# A Distributed Trust Model

**Alfarez Abdul-Rahman & Stephen Hailes**
{F.AbdulRahman, S.Hailes}@cs.ucl.ac.uk
Department of Computer Science, University College London,
Gower Street, London WC1E 6BT, United Kingdom.

## Abstract

*The widespread use of the Internet signals the need for a better understanding of trust as a basis for secure on-line interaction. In the face of increasing uncertainty and risk, users must be allowed to reason effectively about the trustworthiness of on-line entities. In this paper, we outline the shortcomings of current security approaches for managing trust and propose a model for trust, based on distributed recommendations.*

## 1. Introduction

The Internet of the past is one of limited services and a fixed set of users, mainly academics and scientists. From this, it has developed into a pervasive utility, playing host to a vast range of services. The future will see it being used for serious commercial transactions by anyone and from any location.

With all this comes greater uncertainty and risk arising from the intentional hostility or carelessness of on-line entities. Existing examples of the risks include viruses and Trojan horses, applets and macros embedded in documents, subverted databases of sensitive financial information, etc. The level of expertise and experience required to recognise potential risk in every on-line interaction is currently beyond the ability and resources of the average user. To help with this situation, users must be given the ability to assess the trustworthiness of entities it encounters.

Current security technology provides us with some capability to build in a certain level of trust into our communication. For example, cryptographic algorithms for privacy and digital signatures, authentication protocols for proving authenticity and access control methods for managing authorisation. However, these methods cannot manage the more general concept of 'trustworthiness'. Cryptographic algorithms, for instance, cannot say if a piece of digitally signed code has been authored by competent programmers and a signed public-key certificate does not tell you if the owner is an industrial spy. Current security technology is currently lacking the complementary tool for managing trust effectively.

In this paper, we propose our approach to the problem of trust management with a distributed trust model, and a recommendation protocol.

We will continue by clarifying our notion of trust in §2, followed by the motivation behind this work in §3. Other related work is discussed in §4. An introduction to our proposal is given in §5, followed by a detailed description of the trust model in §6. We then describe the Recommendation protocol in §7. An algorithm for calculating trust, using values in recommendations, is described in §8. Further discussion, including feedback from the New Security Paradigms 97 participants, and future work, is in §9. Finally, we conclude the paper in §10. A glossary of definitions is given at the end of this paper.

## 2. Definition of trust

Before beginning our discussions on trust, we must clarify our definition of trust, as it is such a subjective and elusive notion. In this paper, we use Diego Gambetta's definition of trust:

"trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of his capacity of ever to be able to monitor it) and in a context in which it affects [our] own action" [Gam90].

Of importance here are three points made in the definition above: 1) that trust is subjective, 2) trust is affected by those actions that we cannot monitor, and 3) the level of trust depends on how our own actions are in turn affected by the agent's[1] actions.

## 3. Motivation

In the previous section, we gave brief examples illustrating the need for more effective trust management techniques. We now discuss in more detail the properties of current security practices, and its issues which motivate a need for complementary trust management schemes. The following trends in current security practice impact the management of trust.

### 3.1 Hard security

Cryptographic algorithms and firewalls are examples of hard security mechanisms, and they have the general property of allowing complete access or no access at all. Hard security also assumes complete certainty. As the notion of trust precludes some element of uncertainty, an alternative 'soft' approach will be more suitable for trust management. 'Soft security' is the term used by Rasmusson et al [RJ96a, RJ96b, RRJ96] to describe a 'social control' model which acknowledges that malicious agents may exist among benign ones, and attempts to make them known.

### 3.2 Centralised protocols

In centralised protocols, a common trusted intermediary, call it the 'Trusted Authority' (TA), is used to form trust relationships. Hierarchical and Trusted Third Party (TTP) protocols contain examples of these. However, a TA can never be a good enough 'authority' (or recommender of trust) for everyone in a large distributed system. Its credibility depletes, and its recommendations increase in uncertainty, as its community of trustees grows.

### 3.3 Implicit definitions and inconsistent assumptions

In common practice, the 'trusted' label is given to systems that have been tested and proven to have met certain criteria. However, the 'trusted' label is misleading. Firstly, this is due to the failure of 'trusted systems' designers to provide a concrete definition of 'trust'. This, secondly, leads to the often misleading assumption that 'trusted' implies 'nothing can go wrong', which in turn implies that the tests covered all eventualities. This is not always true, and is difficult to guarantee[2]. The Trusted Computing Base guideline [DoD85] is one such area that faces this problem of ambiguous definition of trust[3].

