

An Evidence Based Architecture for Efficient, Attack-Resistant Computational Trust Dissemination in Peer-to-Peer Networks

David Ingram

dmi1000@cam.ac.uk

University of Cambridge Computer Laboratory
15 JJ Thompson Avenue, Cambridge, CB3 0FD, United Kingdom

Abstract. Emerging peer to peer (P2P) applications have a requirement for decentralised access control. Computational trust systems address this, achieving security through collaboration. This paper surveys current work on overlay networks, trust and identity certification. Our focus is on the particular problem of distributing evidence for use in trust-based security decisions. We present a system we have implemented that solves this in a highly scalable way, and resists attacks such as false recommendations and collusion.

1 Introduction

1.1 A metaphor for trust-based security

In the physical world, there are three main approaches to access control. If we wish to secure a building, we could lock the door and issue keys only to those who work in the building. This makes access less convenient though, so it could be better to leave the door unlocked and save time for our legitimate users. Alternatively, we might choose to leave the door unlocked but employ a security guard who sits in the lobby keeping an eye on those who pass by. The guard won't often have to stop anyone because he will recognise those who work in the building; also he can make an assessment on whether strangers are a threat, based on factors such as if they are being accompanied by someone he does know.

In the online world, typically we can only choose the first two alternatives – either to secure the resource and issue (digital) keys to those who are permitted access, or to allow anyone access. Computational trust modelling is a way of implementing the third option (a decision-making security guard) in an internet environment. For many applications this enables a new and more acceptable combination of security and convenience.

1.2 Our approach

We form models for **trust** and **risk** in online entities, and use this information to make access control decisions. This is useful in many application domains, such as Internet

auctions, spam filtering, P2P storage services and so on. A key feature of our model is its use of **recommendations** to exchange trust information between principals. This creates a requirement for an effective and scalable mechanism to distribute such information across the network.

A typical scenario involves millions of principals, most of whom do not know each other. Patterns of interaction may be random and not exhibit much locality of reference. Furthermore, any information sent via the network can be falsified, including recommendations and routing information, yet the system must be secure.

Overlay networks can be used to provide deterministic, scalable data access for P2P applications. Using such techniques we can look up any piece of evidence with a logarithmic number of messages and collate all that is known about each principal in **behaviour profiles**. We combine this with a set of Certification Agencies to limit attacks by making it expensive to obtain extra identities.

Our prototype is named ENTRAPPED (Efficient Network Trust & Recommendation Access by Peer-Peer Evidence Distribution).

1.3 Example: Internet Auctions

Internet auction sites such as E-bay are a familiar example of e-commerce based on trust between mutually unknown participants. E-bay works by allowing buyers to provide feedback on sellers (recommendations). In the case of serious complaints the management can act to bar participants from the site.

Consider what would happen if there were no central authority to police the system, no human being in the decision loop for making purchases, and attackers colluding to recommend each other. If we would like our machine to authorise micropayments automatically when we click on links on the web, for example (desirable since popup windows seeking confirmation are tedious and tend to be dismissed without much thought) then this is exactly the case. The rest of this paper addresses this type of scenario. Note in particular that reading E-bay reports on sellers and judging their legitimacy is an AI-complete task, hence difficult to automate!

1.4 Trust Application Framework

The general requirements for applications to which this framework can be applied are that they should consist of pairwise interactions, and that actions have objective success criteria, so we can exchange information about what has happened.

All entities that can exchange information, and it is meaningful to trust or distrust, are **principals**. Principals are pseudonymous; they are authenticated and named typically by public keys, but we don't know their real-world identity. Pairs of principals may perform **actions**, such as exchanging goods or opinions. A trust-based security decision must be made by one party to determine whether to allow each action.

Trust is seen as a quantified predictor of the principal's future behaviour based on **evidence** of the outcomes of previous interactions, both from direct **observations** and **recommendations**. The established trust in a principal is used to determine the likelihood that they will perform a new action in a cooperative manner. This is combined

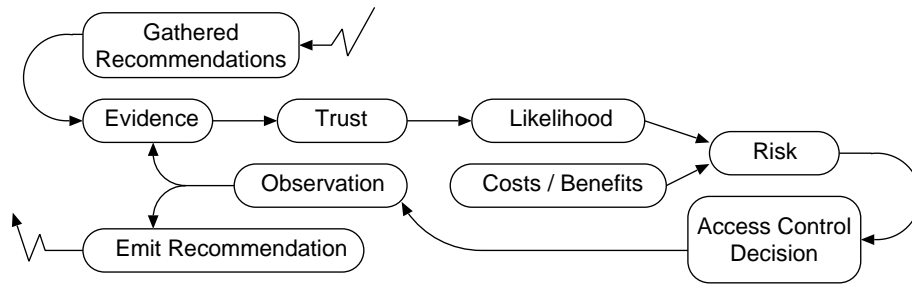


Fig. 1. Framework dataflow

with the cost of any potential negative consequences and a **risk** analysis is done to determine if they may proceed. The dataflow between components is shown in Fig. 1.

2 Background: Overlay Networks

2.1 Distributed Hash Tables

To make trust decisions we need efficient random access to evidence that is widely distributed throughout the network. To solve this problem, it is natural to apply a Distributed Hash Table (DHT), such as Chord [19], Kademlia [15] or Pastry [16]. We therefore pause to consider how these operate. Distributed Hash Tables perform the basic function of mapping keys (associated with data items to be accessed) onto host addresses (such as IP and port number). In our case the data items are collections of trust information and the hosts are members of the P2P network.

In an evidence distribution scenario, it is important that each of the N hosts do not have to be aware of the identities of most other hosts, since that would require $O(N)$ storage per node. Furthermore a steady state is impossible since nodes join and leave the network continuously. Network traffic must be kept to a modest number of messages, both during data lookups and when nodes are added or removed. The system should not require a top-level authority (unlike the DNS, for example).

DHT's have the following useful properties:

- Data lookups take place in $\log N$ steps
- Only $\log N$ storage is required per node for routing tables
- Routing tables can be updated with $\log N$ messages when nodes arrive or depart
- The table can restore consistency after nodes fail silently
- Data item keys are distributed evenly amongst the nodes

The basic operation is to hash keys and node ID's onto the same linear (circular) keyspace, and then place data at the node whose ID is numerically closest to that data's key. The logarithmic properties arise because each node maintains its own small routing table containing complete information about nearby nodes, but also a few routes to more distant nodes. A target can be reached quickly by starting with approximate, large hops and homing in with smaller ones.

