendations is well suited to mobile and distributed applications, since recommendations conveniently factorise and encapsulate trust policy.

This is particularly important for vulnerably connected nodes such as PDAs, which must store the relevant components of others' policy locally, for use when disconnected. Transferring only a few recommendations from a trust policy is justified in our application, since extra recommendations correspond to additional trust information in our partial order. Therefore using a subset of a policy corresponds to weaker policy assertions, and the resulting trust decision will also be weaker.

However, locally-cached policies must be kept up to date in order to be used appropriately. We therefore propose to assign time stamps and validity periods to recommendations which are then refreshed automatically each time devices interact, to remove any burden on the owner to ensure their local cache is not about to expire before embarking on a period of extended disconnection. If a recommendation does expire, using out of date policy may be preferable to no knowledge at all and so we scale-down the weight of expired recommendations rather than discarding them. However, the principal danger of outdated information is that a person may no longer deserve the privileges that they once had, for example, someone who has been fired from the company. Therefore old trust policy that says something negative about a principal cannot cause our security to be compromised and so this scaling is only applied to expired positive recommendations.

Recommendation systems often suffer from issues of long trust chains, because the meaning of "trust" changes with depth in the chain – in PKI a principal who is trusted to recommend other good recommenders must also be trusted to be a good signer [1]. This is not a major problem in this application scenario as, in general, we believe people categorise the people they know according to the type of trust they place in them: close friends are clearly highly trusted; "business colleagues" do not try to sabotage each other's list of contacts but would not usually have access to personal numbers; and so on. People within a single category may have different levels of trust placed in them, but partial belief in category membership caters for this. When necessary, we also allow exceptions to be made; the owner of the PDA can fine tune their policies, via the recommendation system, to customize individual users' permissions. We believe that it is this modelling of human intuitions of trust (including the overloading of the meaning of the term) that makes our system so power-

ful while still being easy to use, although we observe that it is not true in the general case.

## 6. Risk assessment and decision making

As stated in the introduction, we believe security measures must be proportional and appropriate for the risk involved: a user may happily distribute a business card to strangers to advertise their business, but may be quite careful as to whom they give their mobile phone number.

In the same way that a principal's position in the category hierarchy (figure 4) assigns it a permission, the position of a piece of data implicitly gives it a value that can be used to assess the risk of an operation involving it: the higher in the hierarchy, the greater the value. We define the risk of an operation as being the sum of the risks of all the possible outcomes of that operation, where the risk of an outcome is a function of the likelihood and impact of that outcome. This is in line with existing literature on risk management, such as [10] and we take the view that the *impact* of an outcome is the worst-case cost to the user should that outcome occur. This cost will be a combination of two factors: the seriousness of the outcome itself and the value of the data involved.

In the address book scenario, two users may interact in two different ways as shown in figure 7. Either Bob may request a number from Alice, or she may try to send Bob information, unsolicited. Before either side takes part in an interaction, there is a decision to be made (shown as the numbers 1–4 in figure 7). Those decisions are as follows.

1. *Request*. Bob wishes to ask Alice's PDA for a telephone number. As far as Bob is concerned, the possible outcomes from interacting with Alice are (in increasing order of impact):
   - he obtains the number he wanted and it is correct;
   - he obtains the number he wanted but it is incorrect (e.g. out of date);
   - he does not obtain the number he wanted.
2. *Response*. Alice receives Bob's request and must decide what access to her address book she is prepared to give him. From Alice's point of view, the possible outcomes of giving Bob access to an entry in her address book are:
   - Bob obtains the number he wanted;
   - Bob obtains the number, but misinterprets or ignores the attached recommendations and redistributes it indiscriminately.

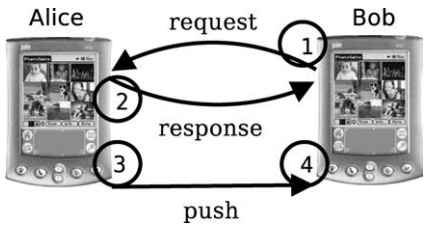


Figure 7. Possible interactions between two PDA users.