### 3.4 Transitivity

A common assumption of most authentication protocols is that trust is transitive, i.e. the assumption that

[1] In this paper, an entity is any object in the network, and an agent is an entity that is capable of making trusting decisions.

[2] In Bruce Schneier's words: "No amount of general beta testing will reveal a security flaw, and there's no test possible that can prove the absence of flaws" [Sch97].

[3] An interesting report on a conference panel regarding this subject can be found in [Zur97].

(Alice trusts Bob) .&. (Bob trusts Cathy)

$\Rightarrow$ Alice trusts Cathy

This is not *generally* true [Jøs96, CH96]. We posit that transitivity may hold if certain conditions are met. We have termed this *conditional transitivity* and the conditions that may allow transitivity are (with reference to the example above):

a) Bob explicitly communicates his trust in Cathy to Alice, as a 'recommendation'.

b) Alice trusts Bob as a recommender, i.e. recommender trust exists in the system.

c) Alice is allowed to make judgements about the 'quality' of Bob's recommendation (based on Alice's policies).

d) Trust is not absolute, i.e. Alice may trust Cathy less than Bob does, based on Bob's recommendation.

In this paper, we will adopt the *conditional transitivity* of trust, i.e. transitivity based on conditions a) – d) above.

## 4. Related work

Yahalom et al (YKB) [YKB93, YKB94] discussed in significant detail the concept of trust in distributed systems. They highlighted the fact that there is no effective formal way to reason about trust in distributed systems and that a formal tool to analyse trust requirements in security protocols is required. To remedy this situation, they defined *trust classes,* made the distinction between *direct* and *recommendation* trust and proposed a formalism for analysing trust in authentication protocols. However, their work falls short of defining a framework for building protocols with extended trust information.

Rasmusson and Jansson [RJ96a, RJ96b] introduced the idea of social control, which is a 'soft' approach to security. The open system modelled in their work represents an electronic marketplace, which consists of buyer and seller 'actors'. It is up to the good actors to identify 'cheaters' and propagate this information throughout the system. Social control attempts to remedy the situation where there is no easy way for a component to know all the other components in open systems, by relying on group behaviour to influence the behaviour of its group members. This approach fits in well with our notion of a distributed trust model. Furthermore, their contribution gave clues on how to build a framework, or model, for distributed trust, which YKB does not provide.

The contribution of this work to our research is essentially the paradigm in which we attack our problem. Just as in the electronic marketplace scenario, our agents do not have to rely on centralised mechanisms. Trust information is propagated throughout the system via the interaction of the agents themselves, and the 'quality' of this information is calculated on the basis of the perceived trustworthiness of recommenders[4]. Trust is then revised upon receipt of new recommendations or new experiences.

Other related work includes Pretty Good Privacy [Zim94], which helped inspire the distributed nature of our model. Trust is also being used as a basis for cooperation among autonomous agents in the area of Distributed AI by Marsh [Mar94]. His approach involves the addition of perceived risk and utility of committing resources in a cooperative relationship, which results in a complex trust calculation algorithm.

## 5. Proposal

Our proposal extends and generalises current approaches to security and trust management, based upon four goals:

1. To adopt a **decentralised** approach to trust management.
2. To **generalise** the notion of trust.

[4]A recommender is a source of trust information, also called 'recommendations'.

3. To reduce ambiguity by using **explicit trust statements.**
4. To facilitate the exchange of trust-related information via a **recommendation protocol**.

We provide further justification for these goals below.

## 5.1 Decentralisation

With decentralisation, each rational agent will be allowed to take responsibility for its own fate. This is a basic human right. Its policies need not be communicated, so there is less ambiguity and no effort involved in trying to understand them. Each agent then makes decisions for itself, on its own policies.

The disadvantage of decentralisation is that more responsibility and expertise is required from the agent for managing trust policies. However, without leaving this model, agents on the network will still have the option of defaulting to centralised trust models so that this responsibility can be assigned to their trusted authority, if so desired. Decentralisation does not completely replace current centralised approaches, but it gives agents a choice of managing their own trust.

## 5.2 Generalising trust

When we say we trust someone, we know, with a large amount of certainty, exactly which *aspects* of trust we are referring to. For example, we trust that our car mechanic will carry out a satisfactory job of car servicing, but not that she will handle our domestic plumbing needs. There are also instances when we trust one entity more than another, e.g. we trust car mechanic Gary more than mechanic Eric for some reason. This hints at different levels of trust.