To guard against unexpected node failure, data items are actually replicated at a number of locations (typically three or more). If node ID's are essentially random it is convenient to place the replicas on adjacent nodes in the key space so they can all be located with a single search. Replication is also essential when we are storing trust information, as it protects against a single malicious peer changing trust information that it has been assigned to store. The replication factor should be set conservatively to guard against such attacks occurring at the same time as random failures. We will see how to protect against malicious collectives (multiple attackers acting in concert) below.

2.2 Secure DHT's

The problem of making a DHT secure (in the presence of deliberate attacks) has been considered by Sit and Morris [18] and largely solved by Castro et al [5] in the context of secure Pastry (their solutions are also applicable to other DHT's). They identify three requirements: secure node ID assignment, secure routing table maintenance and secure message forwarding.

Secure node ID assignment: The first step to making a DHT secure is to generate node ID's from a hash of the principal's IP address and their public key. Linking node ID's to the public key means that nodes can definitely prove their identity once they have been contacted. Including a hash of the IP address makes it hard for attackers to assume control of chosen parts of the key space, because their IP address is verifiable and they do not have free choice of addresses.

Chord node ID's used in CFS[7] are SHA-1(IP address, virtual node index). Virtual node indices are present to allow for NAT, but must be small integers, which limits a node's choice of Chord ID. The authors observe that "owners of large blocks of IP address space tend to be more easily identifiable (and less likely to be malicious)".

Unfortunately however, IPv6 is likely to give attackers enough IP addresses to exceed the total number of nodes, whereas DHCP (or mobile clients) and NAT cause problems by changing and hiding actual IP addresses. For this reason Castro recommends using Certification Agencies (CA's) that sign certificates binding a random node ID (generated by the CA) to the public key that speaks for it and its IP address. We shall have more to say about this in Section 4 on Identity.

Secure routing table maintenance: In addition to falsifying application data, a malicious node may send incorrect DHT routing information to other participants, for example to divert more queries to nodes it controls. Whilst we cannot prevent this, it is possible to ensure that at most a known fraction of routing table entries are incorrect at any time.

When designing routing table formats there is a tradeoff between allowing them to contain a number of possible alternatives in each slot, versus specifying exactly which routes must be referenced. Some DHT's, such as Pastry, employ the flexibility of the former in order to optimise routes chosen on a geographical ping-time basis (the number of steps is still logarithmic, but it is considerably faster if most hops are not inter-continental). The problem with this is it also gives attackers a lot of flexibility in substituting plausible-sounding next hop addresses which they control.

By contrast, Chord applies strong constraints to routing table entries: they *must* refer to the closest node ID to a calculated point in the space. Note that once contacted a node can prove it is closest by sending its routing table for inspection. Secure Pastry aims for the best of both styles by providing two routing tables; an efficient one, that takes account of geographic proximity, and a constrained one to be used if the former fails to route correctly.

Secure message forwarding: Routing attacks can be prevented by sending requests by several different routes in parallel. In this way we can be assured with suitably high probability that at least one will reach the true destination. Unfortunately the extra messages add considerable overhead. Castro's solution is to first try efficient routing, detect failures, and then use more expensive redundant routing if necessary.

Attacks on routing tables which result in contacting the wrong node can be detected by examining routing tables of nodes that the target claims are nearby. If they agree, a comparison between their proximity in the keyspace with the average network density reveals whether the target has named faraway accomplices rather than true neighbours. It is also necessary to maintain a timer that will fire if routing is not successful for a given time.

3 Computational Trust

The trust and risk models we employ are based on our experience within the SECURE project (Secure Environments for Collaboration among Ubiquitous Roaming Entities) [4, 17, 11].

3.1 Trust Values

Trust values may be stored in arbitrary formats. For example, Yu and Singh [22] use belief and disbelief values from Dempster-Shafer theory to measure trust and recommendations. We believe it is better to store the evidence itself rather than derived trust attributes, since transformations lose information. For this reason we restrict to storing raw observations, or in the aggregate case event frequency counts. Probabilities, confidence intervals and so forth can be derived from the evidence by higher layers; our goal is to distribute relevant information in an efficient and secure way.

In our system, trust values are represented by pairs (m, n) , where m is the number of successful interactions (good outcomes) and n the number of failures (bad outcomes). The use of a DHT gives us sufficient space to record each individual observation, without any need to amalgamate the evidence (requiring summaries). This is because nodes are assumed to have, on average, enough storage to remember everything they have personally observed (multiplied by a small constant factor to handle fault-tolerance replication), and this information is distributed evenly around the entire network by the DHT.

3.2 Recommendations

Formally, a **recommendation** is information passed from principal W, the **witness**, to principal E, the **evaluator**, describing their judgement of principal S, the **subject**.

The use of recommendations increases the amount of evidence we have regarding the subject's expected behaviour, but we must decide if we can trust the advice itself (see the next section on meta-trust).

One solution to scaling the distribution of recommendations is to form recommendation *chains*. Trust chains have been analysed in detail by Jøsang and Gray [13]. The difficulty with recommendation chains is that of actually finding them.

An alternative is to directly look up all reports of interactions with the subject and to make a statistical assessment from this. Even newcomers to the system can then decide who to trust by correlating the opinions of multiple independent strangers. A problem with the statistical method is that we will be fooled if all those who have interacted with the subject so far are their accomplices (a malicious collective).

The essential problem of accessing evidence is that observations are initially indexed by *observer*, and need to be looked up by *subject*. Furthermore if the population is large we can only hope to store partial information at each node.

3.3 Meta-trust

Our **meta-trust** (MT) in another principal measures the accuracy of their recommendations. The value of evidence delivered by a trust chain is **discounted** (scaled down) at each stage in the chain by the meta-trust for each principal in turn. One purpose of meta-trust is to identify principals who perform acceptable interactions, but give misleading positive recommendations to a series of bad collaborators. Without it they could get away with this repeatedly and give a trust "boost" to each bad newcomer. It must be tracked *independently* from ordinary trust so that such behaviour cannot be masked by numerous records of performing good actions.

3.4 Searching for chains

Small world theory has found that, for certain types of graph arising naturally in social networks, short chains of connections exist between any two individuals, and that they can be found efficiently by a greedy algorithm. We investigated whether this can be used to help locate trust chains.