3. *Push-number*. Alice wishes to automatically send her number to certain PDAs she comes into contact with. For example she may have recently changed her home telephone number and wishes to inform all the friends she meets, but not business colleagues. Her PDA must decide whether to automatically send the number to Bob; the possible outcomes for Alice are (again in increasing order of impact):
   - Bob stores the number and respects Alice's accompanying recommendations on redistribution;
   - Bob discards the number;
   - Bob stores the number but ignores the accompanying recommendations on redistribution.
4. *Receive-number*. Alice wishes to send Bob some information. Bob must decide what to do with the received information. The possible outcomes from his point of view are:
   - Bob finds it useful;
   - Bob finds the information unhelpful or incorrect;
   - Alice attempts a denial of service attack against Bob's PDA by sending many numbers, aiming to fill its storage space or saturate its connectivity.

As stated above, the risk of an outcome is a function of the worst case cost in the event of the outcome occurring, and the probability that the outcome will occur, which is solely dependent on the principal(s) involved. Using the idea that trust is a measure of how well an actor is known, it is possible to assign a probability to each outcome.

We will now consider one trust-decision, *Response*, in more detail.

### 6.1. Deciding whether to participate

When Bob asks Alice for a number from her address book, in access control terms, she must decide whether to grant him read permission on that number or not. The aim of our model is to make this decision as automatic as possible, but in situations where the correct response is unclear the PDA may then attract Alice's attention and ask for her guidance. However, the cost of Alice's time to give that guidance must also be factored into the decision, so our cost-benefit analysis must take into account the benefit from helping someone by giving them a number, the worst-case cost of giving a number to an inappropriate person and the cost of asking the owner for guidance.

We now derive formulae for calculating the benefit of each of the three possible courses of action. Since principals and data may be filed under multiple categories in an addressbook, for example colleagues who are also considered friends, we must consider all pairs of categories, $(c_p, c_d)$, which is the principal's category and the category of the data item they wish to read respectively, where there is a recommendation (or recommendations) with $b > d$, that permit $c_p$ to read $c_d$.

There is clearly a benefit to not giving out a number if we have no trust in that person's right to that number, that is, if we do not believe them to be a member of $c_p$. We define:

$$\text{Benefit}_{no}(b-d, val_{c_p}) = -val_{c_p}.(b-d)$$

where $(b, d)$ is Alice's belief and disbelief in Bob's membership of $c_p$ and $val_{c_p}$ is the value Alice has associated with category $c_p$ – the more valuable the category and the greater Alice's distrust in Bob's membership, the greater the benefit of saying, "no".

For calculating $\text{Benefit}_{yes}$, in addition to considering how strongly Bob is associated with $c_p$ and the expected benefit of that association ($val_{c_p} \times (b-d)$) we must also consider the relative importance of Bob himself (inferred from the value of the category of which he is a member) compared to the importance of the data he is trying to read, which encodes the potential cost of Bob ignoring Alice's recommendations and redistributing the number indiscriminately. Our function must represent the fact that there is benefit in helping someone who is potentially a close friend to read a low-value number, while we may require greater assurance to allow access to a more valuable number.

It is also necessary to allow the user to configure their disposition to trust [9], and it may be useful to take any available contextual information (such as location or the status of the owner) into account. For this purpose, we introduce $val_{read}$, the importance of generally being a helpful source of information which leads us to define:

$$\begin{aligned}&\text{Benefit}_{yes}(b-d, val_{c_p}, val_{c_d})\\ &\quad = val_{c_p}.(b-d) - \max(val_{c_d} - val_{read}, 0).\end{aligned}$$

This definition balances the expected benefit of assisting Bob, $val_{c_p}.(b-d)+val_{read}$, against the value of the data being read, $val_{c_d}$. The maximum function prevents the benefit of giving out information from being greater than the expected benefit of interacting with Bob, even if the value of $val_{read}$ is greater than the value of the data involved. Section 6.2 discusses the $val_{read}$ variable further.

If a positive trust-relationship between Bob and the requested data does exist, but it is sufficiently tenuous that there is no clear benefit to granting the request, then it may be worth asking Alice for guidance on the decision. To represent the cost of Alice's time in having to focus on her PDA and input the correct decision, we introduce a second user-configurable, context-dependent variable, $val_{time}$.

$$\text{Benefit}_{ask}(b-d, val_{c_p}) = val_{c_p}.(b-d) - val_{time} + val_{read}.$$

This equation simply compares the expected benefit of helping Bob (and also the potential cost if we make an erroneous negative decision) to the cost of Alice's time that is taken up making the decision. $val_{time}$ is discussed further in section 6.2.

