Accepted for 1st IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003), March 2003. Please do not redistribute.

Trust for Ubiquitous, Transparent Collaboration

Brian Shand, Nathan Dimmock, Jean Bacon
University of Cambridge Computer Laboratory
JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom
{Firstname.Lastname}@cl.cam.ac.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. This is illustrated with applications such as personal address books and electronic diaries. Recommendations associate 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. 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.

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 tasks, 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.

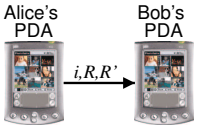
Instead, the members of each company might decide to find potential meeting times among themselves, then share only this aggregate information between the companies. This paper proposes trust and risk models to help automate interactions of this sort, making the computations as unobtrusive as possible 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 [6] to their users.

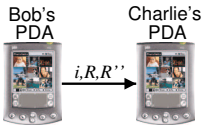
Our trust framework uses a homogeneous recommendation system, to allow users to share and exchange privileged information. This information can include conventional data such as personal contacts and calendar entries, and also trust beliefs about principals.

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



$i = \{Phone\ \#\}_{Alice}$
$R = \{Rec(i,Alice,t)\}_{Alice}$
$R' = \{Rec(i,business,t')\}_{Alice}$

(a) Alice sends Bob her work phone number.



$R'' = \{Rec(i,work,t'')\}_{Bob}$

(b) Some time later, Bob forwards Alice's number to Charlie.

Figure 1. Recommendations in action.

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 [5]. 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 others [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. This gives us a well-founded approach to trust management decisions, which is suitable for distributed computing applications.

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.

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, 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.

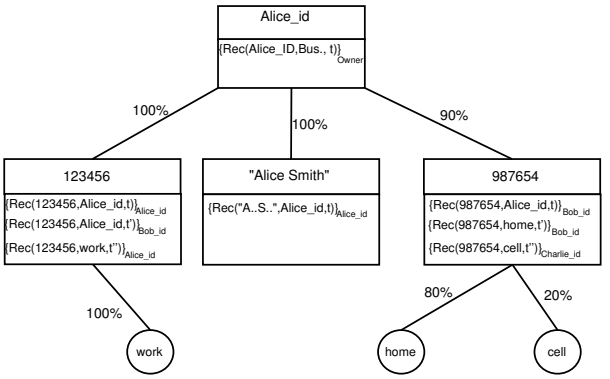


Figure 2. Each piece of information is stored separately and links between them are determined by the trust model.

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.

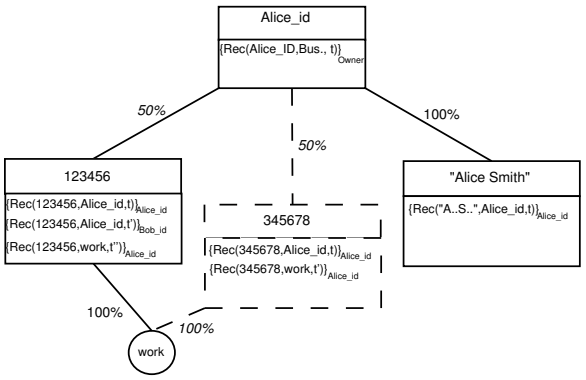


Figure 3. David receives a different view on Charlie's information about Alice, plus additional information from his own database (shown as dashed lines).

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 attached to monitors for example, because proper user-account administration is considered to be too much work — and so we aim to create a security mechanism suitable for use in the pervasive computing environment where human intervention is a valuable resource [6], 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 [4], defined in section 4.3, analogously to Weeks' proposal for formalising access control system policies [9].

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.

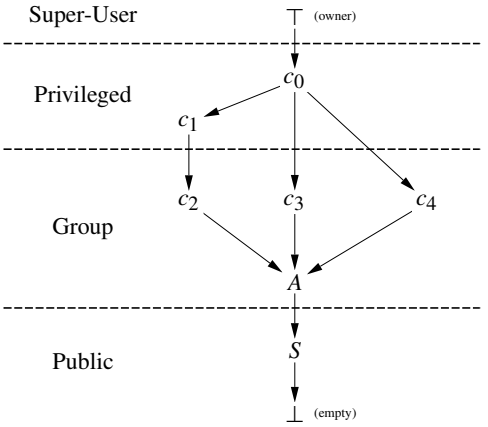


Figure 4. Example of a category hierarchy.