The Freenet information storage system [6] is notable for searching along paths within a graph. It falls somewhere between systems that flood the whole network, and those which deterministically route to the required data. It employs *directed routing* based on nodes' best guesses at where the data will be stored, combined with backtracking if it doesn't happen to be there. Data is cached nearby while retrieving it. The premise is that over time nodes will come to specialise in parts of the keyspace, making searches more efficient. It provides no guarantees, so paths may be long and the system will fail to locate data when the maximum hop count is exceeded.

Guided search algorithms such as Freenet and DHT's work by looking up ID's which have an *order* defined upon them. A structure can be imposed on this which gives a notion of "nearer" and "further away" to guide and prune the search (this is deterministic for DHT's and probabilistic for Freenet). In our case the objects we are looking for are *paths* themselves (we must find a route via the subgraph of trust relationships, the structure of which is outside our control). There is no ordering on paths,

so looking for chains reduces to a graph search problem, which cannot be made to scale. Also, so-called “super nodes” (particularly well-connected principals) do not help unless they are an immediate neighbour of the target – we’re still no closer to knowing which route to leave them by.

N	k	Localities				Unrelated			
		Fail	Obs	d	Visit	Fail	Obs	d	Visit
64K	3	11%	4%	4	29K	10%	0%	5	40K
64K	5	2%	4%	3	17K	1%	0%	3	35K
64K	10	0%	13%	1	5700	0%	0%	2	35K
64K	20	0%	20%	1	2900	0%	0%	2	31K
1M	3	11%	2%	6	489K	13%	0%	6	688K
1M	5	3%	3%	4	274K	1%	0%	4	614K
1M	10	0%	7%	2	62K	0%	0%	3	585K
1M	20	0%	10%	1	24K	0%	0%	2	544K

N = population size, k = average acquaintances, d = search depth,
Fail = chain non-existence, Obs = direct observations, Visit = nodes visited during search

Fig. 2. Searching for chains

We ran simulations to test our conjecture. Fig. 2 shows the results of a breadth-first search for chains between two randomly selected nodes. We used networks with either 64,000 or a million nodes. In each case we varied the average number of direct acquaintances each principal had, from 3 (a sparsely connected network) up to 20 (a well connected network). The columns marked “Unrelated” refer to networks where acquaintances are formed at random, whereas “Localities” are those with a significant amount of clustering (two corresponding bits in the ID numbers of two acquaintances only differ 10% of the time, to be precise). Partners for new interactions were chosen in the same way.

Note that when the average number of direct acquaintances is only 3, the search fails some of the time (no chain connecting the two participants existed). In the case of localities however sometimes there was already a direct observation link between the two participants, making a chain of length one; this never happened with unrelated acquaintances. The average search depth to find a chain is given by d , and the exact number of nodes visited before hitting the target in the “Visit” columns. When nodes are unrelated, this is generally about half the population. In the case of localities it is lower, and more so in graphs with a higher degree of connectivity. Unfortunately even in the best case we still have to visit thousands of nodes before we find a chain.

Our simulation results suggest that short MT chains exist, but can’t be found efficiently. If we look for trust chains we must either flood the whole network or accept a low chance of finding a chain. In the light of these results we have chosen a statistical method for collecting recommendations and evaluating meta-trust called “behaviour profiles”, instead of building chains.

4 Newcomers, Identity and Sybil Attack Protection

Newcomers present a particular problem for recommendation systems. We would like to allow bona-fide newcomers to start to participate, but also without risking an action on an unproven entity. The Eigentrust [14] system solves this by directing 10% of all actions to a pool of newcomers. We could also allow acquaintances to initially vouch for them using pre-trusted peers (statically configured and set per user, not globally, in their policy configuration files).

Friedman and Resnick present a very thorough analysis of “The Social Cost of Cheap Pseudonyms” [10] in the context of game theory. Specifically, they look at a massive online pairwise prisoners dilemma with a changing set of pseudonymous players, some of whom may be irrational. It is assumed that the whole interaction history is common knowledge. They note that suspicion of strangers is costly to society, but prove that distrust of newcomers is an inherent cost of easy identity changes.

4.1 Sybil Attacks

Untrustworthy newcomers are not too serious a problem if there are only one of them per attacker. A Sybil Attack [9] is a situation whereby a single malicious participant creates multiple apparently unrelated identities and uses them in concert to defeat the system. It occurs whenever there is pseudonymity and no cost associated with the formation of new unlinked identities.

We might consider placing restrictions on principals who have recently joined the system (in a chronological sense); however a “Rolling Sleeping Army” attack may then be employed. In this situation, an attacker creates a large number of new identities each day, continues until the first set of them are eligible to perform risky actions and then deploys them in their respective batches every day from then on.

A group of identities under the control of a single real-world entity is described as a **collective**. In our experiments, for simplicity, the behaviour of each member of a collective follows a uniform policy. We assume however that it is impossible to identify collectives by spotting patterns, since in reality a large enough group could configure its interactions in arbitrarily complicated and realistic-looking ways, so as to resemble a normal part of the population.

Behaviour profiles are directly susceptible to Sybil attacks. MT chains are also vulnerable since new principals may initially agree with your opinions in order to build meta-trust chains, and then start to mislead.

4.2 Certification Agencies

A Certification Agency (CA) is a special trusted entity which signs others’ public keys to show they are valid for use in the system. Note that a CA is a central point of failure, and hence undesirable.

There is an emerging consensus that Certification Agencies in some form are required to counter Sybil attacks. Douceur [9] has established that in the absence of CA’s a Sybil attack can undermine *any* recommendation system. It should be noted that CA’s

may be *implicit* (for example CFS uses IP addresses; DNS approaches implicitly rely on ICANN as the trusted agency) as well as explicit (such as Verisign).

Castro [5] notes that Sybil attack protection can be achieved by charging for identity certificates, or binding them to real-world identities, and suggests that in practice different forms of CA are appropriate for different situations.

Friedman and Resnick [10] recommend free but unreplaceable pseudonyms which they call “once in a lifetime ID’s”. A CA is shown proof of real-life identity before issuing a single corresponding ID. Anonymity is preserved with blind signatures, so that even the CA itself does not know the mapping between real identities and keys, but can ensure it is 1-1. Names are valid within different “arenas” (namespaces used for different types of application); this is a tradeoff between accountability and anonymity. The authors suggest auctioning off the ID server franchise.

Before returning to Certification Agencies we first consider some possible alternatives for regulating the creation of new identities.