It follows that Alice's response to Bob's request can now be determined by:

$$\begin{aligned}\text{Answer} = {}&\texttt{if } \text{Benefit}_{no} \geqslant 0 \texttt{ then } \text{“No”}\\ &\texttt{else if } \text{Benefit}_{yes} > 0 \texttt{ then } \text{“Yes”}\\ &\texttt{else if } \text{Benefit}_{ask} > 0 \texttt{ then } \text{“Ask”}\\ &\texttt{else } \text{“No”}.\end{aligned}$$

Through these formulations we have effectively asked whether there exists $c_p$ of which Bob is a sufficiently strong member to be able to read $c_d$. For simplicity, we use $b-d$ as a measure of the strength of a principal's membership of a category, and since there are three possible responses to the request, *Yes*, *No* and *Ask owner for guidance*, we may see the range of $b-d$, the interval $[-1, 1]$, divided into three corresponding regions, as illustrated in figure 8.

The positions of $x$ and $y$ may be determined by setting $\text{Benefit}_{yes}$ and $\text{Benefit}_{ask}$ to the threshold at which we decide there is sufficient benefit to take that course of action, that is $> 0$, and re-arranging to find the corresponding value of $b-d$. This gives:

$$y = \max\left(\frac{val_{c_d} - val_{read}}{val_{c_p}}, 0\right), \tag{7}$$

$$x = \min\left(\frac{val_{time} - val_{read}}{val_{c_p}}, y\right). \tag{8}$$

The region, $[0, x)$ is when Bob is a member of $c_p$, but there is insufficient benefit in saying "yes" or asking the owner to grant the request, so the answer must be negative. The region $[-1, 0)$ is also logically a negative response as there is no trust at all in Bob's membership of $c_p$.

### 6.2. Fine-tuning policy

The variables $val_{read}$ and $val_{time}$ used in the benefit equations can be tuned by the user to give them fine-grained control over their policy and take any available contextual information into account. For example, the owner may be able to place the PDA into a "Do-Not-Disturb" mode that would scale the value of $val_{time}$ to infinity. The significance of the values of these variables on the decision making process can be seen by considering the effect they have on the thresholds, $x$ and $y$.

Figure 9 shows how the values of $c_p$ and $c_d$ effect the threshold $y$, for constant $val_{read}$. It demonstrates that our $\text{Benefit}_{yes}$ equation has the desired property that as $val_{c_d}$ increases for constant $val_{c_p}$ a greater amount of trust in membership of $c_p$ is required to grant the request, and as the right-hand back corner of the graph shows, not even a "fully" trusted principal may read a number which is of greater value than themselves. Conversely, a principal of much greater value than $val_{c_d}$ needs a much smaller amount of trust to read the number, and if $val_{read} > val_{c_d}$ then only the most tenuous connection with a category is required to be granted the privilege. This leads us to the conclusion that a good choice

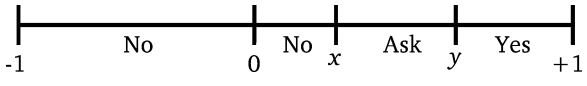


Figure 8. Number line showing how partitions of $b-d$ in membership of a category lead to a decision.