The diagram also shows another important feature of our framework, from a human interaction perspective. We have divided the categories into bands; these bands dictate the extra privileges granted to categories within them. This makes it far more convenient for users to manage their trust poli-

Accepted for 1st IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003), March 2003. Please do not redistribute.

cies, 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.

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 if different collaborating users assign the same category to different bands, to avoid accidental disclosure of sensitive information.

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 $\perp$. 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 $\perp$, 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, categories 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 1. Recommendations are then combined transitively to determine effective trust values, discounted according to the permissions the recommenders hold.

For example, assume 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 a user trusts Alice as a member of "business", then Bob will be considered a member too. Furthermore, if the user 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 recom-

Accepted for 1st IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003), March 2003. Please do not redistribute.

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

Table 1. Acceptable recommendations.

mender. Next, we present the structure of these trust values, before showing how they are combined to make decisions.

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* ≤ 1. This can be compared to Jøsang's logic of uncertain probabilities, based on the Dempster-Shafer theory of evidence [7].

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 $(b_1 \leq b_2)$ and $(d_2 \leq d_1)$, 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 $(b_1 \leq b_2)$ and $(d_1 \leq d_2)$, which we will use in combining recommendations below [4].

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:

- 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

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

where

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

$$(b, d) \otimes (e, f) = \begin{cases} (0, 0) & \text{if } b \leq d \\ (\frac{e}{k}, \frac{f}{k}) & \text{otherwise} \end{cases}$$

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

We also define $\bigoplus X_i$ to combine a number of recommendations monotonically, by averaging their belief and disbelief components respectively. The number of recommendations considered remains constant while computing a particular trust assessment, so this operation is monotonic.

For example, given three recommendations (0.2, 0), (0.25, 0.5) and (0.36, 0.4) as shown in figure 5, the compound recommendation deduced by $\bigoplus$ is (0.27, 0.3).

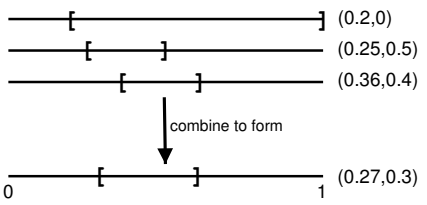


Figure 5. Combination of recommendations.

Finally, all the policy functions are considered, to allow a principal to make a reasoned trust deduction which takes into account others' recommendations. This is achieved by finding the least fixed point of all the policy functions combined, with respect to the partial order $\sqsubseteq$ — corresponding to the least specific trust assignments justified by the available recommendations. Since $\otimes$ and $\oplus$ are suitably monotone, the policy functions Pol_x are monotone too, and so this fixed point exists.

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

Accepted for 1st IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003), March 2003. Please do not redistribute.

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. 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, and so old trust policy that says something negative about a principal cannot cause our security to be compromised.

Our solution is to scale down expired positive information before it is used in the trust-model. There are also considerations of storage space on resource-limited devices and their capabilities to cache sufficient amounts of policy, but that is a topic still under investigation.

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 for us 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. However 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 powerful while still being easy to use.

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 [8] 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.

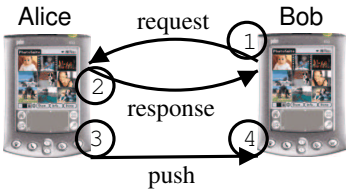


Figure 6. Possible interactions between two PDA users.

In the address book scenario, two users may interact in two different ways as shown in figure 6. 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 to 4 in figure 6). 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.

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 of 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 obviously in some situations the correct response will be unclear so 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.

This section begins with an informal justification of our risk analysis equations, which are then validated formally with the equivalent cost-benefit functions.