4.3 Rate Limits

There have been several proposals for ways to restrict the *rate* at which attackers can generate new identities. A feasible rate limit would be very useful as we could then quantify how much damage a limited proportion of attackers can do, and express this as an overhead loss expected by legitimate principals.

Hash Cash [3] addresses this by making it computationally expensive to generate ID’s. To do so one is required to find a partial hash collision, i.e. two numbers which match in a given number of bits after passing through a one-way function. One number can be derived from your e-mail address (or in our case, public key, to preserve anonymity) so that others can’t reuse it. Douceur [9] also considers other types of resource game, including those which require bandwidth or a lot of storage space.

The problem with all such approaches (taking computation power as an example) is that the puzzle must be easy enough for the slowest legitimate node to solve in a reasonable time, but hard enough to seriously slow down a determined attacker. It does not seem likely that this will be possible, particularly given the disparity between resource-light mobile nodes and the cost-effectiveness of purpose-built crypto hardware.

Another possibility suggested by Eigentrust [14] and others is to require the user to solve a CAPTCHA [20] to receive an ID. This is a task such as obfuscated picture recognition that a human can perform which requires too much AI for a computer to solve, thereby preventing automated attacks. The difficulty with this is that a central component is required to administer the test; it is unclear how it could be used in a distributed fashion.

For these reasons we are not convinced that an effective rate limiting solution for identity generation exists at present.

4.4 Entry Fees

Friedman and Resnick [10] describe several techniques for charging newcomers an entry fee in order to join the system.

Payment can be made to a special agency or distributed amongst the whole population, for example. Of course the former is not a decentralised solution, and takes utility away from the participants. The shared-fees method however leads to entities lurking in the system in a dormant state simply to collect entry fees.

The biggest problem is trying to find an “optimal entry fee” when payoffs are heterogeneous. One is faced with a choice between high registration fees, which discourage poor participants, or the need to set low maximum transaction amounts to keep them below the value of an identity.

An alternative to entry fees is for newcomers to “pay their dues” by accepting poor treatment from principals with established positive reputations. For example they may have to pay up front, accept shipping delays or higher prices. One problem with this is that for many applications it is hard to pay “in kind”. A scenario may have arbitrary non-monetary costs, for example. Another serious issue is that collaborators might “pay their dues” to each other.

4.5 ENTRAPPED approach

Our approach is to make large quantities of fake ID’s expensive, without needing to charge for a reasonable number or limiting legitimate participants to a single ID. These might be described as “few in a lifetime ID’s”

We achieve this by allowing existing well-known companies, whom the participants have a prior business relationship with, to act as (multiple) Certification Agencies. They have already seen participants’ real IDs and need not charge a fee since this can be viewed as a “value-added” service. CA’s use blind signatures to preserve anonymity. Attackers will find it increasingly expensive to obtain more identities, since each CA will only issue one and they will run out of easy-to-obtain sources.

Typical companies might include ISP’s, banks, phone companies, utility companies, the post office, vehicle licensing agency, passport agency, solicitors, Verisign, VISA and large shopping chains. The cost to attackers is initially due to the need to open accounts with companies or pay registration fees, and eventually becomes the cost of fabricating a complete new identity (forged passport etc) in order to access another batch of ID’s.

Making some identities free is useful because it ensures a low cost of entry. Allowing users to have multiple ID’s is very important since they can use different personas for sensitive topics, such as commenting on a political forum that their employer or state might disapprove of. Users may choose how they wish to partition their own identity since we do not enforce particular namespaces on them.

The restriction on CA’s issuing a second ID is time-based, for example one per year. This allows legitimate users to discard and eventually replace their ID’s if they get stolen, say. Attackers can get replacement ID’s too but only at a very slow rate.

Identity Presentation Protocol An ENTRAPPED identity is a vector, which allows signatures from multiple CA’s to be presented at once. Each principal also has a list of CA’s which they accept as valid. The following process describes what happens when an identity is presented to another principal:

1. The requester’s public key has been signed by various CA’s.

2. CA's publish nominal value pairs (registration fee, cost of real-world credentials).
3. The recipient masks off CA's he/she doesn't recognise.
4. Component CA values are adjusted according to local policy (maximum amounts, etc) and then a combined ID value function is evaluated. This could be a simple sum of the components, or it might be superlinear since extra ID's are progressively harder to find. It could also insist that at least two or more components match.
5. This value is the cost to the requester if their identity becomes unusable due to low trust and MT. If non-zero, it indicates the maximum recommended transaction value.

The third step is most important – a match is made between the CA's offered and those recognised by the recipient. We anticipate a market for identity servers; agencies will try to become sufficiently well known that the chance of being mutually listed is high. To facilitate commerce we also recommend an API for principals to publish which CA's they accept. This gives requesters a clue as to which identities they need to acquire, and lets friends learn what is considered valuable or reputable.

ENTRAPPED is suitable for a range of transaction sizes from tiny (micropayments) up to medium value (a few \$100). Free transactions are not relevant, because there is no risk. Large transactions are unsafe since they may exceed the value of identities, but fortunately users are generally happy to use conventional credit card mechanisms for these, since convenience is not so important.

We also envisage *Transaction Guarantors*, which are Certification Agencies that also provide *insurance* for actions, up to a transaction limit. Care must be taken over timing attacks – distributed locks are needed on identities during transactions. Again, participants will prefer well-known but also lower commission agencies.

5 The ENTRAPPED Platform

We now describe our solution for distributing trust information. Our implementation is built from four main components; a Distributed Hash Table, a Statistical Correlator, the SCOP[12] events system and a distributed database of tables stored at each node.

5.1 State Tables

ENTRAPPED is designed to work with any DHT. We prefer Chord or similar, due to the availability of neighbour sets for efficient replica location and the strong security constraints on routing table entries. Three-way application-level replication and comparison is used to detect attempts by an attacker to substitute false evidence for a given key (if they are lucky enough to control the node it hashes to). Each node is responsible for storing seven different tables; their own personal observations together with behaviour and meta-trust profiles for three other principals. The DHT ensures that a node cannot predict which subjects they will be maintaining profiles for.

Personal observations are the most direct and hence most useful source. If they exist they are used first. Recommendations are fetched from behaviour profiles and weighted equally for each recommending principal, except those which are eliminated

by meta-trust. Meta-trust profiles tell us what is known about the accuracy of the recommendations from a particular principal. They are replicated and distributed in a similar manner to behaviour profiles. Information from meta-trust profiles is fed back to the “global outlier” column in behaviour profiles.