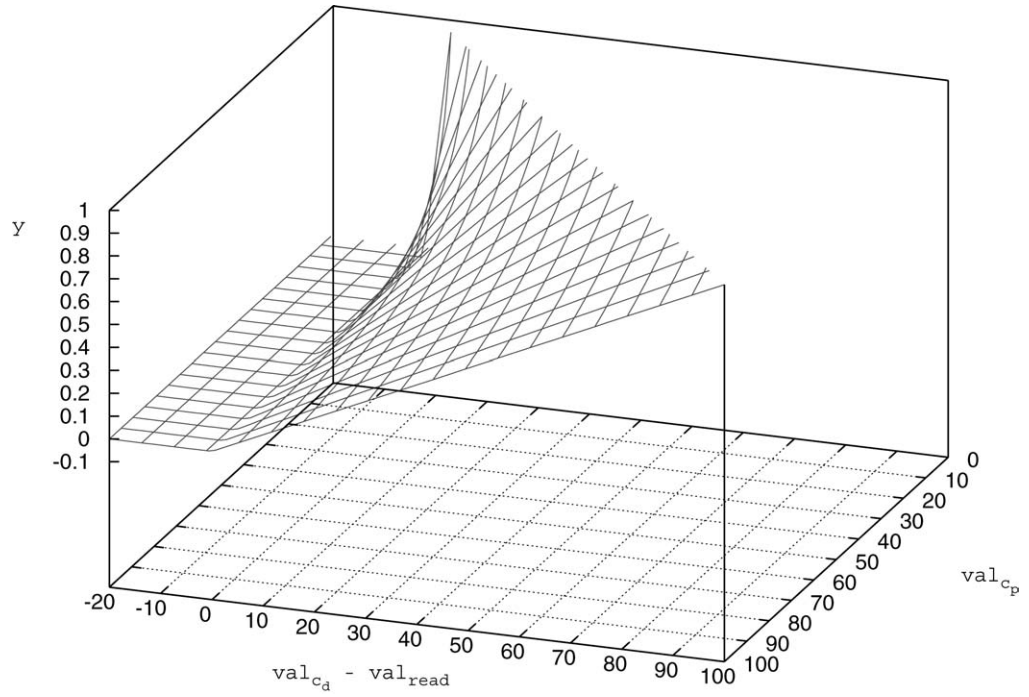


Figure 9. Plot showing how the amount of trust required, $y$, varies with the values of $val_{c_p}$ and $val_{c_d} - val_{read}$.

of value for $val_{read}$ is the value of the category for which a principal should only need the smallest amount of belief in membership in order to be able to read it.

The Benefit$_{ask}$ equation shows that the choice of $val_{time}$ is dependent on the choice of $val_{read}$ as if $val_{time} \leqslant val_{read}$ then the possible benefit of being helpful always outweighs the cost of the time involved. The result is that the PDA will always ask the user if $b \geqslant d$, unless the answer is obviously "Yes".

From equation (8) it can be seen that if $val_{time} - val_{read}$ is a constant, the amount of trust required in the membership of category $c_p$ is inversely proportional to the value of the category. Accordingly, the lower the potential value of the other principal, the greater the trust required for it to be worth bothering the PDA owner. In practical terms, this indicates a good choice of value for $val_{time}$ is the value of the lowest category with which the user wishes to be consulted when a principal with strong membership (high value of $b-d$) of that category attempts to read a number of equal value.

### *6.3. Other trusting-decisions*

We now consider the cost-benefit analysis of the other trust-decisions shown in figure 7.

#### *6.3.1. Request*

The PDA owner enters the name of the person about whom they require information and the category under which they intend to file the data. The addressbook application now computes the expected benefit of asking each person within communication range and then polls each one in turn until one returns an answer. If the PDA is unable to obtain the number then it can store the request and ask other PDAs that it encounters in the future.

The expected benefit of asking Alice for a number depends on the relationship between Alice and the number. If the owner is looking for Alice's number then she is clearly the best person to ask, but if Bob knows Alice as a friend and he is looking for the number of a colleague then she is unlikely to be able to help. In access control terms this is represented as the ability of Alice to write to the category under which we intend to file the number and to link numbers with the principal that Bob is looking for. This is an analogous decision to the one made in section 6.1 when the PDA must make a decision based on the read permissions held by the other principal.

Therefore, assuming that Bob is looking for the number belonging to someone other than Alice, the expected benefit of asking Alice for a number is:

$$\begin{aligned}&\text{Benefit}_{yes}(b-d, val_{c_p}, val_{c_d})\\ &\quad = val_{c_p}.(b-d) - \max(val_{c_d} - val_{write}, 0).\end{aligned}$$

$c_p$ is a category which has write permission on the target category $c_d$ supplied by the owner of the PDA and of which Alice is a member. $val_{c_p}$ and $val_{c_d}$ are the values of $c_p$ and $c_d$, respectively, and hence this equation is analogous to the benefit equations used to determine whether a principal may read a number in a category $c_d$. $val_{write}$, like $val_{read}$, is a tunable parameter which represents how aggressively the PDA should search for a number. This parameter is much more dynamic than $val_{read}$ – it can be influenced by the user when initiating a search via an "urgency" rating on the search screen, but it also takes environmental factors into account such as remaining battery power versus the cost of communication.