In order to decide whether a principal p, has read access to a phone number, n, we have to consider all the categories to which n and p are linked. n is considered to be linked to a category if the trust-model has a greater belief than disbelief in the link. For each category, c' to which n is linked, we consider p to be a member of the category c for which the product of val_c and $(b - d)$ is greatest, where val_c is the user-assigned value of a category c which has read permission on c' and b, d are the PDA's belief and disbelief, respectively, that p is a member of c.

Effectively we have now reduced the access control decision to whether p is a sufficiently strong member of c to be able to read c'. Using the variable $(b-d)$ as a measure of the strength of p's membership of c, there are three possible answers to the question: *Yes*, *No* and *Ask owner for guidance* as illustrated in figure 7. How should these partition boundaries (x and y in the diagram) be determined?

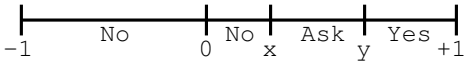


Figure 7. Number line showing how partitions of $(b-d)$ in membership of a category lead to a decision.

It is logical that if disbelief in membership is greater than belief then p is not a member of c and so the region from -1 to 0 must be *No*. The answer must be *Yes* if the value of the data is significantly lower than the value placed in the principal, which leads to:

$$y = \max\left(\frac{val_{c'} - val_{read}}{val_c}, 0\right)$$

where val_c is the user-assigned value of category c which represents the maximum benefit from trusting p with n, and $val_{c'}$ is the user-assigned value of category c' which represents the potential cost of p ignoring our recommendations and redistributing n indiscriminately. val_{read} is the fixed benefit of allowing someone to read a number, with which the owner can customize how helpful they wish to be to others: the higher the value of val_{read}, the more likely they are to let someone they do not know very well read a number.

The position of x must offset the value of helping the person against the cost in time to the owner for disturbing them. Therefore:

$$x = \min\left(\frac{CT - val_{read}}{val_c}, y\right)$$

where CT is the cost to the owner of being asked for guidance, a user-specified constant value.

These equations can be reformulated as benefit functions, that perform a risk analysis of costs and benefits to reach a decision.

$$\begin{aligned}\text{Benefit}_{yes}(b-d, val_c, val_{c'}) &= val_c.(b-d) - \\ &\quad \max(val_{c'} - val_{read}, 0) \\ \text{Benefit}_{ask}(b-d, val_c) &= val_c.(b-d) - CT + val_{read} \\ \text{Benefit}_{no}(b-d) &= -val_c.(b-d)\end{aligned}$$

The term $val_c.(b-d)$ is the expected benefit of trusting p and so it follows that the benefit of saying *No* is the expected benefit of not trusting p. In the Benefit_{yes} equation the term $\max\left(val_{c'} - val_{read}, 0\right)$ cannot be less than 0 since benefit of saying "Yes" cannot be greater than the benefit of trusting p, even for really unimportant data items.

Using these equations, the decision is then answered by:

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

6.2 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 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. We have applied this to our prototypical examples, a phone book and an appointment diary although we believe it is applicable to all recommendation-based systems, especially ones in mobile environments. Further work includes a 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, and the detection of untrustworthy principals by examining the source(s) of information found in other people's PDAs.

These studies will further validate our recommendation framework, and its unification of personal trust with access control for ubiquitous computing applications.

Acknowledgment

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, Mar. 1998. [Online]. Available: `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 *Hawaii International Conference on System Sciences 33*, pp. 1769–1777, 2000.

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

[4] M. Carbone, O. Danvy, I. Damgaard, K. Krukow, A. Møller, J. B. Nielsen, and M. Nielsen. A model for trust, Dec. 2002. EU Project SECURE IST-2001-32486 Deliverable 1.1.

[5] T. Finin, A. Joshi, L. Kagal, O. Ratsimore, V. Korolev, and H. Chen. Information agents for mobile and embedded devices. *Lecture Notes in Computer Science*, 2182:264–286, 2001.

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

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

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

[9] S. Weeks. Understanding trust management systems. In *IEEE Symposium on Security and Privacy*, pp. 94–105, 2001.