The format of behaviour profiles is shown in Fig. 3. Each behaviour profile stores a complete dossier about a given principal. For example, Alice’s behaviour profile contains recommendations about Alice made by everybody who has interacted with her. This table is the only information which is needed when one makes a decision about whether to perform an action with Alice.

In the example table, principal David has been detected as a local outlier given his unusual success rate of 95% (suggesting he is an accomplice of Alice). Emily isn’t an outlier with regard to Alice but her meta-trust profile has reported she was unreliable elsewhere, so we have marked her as a global outlier. The remaining recommendations from Claire, Bob and Fred lead to an overall trust value of 6.7%.

Subject	Witness	Action	Success rate	Local Outlier	Global Outlier
ALICE	EMILY	ECOM	50%	0	1
ALICE	CLAIRE	" "	10%	0	0
ALICE	DAVID	" "	95%	1	0
ALICE	BOB	" "	0%	0	0
ALICE	FRED	" "	10%	0	0
ALICE	Overall	ECOM	6.7%		

Fig. 3. Behaviour Profile Format

5.2 Data exchange between tables

Fig. 4 shows how data is exchanged between the three different sorts of table. A row is copied from an observation table to the relevant behaviour profile locations whenever a new observation occurs. The destination is looked up using the DHT so this is a point-to-point operation (in fact there are three replica destinations, of course).

Whenever the data in a behaviour profile changes, the rows with the same action type are compared to detect outlying recommendations. For example, most of the witnesses might report a high proportion of successes but one witness may state that they had many interactions with the subject all of which were failures. Such a witness is an outlier and is marked as such in the Local Outlier column.

When the set of local outliers in a behaviour profile changes, the row in question is sent to the three locations of the meta-trust profile for that witness. The meta-trust profile is used to calculate *global outliers*. This is a more accurate measure of whether

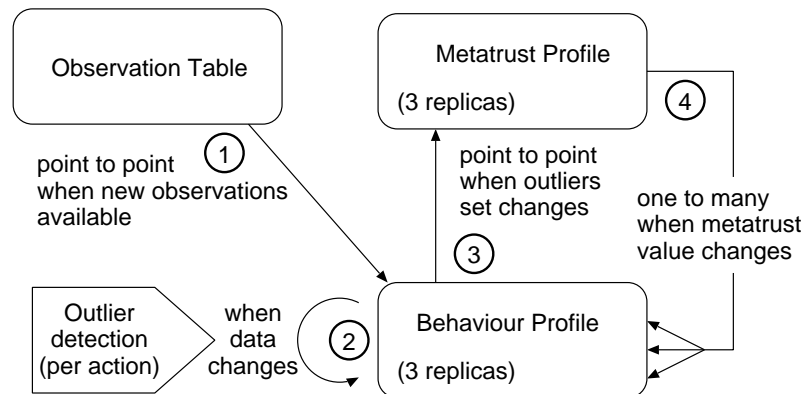


Fig. 4. Dataflow between Tables

recommendations should be believed and it is actually the global outlier state that is used to determine which rows in the behaviour profile are included in the overall summation.

5.3 Implementation

Fig. 5 lists the participants in a small test run of 24 named principals. For this test the 72 trust tables all ran in separate processes on a single machine (although they could have been distributed, of course). Our activity generator assigns different “personalities” to each principal, which govern their behaviour. The third column of the table lists the trust values computed by the system for each individual after approximately 2500 random interactions have taken place.

As expected, good principals have very high trust values and bad ones very low. In some cases they are not precisely 0 or 100%; this is because *chaotic* principals introduce erroneous (random) recommendations. *Careless* principals are configured to behave badly one time in ten, which leads to trust values of approximately 90%.

The right part of Fig. 5 shows the observation table for principal Joe. Joe’s personality is good; note he has encountered multiple failures dealing with Andy, James and Martin (all of whom are bad) and occasional ones with Alice and Lucy (who are both careless).

Fig. 6 shows the behaviour profile for principal Gray. Gray is colluding with Henry and Jessica and indeed they are the only principals who have reported multiple successes interacting with him. Vicky has reported one success erroneously due to her chaotic nature. All three have been detected as outliers, however, as indicated by the “Local” (outlier) column. They are therefore not included in the trust value computation for Gray which is why his value in Fig. 5 is reported as an unambiguous 0%.

Principal	Personality	Trust value
Alice	Careless	87.5 %
Andy	Bad	0.7 %
Becky	Careless	91.0 %
Bob	Good	98.8 %
Cathy	Chaotic	91.5 %
Claire	Tricky	50.4 %
Dom	Good	98.7 %
Gray	Colluding	0.0 %
Henry	Colluding	0.0 %
James	Bad	0.0 %
Jessica	Colluding	33.3 %
Joe	Good	100.0 %
Karen	Good	98.8 %
Laura	Good	98.7 %
Liz	Good	100.0 %
Lucy	Careless	92.0 %
Mark	Chaotic	88.9 %
Martin	Bad	1.2 %
Paul	Tricky	49.5 %
Rachel	Good	98.8 %
Ruth	Good	100.0 %
Sarah	Good	100.0 %
Vicky	Chaotic	90.9 %
Victor	Good	98.8 %

MOBS Viewer window showing observation table for Joe. The window title is "MOBS Viewer" and the subject is "Joe". The table has columns: Subject, Witness, A, Success, and Failures. The "Obs" radio button is selected.

	Subject	Witness	A	Success	Failures
1	Andy	Joe	E	0	6
2	Ruth	Joe	E	10	0
3	Paul	Joe	E	5	0
4	James	Joe	E	0	6
5	Cathy	Joe	E	4	1
6	Dom	Joe	E	1	0
7	Claire	Joe	E	7	0
8	Victor	Joe	E	1	0
9	Martin	Joe	E	0	7
10	Rachel	Joe	E	6	0
11	Bob	Joe	E	9	0
12	Lucy	Joe	E	4	1
13	Mark	Joe	E	3	0
14	Laura	Joe	E	4	0
15	Becky	Joe	E	5	0
16	Sarah	Joe	E	5	0
17	Vicky	Joe	E	9	0
18	Liz	Joe	E	7	0
19	Karen	Joe	E	4	0
20	Henry	Joe	E	0	1
21	Alice	Joe	E	4	1
22	Gray	Joe	E	0	1

Fig. 5. Personalities and Observation Table Snapshot

MOBS Viewer window showing behaviour profile for Alice. The window title is "MOBS Viewer" and the subject is "Alice". The table has columns: Subject, Witness, A, Success, Failures, Local, and Global. The "Rec" radio button is selected.