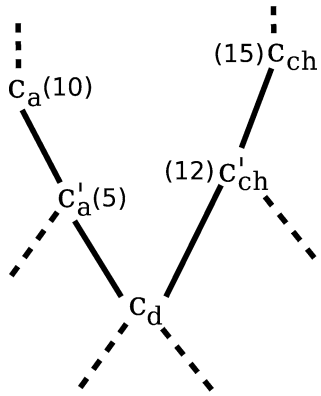


Figure 10. Part of a category hierarchy. Category values are shown in parentheses.

However, this equation does not take into account that Alice may be associated with multiple categories $c_p$, that have write permission on $c_d$ and, since membership of one category may imply membership of another, summing the benefits for all $c_p$ that can read $c_d$ would lead to an inflated result. Using the category structure shown in figure 10 as an example, suppose Alice is a member of $c_a$ with trust, $(0.7, 0.1)$ and Charlie is a member of $c'_{ch}$ with trust $(0.8, 0.2)$. Since the strength of membership $(b - d)$ of their respective categories is equal and the value of $c'_{ch}$ is greater than that of $c_a$ (12 compared to 10) there should be greater benefit in asking Charlie than Alice. However, by default all members of $c_a$ inherit the permissions of $c'_a$ so the naïve solution would give the benefit of asking Alice as 15. Simply using the category with the greatest value would also lead to errors: if some weaker recommendations also place Charlie in category $c_{ch}$ with trust $(0.2, 0.1)$, asking Alice would again be assigned a higher benefit than asking Charlie, despite the greater value of $c_{ch}$.

The solution we employ is for a particular traversal of the lattice, to consider only the category which maximises the principal's expected benefit (that is, $val_c \times (b - d)$). Since a principal may appear in multiple paths through the category lattice (the previously mentioned example of colleagues who are also friends), we define a set, $C$, of these categories. The benefit of asking Alice for the required number is therefore:

$$\sum_{c_p \in C} val_{c_p}.(b - d) - \max(val_{c_d} - val_{write}, 0).$$

Suppose Bob is searching for Charlie's number. If Charlie is within communication range, his PDA will ask Charlie's first but if Charlie is unwilling to give his number to Bob (perhaps their PDAs have yet to be "introduced") or if Charlie is not contactable then Bob's PDA contacts the other available PDAs in decreasing order of the expected benefit of asking them for the number, until the number is successfully obtained.

*6.3.2. Push-number*

It is likely that Bob wishes to distribute his details to those people who would normally be able to read it if they were to request it so it should only be pushed to those users who have *read* permission on the category in which it belongs. Therefore, for each of the principals within communication range we evaluate the $\text{Benefit}_{yes}$ function described in section 6.1, and if it is positive, the number is sent to that principal. The $val_{read}$ variable may be used to determine how aggressively the PDA pushes its number to potential receivers.

*6.3.3. Receive-number*

To avoid wasting resources, the decision as to whether to receive a number from another PDA should be taken as early as possible in the interaction. Unfortunately our current prototype implementation does not have access to the lower levels of the network stack and can only make a decision whether to accept a piece of data once it has been received so we have taken the approach that we will accept all recommendations and compute their usefulness upon demand. This also means that should we later obtain further recommendations that render the received recommendations more useful they will be treated as such, instead of possibly being discarded when they are first received. The main problem with this approach is storage constraints, a topic that, as mentioned in section 5 is still under investigation.

### *6.4. Deciding what to display*

There is one other operation where the trust-model is invoked, and that is choosing what to display to the owner of the PDA when he or she wishes to view some information. Suppose Alice wishes to view Bob's number. She searches for his name and the PDA finds ten telephone numbers that are linked to him with varying degrees of strength. Since ten numbers will not fit onto the PDA display at one time, they are displayed in an order given by the product of the strength of the trust-model's belief they belong in a category $(b - d)$, and the value of that category – the expected benefit of that number. The interface is designed to allow the user to give feedback on which number they were looking for and how successful they were at using it. This means that if Alice tries to use a number which is, for example, out of date, she can click a button next to it and the system takes this to be a recommendation from her (which is implicitly highly trusted) that this number is not Bob's and updates its trust values accordingly.

Alternatively, Alice might browse entries by address book category. These could be ordered either conventionally (alphabetically), or by the degree of category membership. Again, the interface allows feedback for incorrect entries, in the form of extra recommendations.