To be able to capture this potentially large amount of trust information, we need to generalise trust information. In our model, we use *trust categories* to represent which aspect of trust we are referring, and *trust values* for the different levels of trust within each category.

## 5.3 Explicit trust statements

The reason for making trust explicit is straightforward, i.e. to lessen ambiguity in recommendations which contain trust statements. The issues relating to implicit assumptions are discussed in §3.3. In our model, trust categories and trust values serve to make trust statements more explicit.

## 5.4 Recommendations

In a large distributed system, it is difficult to obtain knowledge about every entity in the network, let alone first hand knowledge and experience of them. Yet, any entity in the network is a potential target of communication. Human beings cope with unknown entities via first impressions, and word of mouth. Entities in the network represent human beings one way or another, and they exist in their own social system. Consequently, it makes sense for them to have, as far as possible, the capability to reason about trust in the same ways in which humans would. First impressions rely on complex sensory perceptions. Thus, the only option open to network agents for coping with uncertainty is via word of mouth, or recommendations.

The need to consider the basis of trust is also important. Although it is the ideal case to have complete information about the basis of a recommender's trust in another entity, this is virtually impossible to attain. Constraining the basis for trust will not be a remedy to this situation as this assumes that trust is objective. Trust, as defined in §2, is subjective, and there will always be hidden factors (intentional or subconsciously) behind a decision to trust or distrust.

By allowing agents to choose their own trusted recommenders, the uncertainty resulting from the 'hidden factors' may be accommodated. With repeated interaction, the subjective 'quality' of a recommender's recommendations can be judged with increasingly better accuracy.

We have proposed a *Recommendation Protocol* to facilitate the propagation of trust information. A protocol is essential as a standard vehicle for the exchange of trust information, and to avoid ambiguities in queries or requests for recommendation. We argue that it makes sense to recommend trust because in the absence of an infinite pool of resources, agents, just as humans do, rely on information from others.

## 5.5 Novelty and suitability of proposed approach

Our approach is intended to complement current security practices by forming a general model within which trust can be managed more effectively. This is not 'yet another certification mechanism'. In a world where people live with uncertainty, our model copes with these uncertainties by allowing agents to reason with different degrees of trust, and obtain information from sources that they trust.

In this work, we are concerned with the general notion of trust, one that goes beyond cryptographic protocols. This is important because agents need a flexible means to ascertain a variety of properties about a variety of other entities. For this to work we need to generalise trust.

We believe that our model will be most suited to trust relationships that are less formal, temporary or short-term trust relationships, or ad-hoc commercial transactions. Our model will not be suited to formal trust relationships based on legally binding contracts. In such contractual relationships, trust is placed in the 'Trusted Authority' to enforce the law upon parties that breach the contract.

## 6. The Trust Model

In this section, we explain how trust is defined in the trust model by describing its elements. This section can also be regarded as containing the assumptions that we have made in designing our trust model.

### 6.1 Agents

Entities that are able to execute the Recommendation Protocol are called *agents*. This is to differentiate them from static entities like printers and disk volumes. Any entity may be recommended, but only agents can send and receive recommendations.

### 6.2 Trust relationships

A trust relationship exists between Alice and Bob when Alice holds a belief about Bob's trustworthiness. However, the same belief in the reverse direction need not exist at the same time. In other words, Alice's trust relationship is unidirectional.

The properties of a trust relationship in our model are:

1. It is always between exactly two entities.
2. It is non-symmetrical (or unidirectional).
3. It is conditionally transitive.

If mutual trust exists between the same entities, we represent them as two separate trust relationships. This allows each of these relationships to be manipulated independently.

Two different types of relationships are distinguished. If Alice trusts Bob, then there is a *direct trust relationship*. If Alice trusts Bob to give recommendations about other entities' trustworthiness, then there is a *recommender trust relationship* between Alice and Bob.

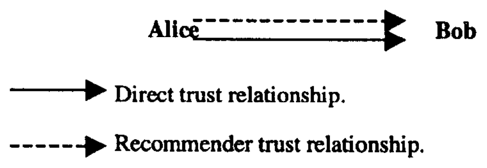

**Figure 1** Types of trust relationships

Trust relationships exist only within each agent's own database. Therefore, there is no such thing as a 'global map' of trust relationships in our model. This also makes trust relationships in our model highly volatile. The ability for each agent to revise the properties of each relationship at any time also makes trust relationships potentially unstable.