	Subject	Witness	A	Success	Failures	Local	Global
1	Gray	Dom	E	0	1	0	0
2	Gray	Jessica	E	24	0	1	0
3	Gray	Ruth	E	0	1	0	0
4	Gray	Henry	E	26	0	1	0
5	Gray	Mark	E	0	1	0	0
6	Gray	Victor	E	0	1	0	0
7	Gray	Vicky	E	1	0	1	0
8	Gray	Andy	E	0	1	0	0
9	Gray	Lucy	E	0	4	0	0
10	Gray	Claire	E	0	1	0	0
11	Gray	Becky	E	0	1	0	0
12	Gray	Karen	E	0	1	0	0
13	Gray	Joe	E	0	1	0	0
14	Gray	Martin	E	0	1	0	0

Fig. 6. Behaviour Profile Snapshot

5.4 Action Correlation

The power of trust-based access control is increased if we can use evidence gathered from one application to bootstrap trust in another. For example, trust to access a library could be initialised from other library domains, from qualifications, or reports from employers. A car hire firm could use recommendations from other car hire companies, from equipment hire, driving test reports, car insurance companies, credit and police records.

Trust from related applications is particularly useful when a principal performs an action they haven't done before. Otherwise we would have to conclude there is no relevant evidence and fall back on insurance or the cost of identity creation.

We support action correlation by distributing evidence for different applications within the same framework. Principals can specify in their context policy which actions they consider relevant to others. They may choose to publish this (which is similar to saying that access to a resource can be granted if one presents good results in certain qualifications). They may also use data mining on the raw evidence to generate context policies if desired.

We considered, but rejected, automatically judging which actions are similar via statistical correlation of observations. Each behaviour profile would contribute one summary line to a notional global action correlation matrix, with a column for each type of action. Unfortunately, to perform Pearson linear regression tests on this matrix would require Nt storage and Nt^2 computations for N nodes and t action types. Additionally, since we can't appoint a central authority to do it, the all-all communication involved in collecting this information would overload the network.

6 Applications

The ENTRAPPED architecture has wide applicability; to demonstrate this we briefly describe two broad application areas for which these techniques are appropriate.

6.1 Collaborative Online Guidebooks

Our first application area is the general category of collaborative review systems, in which contributors submit ratings of places, products and so on. These might include movies, books, music, websites and so on. Our work has concentrated specifically on an Augmented City Guide [11] application in which users can rate Restaurants, Shops, Attractions and Amenities. Virtual post-it notes are placed around the city describing different locations.

Any number of people can rate each place; when we visit somewhere new the system must select the reviews that are most likely to be reliable and informative, based on their contributors' past history. The trust model is used to determine this. Selected notes are then displayed to the user.

ENTRAPPED can only be used for *objective* measurements. This makes it suitable for modelling review accuracy but not the personal taste of other principals (in that case the comparison algorithm to find outlying recommendations would be meaningless).

As a result we do not assess personal preferences, unlike some music recommendation systems, for example.

The application has one action which is deciding whether to display a note by a given author (effectively the subject principal in the “interaction”). If we choose not to, there is zero cost. If the note is displayed, the outcomes are that it may be acceptable or inaccurate; costs depend on the principal’s aversion to the risk of misleading information.

In this scenario principals are likely to have different degrees of reliability when assessing different features (architecture versus restaurants, for example). This is accommodated by classifying topics into separate but related actions. An action correlation matrix is used to derive initial trust when principals review previously unobserved categories.

6.2 P2P Distributed Backup Service

A very promising application of peer-to-peer networks is to provide a distributed, redundant file backup service. Clients can choose which server(s) to entrust their data to, within cost or quota constraints. We assume data is encrypted so that privacy is not an issue.

The risk analysis for distributed backup is atypical; instead of a pairwise decision it is a general optimisation problem to determine which server(s) should be chosen, to minimise the chance of any data becoming unrecoverable. We may need to trade storage space against additional replicas for safety. Also we may assign different values to different types of data.

Despite the more complex risk assessment, we can still use our evidence distribution framework to determine server reliability. Detailed observations can be gained from periodic probing, not just in the rare events when we need to actually retrieve data.

Measurements of interest include *availability* (chance of file retrieval in 15 mins), *failure probability* (chance of no file retrieval within 24 hours) and *access speed* (average bytes/sec). These can all be tested and communicated via the evidence distribution mechanism. In practice a backup policy would then be used to weight and combine these parameters, together with *server persistence* (its longevity since joining the system) and *service cost* before making a decision.

An interesting possibility for making decisions involving *comparative risk* (when we may choose between different candidates to interact with) is to pick hosts probabilistically, so that the more reliable ones have the greatest chance of being chosen and the others a smaller chance. Comparative risk could also be appropriate for other applications, such as multiplayer gaming.

7 Evaluation

7.1 Malicious Collectives

Fig. 7 shows how our system exposes malicious collectives over time. For this test we created 10000 principals of which 2000 are hostile. The graph illustrates the number

of successful attacks that have occurred after a given number of random interactions. We also varied the maximum allowed size of malicious collectives from 8 to 64 principals. The number of malicious principals in all cases total 2000, so for example with a collective size of 16 these are split into 125 independent colluding groups.

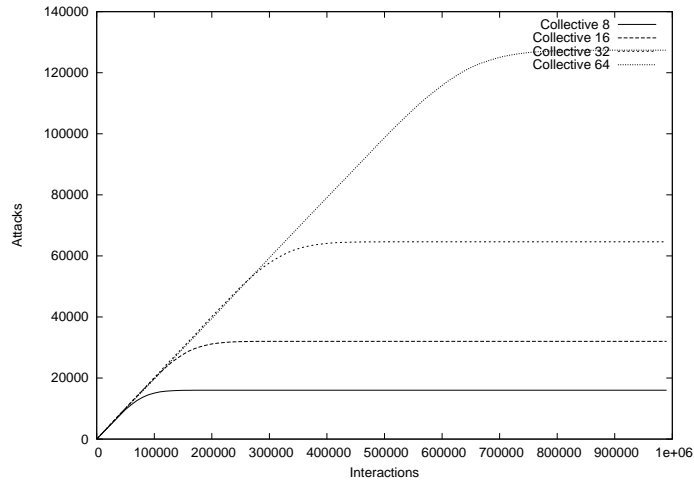


Fig. 7. Time evolution of attacks from malicious collectives

The graph shows that in all cases attacks start at a linear rate of 20%, since none of the attackers have been detected. It starts to level off when there are enough legitimate observations to identify some of the collectives, and not long thereafter the attacks halt altogether, since all the attacking identities have been exposed. Note that for the same number of hostile principals it takes longer to detect them (hence, more successful attacks) if they are organised into larger collectives.