## 7. Conclusions

We have outlined a framework for an unobtrusive and mostly automated security model for ubiquitous devices, using a system of trust-evaluated recommendations combined with an explicit risk analysis. This model is useful for a wide range of pervasive and ubiquitous computing applications, in which the user's time is a valuable resource and transparent interaction is needed wherever possible. We have tested the model

by applying it to our prototypical examples, a phone book and an appointment diary, and we believe it is applicable to all recommendation-based systems, especially ones in mobile environments. Ongoing work includes optimising our algorithms for mobile devices and a detailed user acceptance and evaluation study of our assumptions regarding the reuse of the natural organisation of a user's address book to assign access permissions. Initial investigations in this area suggest that this is very useful from a user interface perspective, although it is not clear whether paradoxical recommendation chains can automatically be resolved consistently with human intuition.

Further work includes the detection of untrustworthy principals by examining the source(s) of information found in other people's PDAs and a general trust-based access control model for personal devices. In this way, many applications could share a common recommendations, allowing automatic and intuitive collaboration for mobile applications.

## Acknowledgements

This work has been inspired and supported by the EU-funded SECURE project (IST-2001-32486), part of the EU Global Computing initiative. The authors would like to acknowledge the very helpful interaction we have had with all the members of the project consortium, and especially BRICS, at Århus, Denmark, for helping to formally ground our trust model.

## References

[1] A. Abdul-Rahman, Problems with trusting recommenders to recommend arbitrarily deep chains (March 1998), available at `http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/levnprob.html`

[2] A. Abdul-Rahman and S. Hailes, Supporting trust in virtual communities. in: *Proceedings of the 33th Hawaii International Conference on System Sciences* (IEEE, 2000) pp. 1769–1777.

[3] J. Bacon, K. Moody and W. Yao, Access control and trust in the use of widely distributed services, in: *Proceedings of Middleware 2001*, Lecture Notes in Computer Science, Vol. 2218 (Springer, 2001) pp. 295–310.

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

[5] M. Carbone, M. Nielsen and V. Sassone, A formal model for trust in dynamic networks, Research Series RS-03-04, BRICS, Department of Computer Science, University of Aarhus, EU Project SECURE IST-2001-32486 Deliverable 1.1 (January 2003).

[6] T. Finin, A. Joshi, L. Kagal, O. Ratsimor, V. Korolev and H. Chen, Information agents for mobile and embedded devices, Lecture Notes in Computer Science, Vol. 2182 (Springer, 2001) pp. 264–286.

[7] D. Garlan, D. Siewiorek, A. Smailagic and P. Steenkiste, Project Aura: Towards distraction-free pervasive computing, IEEE Pervasive Computing 1(2) (2002) 22–31.

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

[9] D.H. McKnight and N.L. Chervany, Conceptualizing trust: a typology and e-commerce customer relationships model, in: *Proceedings of the 34th Hawaii International Conference on System Sciences*, Vol. 7 (IEEE, January 2001) p. 7022.

[10] G. Stoneburner, A. Goguen and A. Feringa, Risk management guide for IT systems, Technical Report SP800-30, National Institute for Science and Technology (January 2002).

[11] S. Weeks, Understanding trust management systems, in: *IEEE Symposium on Security and Privacy* (2001) pp. 94–105.

**Brian Shand** is a Ph.D. student at the University of Cambridge Computer Laboratory. His research currently focuses on trust-based middleware for distributed computational services. Before this, he investigated distributed objects for image processing applications, for an M.Sc. at the University of Cape Town. He lives with his wife, Olivia, in Cambridge, England, but sadly they have no cats.
E-mail: Brian.Shand@cl.cam.ac.uk

**Nathan Dimmock** is a Ph.D. student at the University of Cambridge Computer Laboratory. His research interests include trust-management systems, security and privacy in ubiquitous computing and middleware. He received his Bachelors degree in Computer Science from the University of Cambridge and is a member of the IEEE and British Computer Societies.
E-mail: nathan.dimmock@cl.cam.ac.uk

**Jean Bacon** is a Reader in Distributed Systems at the University of Cambridge Computer Laboratory. She is a Senior Member of the IEEE and has acted as Editor in Chief of IEEE Distributed Systems Online since its start in 2000. She is a member of the Board of Governors of the IEEE Computer Society.
E-mail: jean.bacon@cl.cam.ac.uk