By relaxing the constraints on how to build trust relationships, we are able to allow this model to be used for any type of trust architecture[5], e.g. hierarchical, digraphs or hybrids. Most architectures are policy driven, i.e. the shape of the architecture reflects the policies used to build them. Since we do not incorporate policies in our model, it is open to arbitrary architectures.

## 6.3 Trust Categories

Agents use *trust categories* to express trust towards other agents in different ways depending upon which particular characteristic or aspect of that entity is under consideration at that moment. For example, we trust a CA to certify public keys (category "Sign-key"), but not to attest to the key-holder's credit status (category "Credit").

## 6.4 Trust Values

Trust values are used to represent the different levels of trust an agent may have in another.

Naturally, there is no one universal value system because its use is application specific. However, standardisation is important for interoperability. Therefore, it is important that a value system is proposed, even if we must base its semantics on pure intuition. Below, we outline the trust values and their meaning as used in our trust model.

Trust values in our model are constrained within each category and are independent of values in other categories.

The qualitative nature of trust makes it difficult to represent trust with continuous values with any meaningful accuracy. Thus, discrete levels of trust are used in this model.

[5] The combined graph of trust relationships.

Two types of values are used, and they relate to the types of trust relationships described in §6.2:

1. *Direct trust value*: This is relevant to direct trust relationships.
2. *Recommender trust value*: This is relevant to recommender trust relationships.

The values and their descriptions are given below.

| *Value* | *Meaning* | *Description* |
|---|---|---|
| -1 | Distrust | Completely untrustworthy. |
| 0 | Ignorance | Cannot make trust-related judgement about entity. |
| 1 | Minimal | Lowest possible trust. |
| 2 | Average | Mean trustworthiness. Most entities have this trust level. |
| 3 | Good | More trustworthy than most entities. |
| 4 | Complete | Completely trust this entity. |

**Table 2** Direct Trust Value Semantics

| *Value* | *Meaning* | *Description* |
|---|---|---|
| -1 | Distrust | Completely untrustworthy. |
| 0 | Ignorance | Cannot make trust-related judgement about agent. |
| 1 | 'Closeness' of recommender's judgement to own judgement about trustworthiness. | |
| 2 | | |
| 3 | | |
| 4 | | |

**Table 1** Recommender Trust Value Semantics

## 6.5 Reputation and Recommendation

The concatenation of an agent's ID or name, the trust category and a trust value is called a *Reputation*:

*Reputation* = {Name, Trust-Category, Trust-Value}

A *Recommendation* is a communicated trust information, which contains reputation information.

Each agent stores reputation records in its own private database and uses this information to make recommendations to other agents.

## 7. Recommendation Protocol

For brevity and clarity, we will leave out details on message integrity and privacy issues and concentrate on the trust-related content of the recommendation protocol messages. To recap, each agent may be a *recommender*, or a *requestor* of a recommendation. Any entity may be a *target* for a recommendation.

### 7.1 Message structure

A requestor issues a recommendation request message, or an `RRQ` , and receives a `Recommendation` message. Recommendations can be refreshed or revoked using the `Refresh` message. These messages have the following structure (shown in BNF-like format):

#### 7.1.1 RRQ

```
RRQ ::= Requestor_ID, Request_ID, Target_ID,
     Categories, RequestorPKC, GetPKC,
     Expiry

Categories ::= SET OF {Category_Name}
```

#### 7.1.2 Recommendation

```
Recommendation ::= Requestor_ID, Request_ID,
     Rec_Path, [SEQUENCE OF
     {Recommendation_Set, TargetPKC} | NULL]

Rec_Path ::= SEQUENCE OF {Recommender_ID}

Recommendation_Set ::= SET OF
     Recommendation_Slip

Recommendation_Slip ::= SET OF SEQUENCE
     {Target_ID, Category_Name, Trust_Value,
     Expiry}
```

#### 7.1.3 Refresh

```
Refresh ::= Rec_path, Recommendation_Set
```

`Requestor_ID`, `Request_ID`, `Target_ID` and `Recommender_ID` are self-explanatory. `Categories` is a set of category names that the requestor is interested in enquiring about. `RequestorPKC` is the public-key certificate of the requestor which can be used to encrypt `Recommendation_Set` if the recommender wishes to do so. `GetPKC` is a Boolean flag which, when set to true, indicates that the requestor would also like a copy of the target's public key certificate for further communication. If a public-key certificate is available, it is returned in the `Recommendation`, in the `TargetPKC` field.

The `Rec_Path` field contains an ordered sequence of recommender IDs. This shows the path through which the `Recommendation` propagated from the *recommender* to the *requestor*.