7.2 Threat analysis

We now describe our threat model by enumerating a large number of different attacks, with some notes as to how they can be neutralised. The attacker knows the parameters of the system and can tune his parameters to manipulate trust and meta-trust. We cannot prevent all attacks from succeeding, but we can ensure that all attacks cost an attacker more than they benefit them (so it isn't possible to make a profit by cheating).

Human error is a major problem with security systems, especially if users must create their own policies. ENTRAPPED helps to avoid this because the main trust components are shared; everyone is working from the same data, hence although decisions are made autonomously the same core policy can be used for most clients. Examples of user-controlled configuration parameters are pre-trusted peers and risk sensitivity.

Software error is a problem which scales based on the size of the TCB. We can counter it with simplicity and by providing strategies for recovery in the event of an exploit. The timeouts on user identities help ensure they can be replaced if compromised.

Identity theft has a well-defined meaning here – it is achieved by breaking into another machine and harvesting the private key (we assume passphrases are not used, or if they are can be obtained as well by keyboard sniffers).

Routing attacks target the P2P overlay network, and are countered by the use of a secure DHT.

Bad guys are principals which always behave badly, making them the easiest kind of attacker to spot!

The **Newcomer attack** consists of a principal who joins the system, performs a single bad interaction, and then leaves. A stronger version of this is the **Basic Sybil attack**, in which the attacker employs a long series of badly behaved principals with throwaway ID's. We counter widespread adoption of both of these by economic means, due to “few-in-a-lifetime” certified identities.

The **Waiting attack** consists of performing a series of good interactions in order to build up a positive trust value, and then cheating. A stronger version is the **Oscillation attack**, whereby the principal switches between well-behaved and hostile modes in an attempt to manipulate its own trust value. In a variant which we call the **Mixed behaviour attack**, the principal chooses a mode (cooperate or defect) for each separate interaction probabilistically. Again the objective is to manipulate their trust value to stay just below the threshold for identification as a bad guy.

This has been studied in the context of the Eigentrust [14] project, in which the most effective probability of defecting was found to be 50% (in this case their simulation measured that 28% of all interactions in the system were actually successful attacks). In defence this strategy comes at a cost to the malicious nodes, since performing some good interactions may be undesirable for them; however this is application-dependent and will not always be the case.

Misconfiguration and **Chaotic behaviour attacks** lead to either random or constant, incorrect behaviour.

Carelessness is not strictly speaking an attack, but may be even more common in practice. Careless principals are generally good but suffer from “trembles”, which means they will occasionally behave badly by mistake (at random). Our system is able to distinguish clearly between careless and bad principals (in fact in the experiment of Fig. 7 the “good” principals were all modelled as careless 10% of the time).

Peer-specific attack: Pseudonymity should prevent an attacker matching the real identity of principals and hence attacking a named target (but see concerns under privacy attack below). However an attacking node might try a policy which involves cheating only a subset of other principals (say arbitrarily those whose ID's equal 0 modulo 4), so that the rest give it good recommendations.

A Collusion Clique is an attack on the recommendation system using false praise. All principals in a clique behave badly, but provide artificially good recommendations for each other.

Collusion with Supporters is a more subtle approach whereby just one principal behaves badly, and the others in the collective simply recommend it (they may themselves not perform any actions or perform good ones, depending on the application). Note that systems which do not have a separate meta-trust concept such as [14] never discover supporters, making this attack more effective.

Collusion with Camouflage is a combination of supporters and mixed behaviour in which the active principal only misbehaves some of the time (and the supporters not at all) in order to try and escape detection whilst making a profit.

Defamation consists of attacking a good principal's reputation. This attack is one reason why we cannot simply treat any negative recommendations as proof that the subject is bad (another reason is trembles, as described under carelessness above).

An **Indirect Sybil attack** involves a stream of colluding recommenders boosting the trust of one badly behaved principal; A **General Sybil attack** might include many teams of colluding recommenders and arbitrary numbers of badly behaved principals. These are both limited by identity certification.

Privacy attack: Behaviour profiles reveal a great deal of information about the interaction pattern for a principal. If they can be correlated with some external piece of known data (perhaps a shipment of a certain size last week, or an expected pattern of suppliers), a correspondence with a real identity might be established. Even without this, anonymous "shopping pattern" information can be collected, correlated with known demographics and principals treated differently on this basis.

ENTRAPPED provides a partial defence against privacy attacks by masking off principal ID's in behaviour profiles when they are made visible to the network. The statistical trust value calculation only requires that users know the pattern of interactions, not who observed them. Only the three replica nodes who are randomly assigned a given behaviour profile can see the real node ID's. Data integrity is ensured since the identity-masked copies can be compared; if the pattern is not the same then tampering has occurred.

Denial of service attacks are possible at the network layer, DHT layer, evidence distribution layer, and/or application layer. Care must be taken so that DoS against a third party cannot mislead a target and cause incorrect decisions to be made, as well as merely preventing actions.

We have already seen that **Resource costs** (CPU, storage, connectivity) for each node are acceptable. A larger issue is that of **Deployment costs**. In particular a critical mass user-base may be required for adequate functioning. Our system helps address this by warning when there is insufficient evidence to make a decision.

8 Related Work

Abdul-Rahman and Hailes [1] have identified similar concepts to trust and meta-trust, as well as recommendations and trust categories (their name for trust in different actions). However they have a very basic notion of trust value, with only a few possible states, and the lack of an efficient distribution system makes their trust chains and revocations unconvincing.

Aberer and Despotovic [2] present an efficient, scalable peer to peer evidence locator, based on their "P-Grid" data structure. They monitor "complaints" between principals which have interacted and provide a number of replicas of the information tuned to provide acceptable fault-tolerance. Unfortunately they don't consider routing attacks or different types of action, and don't distinguish trust and meta-trust.