The `Recommendation_Set` contains multiple instances of the `Recommendation_Slip`, which contains the actual trust information that the requestor is interested in. For each category, there is a sequence containing the `Category_Name`, the

Trust_Value of the target with respect to this Category_Name, and the Expiry.

The Expiry field contains the expiry date for the RRQ or Recommendation. In the case of the RRQ, this is used to discard any old RRQs that may still be floating around in the system. In the case of each Recommendation_Slip, this is used to indicate the validity period of the recommendation, after which the recommendation should not be relied upon any further.

If the RRQ reaches a dead end in its path, and fails to reach a recommender who is able to provide a recommendation, the fields Recommendation_Set and TargetPKC will be replaced by NULL values.

## 7.2 Protocol flow

The protocol flow is best described using an example, as depicted in Figure 2.

$$\begin{tikzcd}\textbf{Alice} \arrow[r, dashed] & \textbf{Bob} \arrow[r, dashed] & \textbf{Cathy} \arrow[r] & \textbf{Eric}\end{tikzcd}$$

**Figure 2** Example: Can Alice trust Eric the mechanic?

### 7.2.1 Requests and Recommendations

Let us assume that Alice (the *requestor*) is requesting a recommendation from Bob (the *recommender*) about Eric (the *target*). Alice is interested in Eric's reputation for servicing cars, especially VW Golfs, one of which Alice drives (trust category = "Car_Service"). The protocol run is as follows.

| | | |
|---|---|---|
| *1.* | *Alice->Bob:* | *Alice, rrqA01, Eric, [Car_Service], T, 20000101* |
| *2.* | *Bob->Cathy:* | *Bob, rrqB01, Eric, [Car_Service], T, 20000101* |
| *3.* | *Cathy->Bob:* | *Bob, rrqB01, [Cathy], [(Eric,Car_Service,3,20000131)],* $PK_{Eric}$ |
| *4.* | *Bob->Alice:* | *Alice, rrqA01, [Cathy,Bob], [(Eric,Car_Service,3,20000131)],* $PK_{Eric}$ |

The protocol is simple and straightforward. Each RRQ is sent to the requestor's set of recommenders, trusted to recommend in the category in question. In the example above, Alice sends an RRQ to Bob because she trusts Bob as a recommender for car servicing mechanics, and Bob trusts Cathy in a similar capacity. Since Bob cannot say anything about Eric with respect to "Car_Service", Bob forwards Alice's RRQ to Cathy, who may know. Cathy, in fact, knows about Eric's workmanship, and Cathy believes that Eric's reputation for it is good, i.e. *in Cathy's opinion*, Eric's trust value with respect to category "Car_Service" is 3.

Cathy replies to Bob with a recommendation in message 3. Notice that the Requestor_ID and Request_ID represents the last sender (or forwarder) of the RRQ in the forward RRQ chain, and not the original issuer of the RRQ. This is designed this way to encourage Recommendations to be returned using the forward path, which contains at least one trusted node (the original recommender Bob). Cathy also appends Eric's public key certificate to the end of the recommendation.

Bob receives the recommendation from Cathy, and changes the Requestor_ID and Request_ID fields. Bob also adds his own ID to the tail of the Rec_Path list. He then forwards this to Alice.

### 7.2.2 Revoking and refreshing Recommendations

The reputation of entities changes over time so there is a need to update the reputation information in the system. The classic method for handling this is through revocation where revocation messages are sent out to revoke certificates. In our trust model there is a need to revoke, as well as *refresh* recommendations. In fact, revoking is a subset of refreshing; it is contained in the same *Refresh* message type. To revoke, a recommender resends the same recommendation with trust value 0. The receiver will treat this as any other 0-value recommendation. Changing the trust value to any other value (i.e. {-1 ,1..4}) will refresh the recommendation.

In our previous example, if Cathy found out that Eric had made several bad jobs of servicing her car, Cathy may decide that Eric is not trustworthy after all, and would like to inform her previous requestors of this. These messages show how this will be carried out.

*5. Cathy->Bob: [Cathy], [(Eric,Car_Service,1,20000131)]*

Bob, upon receiving message 5 also decides to propagate this *Refresh* message to his previous requestors, who, in this example, concerns just Alice.