The work by Xiong and Liu [21] is similar to our own. They advocate the use of an overlay network to locate what we call behaviour profiles, in a peer to peer environment. Their implementation uses a P-Grid. They also recognise that recommenders should not be believed to an equal degree, but rely on a cache for this information rather than distributing meta-trust.

The Cooperative File System [7] (CFS) is not a trust-based system, but has many relevant properties. It is built on a DHT (Chord). Files are divided into blocks which are replicated on several adjacent successors of the key value. CFS doesn't use CA's, so disk quotas are enforced per client IP address. Each server applies the quota rule independently, so the maximum storage per IP address increases linearly with the number of CFS servers. This could be fixed by applying an adaptive limit based on the number of servers in the whole network. CFS estimates the total number of nodes by extrapolating from the nearby node density in the ring.

The EigenTrust [14] project is closest to our approach, in particular using behaviour profiles in much the same way. This is done in the context of a file-sharing application. It solves the problem of finding trust chains (transitive trust) by iteratively multiplying a global trust matrix until it converges. We believe this creates a scalability problem, however. Everyone has to participate in calculating the global trust in lock step, which creates inter-dependency and a lot of communication. No distinction is made between trust and meta-trust; the algorithm requires a set of globally pre-trusted peers to break attacks by malicious collectives.

9 Conclusion

We have created a mechanism for the distribution of trust information, using a Distributed Hash Table to store statistical *Behaviour Profiles*.

The system returns all that has been observed about a given principal in $\log N$ steps. All trust values are pre-computed and do not require a process of convergence over multiple iterations. There is a distributed, shared evaluation of the accuracy of recommendations (meta-trust). Trust and meta-trust are represented distinctly.

The system does not depend on caches (which require locality of reference amongst accesses) or trust chains, which we have shown cannot be found efficiently. It is scalable to very large populations: there's no centralised component, no multicast messages, and the storage requirements per node are modest.

The system applies to any kind of evidence-based application. Certification Agencies are used to protect against Sybil attacks by creating "few in a lifetime ID's". Multiple CA's are supported via vector ID's. Finally it has been designed to be reliable in the presence of a range of deliberate attacks, and to protect participants' privacy.

Acknowledgements: This work was funded by the European Commission Information Society Technologies SECURE project, IST-2001-32486. I would like to thank my colleagues at the University of Cambridge Computer Laboratory, at BRICS in the University of Aarhus, the Trinity College Dublin DSG, Université de Genève Object Systems Group and SmartLab at the University of Strathclyde.

References

1. A. Abdul-Rahman, S. Hailes. "Using Recommendations for Managing Trust in Distributed Systems". *Proceedings of the IEEE Intl. Conference on Communication*, Malaysia, November 1997.
2. K. Aberer, Z. Despotovic. "Managing Trust in a Peer-2-Peer Information System". *Proceedings of the 10th Intl. Conference on Information and Knowledge Management*, 2001.
3. A. Back. "Hashcash - A Denial of Service Counter-Measure".
<http://www.hashcash.org>
4. V. Cahill, et al. "Using trust for secure collaboration in uncertain environments". *IEEE Pervasive Computing*, 2(3):52-61, August 2003.
5. M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D. Wallach. "Secure routing for structured peer-to-peer overlay networks". *Proceedings of the 5th Usenix Symposium on Operating Systems Design and Implementation*, Boston, December 2002.
6. I. Clarke, O. Sandberg, B. Wiley, T. Hong. "Freenet: A Distributed Anonymous Information Storage and Retrieval System". *Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.
7. F. Dabek, F. Kaashoek, D. Karger, R. Morris, I. Stoica. "Wide-area cooperative storage with CFS". *Proc. 18th ACM Symposium on OS Principals (SOSP '01)*, October 2001.
8. N. Dimmock, I. Maddison. "Peer-to-Peer Collaborative Spam Detection". *ACM Crossroads Magazine*, Dec 2004.
9. J. Douceur. "The Sybil Attack". *Proc. 1st Intl Workshop on Peer-to-Peer Systems (IPTPS'02)*, March 2002.
10. E. Friedman, P. Resnick. "The Social Cost of Cheap Pseudonyms". *Journal of Economics and Management Strategy* 10(2): 173-199, 2001.
11. D. Ingram. "Trust-based Filtering for Augmented Reality". *1st Intl. Conference on Trust Management*, Crete, May 2003.
12. D. Ingram. "The SCOP Events Library".
<http://www.srcf.ucam.org/~dmil1000/scop/index.html>
13. A. Jøsang, E. Gray, M. Kinatader. "Analysing Topologies of Transitive Trust". *Proc. Workshop Formal Aspects of Security and Trust (FAST)*, September 2003.
14. S. Kamvar, M. Schlosser, H. Garcia-Molina. "The EigenTrust Algorithm for Reputation Management in P2P Networks". *Proc. 12th Intl WWW Conference*, May 2003.
15. P. Maymounkov, D. Mazières. "Kademlia: A Peer-to-peer Information System Based on the XOR Metric". *Proceedings of the 1st Intl. Workshop on Peer-to-Peer Systems*, March 2002.
16. A. Rowstron, P. Druschel. "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems". *Proceedings of the 18th Intl. Conference on Distributed Systems Platforms (Middleware)*, Germany, November 2001.
17. SECURE: Secure Environments for Collaboration among Ubiquitous Roaming Entities. *EU Project IST-2001-32486*, December 2002. <http://secure.dsg.cs.tcd.ie/>
18. E. Sit, R. Morris. "Security Considerations for Peer-to-Peer Distributed Hash Tables". *Proc. 1st Intl Workshop on Peer-to-Peer Systems (IPTPS'02)*, March 2002.
19. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, H. Balakrishnan. "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications". *Proceedings of the ACM SIGCOMM Conference*, San Diego, August 2001.
20. B. Watson. "Beyond Identity: Addressing Problems that Persist in an Electronic Mail System with Reliable Sender Identification". *1st Conference on Email and Anti-Spam (CEAS)*, 2004.
21. L. Xiong, L. Liu. "Building Trust in Decentralized Peer-to-Peer Electronic Communities". *5th International Conference on Electronic Commerce Research*, October 2002.
22. B. Yu, M. Singh. "An Evidential Model of Distributed Reputation Management". *Proc. 1st Intl. Joint Conference on Autonomous Agents and MultiAgent Systems*, Italy, July 2002.