*6. Bob->Alice: [Cathy, Bob],*
*[(Eric,Car_Service,1,20000131)]*

$$\begin{tikzcd} \textbf{Alice} & \textbf{Bob} \arrow[l] & \textbf{Cathy} \arrow[l] \end{tikzcd}$$

**Figure 3** Refreshing recommendations (arrow points direction of *Refresh* message flow)

Public keys are not included in *Refresh* messages because *Refresh* messages are for refreshing trust, not keys. Keys are just piggybacked on *Recommendations* to avoid another round of protocol for obtaining keys.

The Recommendation Protocol makes revocation easier. All that is required is for the original recommender to re-send the *Refresh* message to all previous requestors of the same target and category. With traditional certificate mechanisms, the target entity itself carries the certificate, and it is not easy to determine whom it will present the certificate to next; therefore, distributing the revocation certificate is harder. Furthermore, since there are potentially more recommenders in our model than CAs in normal certification architectures, most recommenders would normally have fewer agents to broadcast revocations to, since decentralisation increases the number of message sources (recommenders) and reduces the number of requestors for each recommender. This shows how much simpler trust management is through decentralisation.

One major risk in sending *Refresh* messages is the propagation delay of messages through the system. This depends on the availability of the agents in the propagation path and the promptness of each agent in the path in forwarding protocol messages. However, since the protocol is decentralised and any agent may be a recommender, it is suspected that the availability of the refreshed reputation messages will be higher than in a centralised system.

In short, the Recommendation Protocol makes revoking and refreshing trust information easier and improves availability of *Refresh* messages.

### 7.2.3 Recommendation about recommenders

In the example above, if Alice does not know Cathy before, then it is difficult for Alice to judge the 'quality' of Cathy's recommendation about Eric. In this case, Bob may help Alice, by sending Alice a recommendation about Cathy to accompany Cathy's recommendation. This, for example, may come after message 4 above:

*4a. Bob->Alice: Alice, rrqA01, [Bob],*
*[(Cathy,Rec_Car_Service,3,*
*19981231)], NULL*

Here in message 4a, "Rec_Car_Service" is used to represent the trust category for "recommending car servicing agents".

### 7.2.4 'Shopping' for recommendations

There may be instances when the requestor needs to acquire service, but does not know from which agents to obtain the service. In this case, the requestor may send out an RRQ to request a 'catalogue' of recommended agents. A question mark '?' in the Target_ID field of the RRQ may be used to indicate this desire, for example, Alice's request to Bob may look like this

7. *Alice->Bob: Alice, rrqA01, ?, [Car_Service], T, 20000101*

and Bob may give his own recommendations

8. *Bob->Alice: Alice, rrqA01, [Bob], [(Fred,Car_Service,3,20000131), (Jim,Car_Service,3,20000131)],* $PK_{Fred}$

Bob may also forward Alice's RRQ to Cathy, which would return message 3.

## 8. Computing trust

The trust value of a target for a single recommendation path is computed as follows:

$$tv_p(T) = tv(R1)/4 \times tv(R2)/4 \times .. \times tv(Rn)/4 \times rtv(T) \qquad (1)$$

Where,

| | |
|---|---|
| $tv(Ri)$: | Recommender trust value of recommenders in the return path including the first recommender (who received the original RRQ) and the last recommender (who originated the Recommendation). |
| $rtv(T)$: | The recommended trust value of target T given in the recommendation. |
| $tv_p(T)$: | The trust value of target T derived from recommendation received through return path *p*. |

A requestor may have multiple recommendations for a single target and thus the recommendations must be combined to yield a single value. For the moment, we have adopted the averaging method used by Beth et al in [BBK94]. Averaging evens out the impact of any single recommendation. The final single trust value for target T is computed as follows:

$$tv(T) = Average(tv_1(T),..,tv_p(T)) \qquad (2)$$

We will illustrate this algorithm with our previous example with Eric the mechanic. From the previous example, we have the recommendation path from Cathy to Alice (refer to this as *Rec-path-1*):

Cathy → Bob→ Alice

Furthermore, we have the following trust statement:

- Cathy trusts Eric value 3 (from example).

Assume further that:

- Alice trusts Bob's recommender trust value 2.
- Alice trusts Cathy's recommender value 3 trust (after recommendation from Bob).

We also assume that Alice had sent out a second RRQ for the same trust category "Car_Service" to David, another of her trusted recommenders, and had received a recommendation from him about Eric (refer to this as *Rec-path-2*).

Alice calculates trust for Eric on *Rec-path-1* as follows:

$$\begin{aligned} tv_1(Eric) &= tv(Bob)/4 \times tv(Cathy)/4 \times rtv(Eric) \\ &= 2/4 \times 3/4 \times 3 \\ &= 1.125 \end{aligned}$$

We assume that by using the same algorithm (1), Alice obtains a value of 2.500 ($tv_2(Eric) = 2.500$) on *Rec-path-2*. Now Alice can apply algorithm (2) to obtain the following:

$$\begin{aligned} tv(Eric) &= Average(tv_1(T),tv_2(T)) \\ &= Average(1.125,2.500) \\ &= 2.375 \end{aligned}$$

Computing trust is a difficult area and, at this moment, the trust computation algorithm above was derived largely from intuition. A standard algorithm is necessary to lessen ambiguity in trust value recommendations, and to allow most *requestors* to be confident that what is received in recommendations comes *close* to a universal[6] standard.

## 9. Discussion and Future Work

The workshop provided an excellent forum for the proactive analysis of new ideas for security and this work has benefited greatly from it. We highlight the important issues that were raised below.

a) The role of reputation

It was pointed out that in some societies, reputation and recommendation does not work well. In China for example, personal and family relationships overcome public reputation. Clearly, in such a community, complementary methods for managing trust will be used, in addition to those proposed here. Although the notion of reputation was proposed as the guiding information for making trust decisions, the model is not limited to just that. Since the model does not strictly define the basis of relayed trust information (in other words, the context of the information), the social mechanism of personal and family relationships can be built into the model as an identity-based model of trust ("I trust you because you're my uncle Ching's daughter-in-law"). Obviously, the trust algorithm for calculating trust based on this model will be different. In short, the model does not completely rule out relationships that are not built on reputation.

b) Clearly defined terms

In an area as elusive as trust, it is important that the terms used are clearly defined. The participants seemed happy with the definition of 'trust given in §2 above. Various participants wanted a more precise definition of risk when discussing trust, a notion which is very tightly related to risk. One suggestion was that Nancy Leveson's discussion on risks be studied in this context [Lev95].

c) Actions of agents

When discussing trust, we are concerned with the behaviour of agents as our disposition to trust depends on our prediction of their course of action. With regards to this, there are different kinds of actions that we need to consider. The first is i) proper action, i.e. doing the right thing. In addition, we should also consider three other possible actions: ii) misfeasance, i.e. performing the wrong action, iii) malfeasance, i.e. performing the right action incompetently and iv) nonfeasance, i.e. performing no action when action is required. Each agent differs in their probability in i) - iv) above. Therefore, care in determining the likelihood of each possible types of action will ensure a more appropriate trusting decision and fairer dissemination of reputation information.

d) Concreteness of *tv()*

The trust calculation algorithm *tv()* was criticised for being too ad-hoc, and the authors acknowledges this fact. However, an algorithm is required to evaluate the recommended trust values. In the absence of concrete formulas, we were forced to devise a version of the algorithm intuitively, until more work reveals a better algorithm. There was also the suggestion of providing user guidelines for the use of recommended trust values, and give a specific algorithm as merely an example, as any algorithm will be ad-hoc. In relation to this, one participant commented that as a psychological phenomenon, trust is not even partially ordered. Another alternative suggestion at the workshop was for trust to be represented as fuzzy, rather than ordinal, values. To add to this array of possible representations of trust, the authors themselves have thought about representing trust as a relation between agents, rather than absolute trust values. The representations will

[6]In discussions about areas as subjective as trust, it makes more sense to think of the term *universal* as being constrained by a particular application domain where common standards exist, e.g. the domain of business or finance, instead of taking 'universal' as a synonym for 'global'.

basically have the form of "Alice is more trustworthy than Bob" in the requestor's database. It is unclear how these relations may be exchanged and used more effectively than values and more work is certainly needed in investigating the vast possibilities of representing trust 'values' concretely and mathematically.

## 9.1 Issues ignored

So far, we have ignored a large number of issues in our work, which include provisions for anonymity, entity naming, memory requirements for storing reputations, and the behaviour of the Recommendation Protocol. These issues have been ignored deliberately so that the more complex and understudied area of trust can be satisfactorily pursued, since the issues above are being tackled in work by other researchers. For example, the work in SPKI [Ell96] includes a novel attempt at eliminating the need for global name spaces and interfaces well to what we proposed, as a means of delivering information.

## 9.2 Future work

One of our concerns is the lack of understanding in the meaning of trust in computer systems. Currently, we are looking into this problem by surveying the different semantics of trust within areas as diverse as sociology, psychology and philosophy, as well as distributed systems.

There is also a need to look into monitoring and revising trust of other entities, to maintain the dynamic and non-monotonic properties of trust in the model.

Finally, we intend to test the behaviour of our protocol and trust calculation algorithms, based on simulation.

## 10. Conclusion

In this paper, we highlighted the need for effective trust management in distributed systems, and proposed a distributed trust model based on *recommendations*. This work, and those being carried out by other researchers, has barely scratched the surface on the issues related to the complex notion of trust. Nevertheless, it is an issue vital to the engineering of future secure distributed systems, and an area with much scope for work.

## Glossary

| | |
|---|---|
| *Entity* | Any object in a network. |
| *Agent* | Any entity that is capable of making trust-related decisions (therefore able to participate in the Recommendation Protocol). |
| *Direct trust* | Trust in an entity, within a specific category and with a specific value. |
| *Recommender trust* | Trust in an agent to recommend other entities. |
| *Trust Category* | The specific aspect of trust relevant to a trust relationship. |
| *Trust Value* | The amount of trust, within a trust category, in a trust relationship. |
| *Reputation* | Trust information that contains the name of the entity, the trust category, and a trust value. |
| *Recommendation* | Reputation information that is being communicated between two agents about another entity. |

## Acknowledgements

We are extremely grateful to Prof. Jon Crowcroft for his help in the preparation of this paper, and to the anonymous referees and workshop participants for their critical comments. We also express our gratitude to the ever-helpful workshop organisers for funding Alfarez's attendance at the workshop.

## References

[Abd97] Alfarez Abdul-Rahman. *The PGP Trust Model*. EDI-Forum, April 1997. Available at http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/

[BBK94] Thomas Beth, Malte Borchedring, B. Klein. *Valuation of Trust in Open Networks*. In Proceedings, European Symposium on Research in Computer Security 1994, ESORICS94, pp 3-18.

[DoD85] U.S. Department of Defense. *Department of Defense Trusted Computer System Evaluation Criteria*. DoD 5200.28-STD, 26 December, 1985.

[El196] C.M. Ellison, B. Frantz, B. Lampson, R. Rivest, B.M. Thomas, T. Ylonen. *Simple Public Key Certificate,* Internet Draft, 29 July 1997. Available at http://www.clark.net/pub/cme/html/spki.html.

[Gam90] Diego Gambetta. *Can We Trust Trust?*. In, Trust: Making and Breaking Cooperative Relations, Gambetta, D (ed.). Basil Blackwell. Oxford, 1990, pp. 213-237.

[Jøs96] Audun Jøsang. *The right type of trust for distributed systems*. In Proceedings, New Security Paradigms '96 Workshop, 1996.

[Lev95] Nancy Leveson. *Safeware: System Safety and Computers.* Addison-Wesley. New York, 1995.

[Mar94] Stephen Marsh. *Formalising Trust as a Computational Concept*. Ph.D. Thesis, University of Stirling, 1994.

[RJ96a] Lars Rasmusson, Sverker Jansson. *Simulated Social control for Secure Internet Commerce (position paper).* In Proceedings, New Security Paradigms '96 Workshop.

[RJ96b] L. Rasmusson, S. Jansson. Personal *Security Assistance for Secure Internet Commerce (position paper).* In Proceedings, New Security Paradigms '96 Workshop.

[RL96] Ronald Rivest, Butler Lampson. *SDSI – A Simple Distributed Security Infrastructure*. http://theory.lcs.mit.edu/¬cis/sdsi.html.

[RRJ96] Lars Rasmusson, Andreas Rasmusson, Sverker Jansson. *Reactive Security and Social Control*. In Proceedings, 19th National Information Systems Security Conference, Baltimore, 1996.

[Sch97] Bruce Schneier. *Why Cryptography Is Harder Than It Looks*. Information Security Bulletin, Vol. 2 No. 2, March 1997, pp. 31-36. Available at http://www.counterpane.com.

[YKB93] Raphael Yahalom, Birgit Klein, Thomas Beth. *Trust Relationships in Secure Systems - A Distributed Authentication Perspective.* In Proceedings, IEEE Symposium on Research in Security and Privacy, 1993.

[YKB94] R.Yahalom, B.Klein, T.Beth. *Trust-Based Navigation in Distributed Systems*. Computing Systems v.7 no. 1.

[Zim94] Phil Zimmermann. *PGP User's Guide*. MIT. October 1994.

[Zur97] Mary Ellen Zurko. *Panels at the 1997 IEEE Symposium on Security and Privacy, Oakland, CA, May 5-7, 1997.* CIPHER, Electronic Issue #22, 12 July, 1997. Available at http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/