Towards a Framework for Assessing Trust-Based Admission Control in Collaborative Ad Hoc Applications

Elizabeth Gray*, Paul O'Connell*, Christian Jensen†, Stefan Weber*, Jean-Marc Seigneur*, and Chen Yong*
*Distributed Systems Group, Department of Computer Science,
Trinity College, Dublin 2, Ireland
Email: {grayl, sweber, seigneuj, cheny}@tcd.ie
† Informatics and Mathematical Modelling, Technical University of Denmark,
Richard Petersens Plads, Building 322, DK-2800 Kgs. Lyngby, Denmark
Email: cdj@imm.dtu.dk

Abstract

The proliferation of mobile devices and the development of ad hoc networking technologies has introduced the possibility of a vast, networked infrastructure of diverse entities partaking in collaborative applications with each other. However, this may require interaction between users who may be marginally or completely unknown to each other, or interaction in situations where complete information is unavailable.

Humans use the concept of trust to help decide the extent to which they cooperate with others. It provides a mechanism for lowering access barriers and enables complex transactions between groups. Trust, however, takes many different forms and is difficult to stringently define or understand.

The aim of our work is to develop a trust framework that enables access control based on trust-based admission control policies that define the trust relationship between entities in collaborative ad hoc applications. We do this by integrating a trust-formation element into an admission control mechanism to manage interaction between previously unknown users. A simple distributed blackjack card game application implements the trust-based admission control system to assign roles to users according to their trust-based admission rights.

1. Introduction

The proliferation of mobile devices and development of vast ad hoc networks introduces the possibility of an environment where multitudes of diverse entities will partake in collaborative applications with each other. A mobile ad hoc network is an autonomous system of mobile entities connected by wireless links. The entities are free to move randomly and self-organising, so the network is highly dynamic and subject to rapid and unpredictable changes. As in traditional networks, access to collaborative resources in mobile ad hoc networks requires varying levels of control. Difficulties arise, however, when traditional access control methods are applied in a decentralised collaborative ad hoc environment. For example, in traditional groupware applications, access to a group is controlled by an administrator with a predefined list of possible group members. The administrator grants access rights based on whether the requesting entity is identified as meeting the appropriate criteria. However, in a vast networked infrastructure of diverse entities partaking in collaborative applications with one another on an ad hoc basis, access control based on the centralised administrator model becomes ineffective.

For example, suppose Alice takes the 8am commuter train into the city to work every weekday morning. To pass the time, she joins an ad hoc wireless network to see what collaborative gaming applications are available. She discovers an ongoing blackjack session in which Bob is the dealer, and she requests admission to the game. To Bob, Alice is an unknown entity, who may or may not be trusted to behave correctly, i.e. pay her debts, if given access to his game. In the traditional model, Bob would be able to contact a centralised administrator to determine if Alice should have access rights to participate in the blackjack game. Moreover, making the traditional model work in this scenario would require a global authorisation structure, since Alice and Bob may live in different countries and only be meeting by chance. This approach does not scale to the large, dynamic, decentralised ad hoc networks envisioned.

Trust management systems such as KeyNote [1], PolicyMaker [2], Simple Public Key Infrastructure (SPKI) [3], and Simple Distributed Security Infrastructure (SDSI) [4] attempt to manage security in large-scale distributed networks through the use of credentials that delegate permissions. However, even these systems falter in ad hoc networks where it would be virtually impossible to maintain any centralised record of authorisation credentials.

Moreover, neither traditional security approaches nor trust management systems enable an independent entity to answer one fundamental question: '*Why* do I trust this unknown party to perform this action?' That is, what does trust *really* consist of?

Trust has been well-researched in another kind of massive ad hoc collaborative network infrastructure. Humans must regularly determine, with no assistance from a trusted third party, how to interact with known and unknown people. Trust provides a mechanism for lowering access barriers and enables complex transactions between groups. Humans use the concept of trust to help decide the extent to which they cooperate with others in these types of situations, where complete information is not available.

Human trust is a difficult concept to formalise, and many different definitions of trust exist. We believe, however, that, for the computing world, some definition of trust can be formalised such that groupware is possible in which principals are able to manage dynamic ad hoc access to their resources by implementing a trust formation and evaluation process. Each principal would be provided with a mechanism that allows him to establish a trust value for other potential group members.

State-of-the-art in dynamic trust formation is currently too imprecise to suggest that trust could be used to decide access rights at a low level. Instead, we propose to use trust-based admission control together with standard role-based access control. In this context, high level admission control polices parallel human decision-making behaviour in trust-based situations where complete information is unavailable. Using a policy-based approach instead of a traditional approach based on low level access rights maps better to the ad hoc network environment, where access control decisions must often be made with incomplete information about the requesting principal.

Our work intends to map a formal application of the human dynamic trust-formation capability to a platform with which we can then assess trust-based admission control in collaborative ad hoc applications. Essentially, we aim to implement a system that allows trust formation based on criteria that satisfy the question 'Why is this principal trusted to perform this action?' while allowing principals to implement high level trust-based security polices to perform admission control.

The structure of the paper is as follows. Section 2 is an examination trust concepts and policy formation. Section 3 specifies the design of the trust-based admission control system framework. This includes the development of a generic framework that will allow users to form, join, and exit collaborative ad hoc groups as well as the integration of a trust production and admission control element. In Section 4, we outline the implementation route and the technologies used in the system, incorporating a simple distributed blackjack application to implement the trust-based admission control system to assign roles to principals according to their trust-based admission rights. In Section 5, we present the evaluation of the trust framework and discuss the results of the sample application. Finally, Section 6 gives conclusions and ideas for future work.

2. Trust Concepts and Policy Formation

Trust is an important feature in the decision-making process that humans use every day. There are many different definitions and views of what trust is. This leads to difficulties in establishing a common, general definition for trust that holds, regardless of personal dispositions or differing situations. Our goal is to develop a generalised trust definition in such a way that if security system is asked, 'Why do you trust this principal?', it will be able to respond with the parameters and contexts on which it has built its trust value for that principal. Moreover, it will then be possible to formulate admission control policies around these trust values.

The aim of this section is to show that trust can be formalised and produced within a framework. We discuss the concept of trust and examine the properties and operations that can be applied to it. Then we investigate the use of policy formation in trust-based systems. Finally, we summarise the key points from this section that are relevant to the design and implementation of applications and policy specification.

2.1. Trust

Here, we present a number of trust definitions from various academic fields in an effort to mesh different views of trust to give a general understanding of trust as a whole. Next, the properties that apply to trust are discussed. Finally, a framework for modelling trust is explained.

2.1.1 Definitions of Trust

According to Marsh [5], who has contributed significantly to formalising trust as a computational concept, the majority of work on defining trust has come from the three academic areas of sociology, psychology, and philosophy. He identifies four research efforts as major contributions: Morton Deutsch, Niklas Luhmann, Bernard Barber, and Diego Gambetta.

The definition of trust given by Morton Deutsch in 1962 is the most widely accepted. Based on psychology, Deutsch [6] defines trust as follows:

- *If an individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial* (Va^+) *or to an event perceived to be harmful* (Va^-)*;*

- *He perceives that the occurrences of* (Va^{+}) *or* (Va^{-}) *is contingent on the behaviour of another person; and*
- *He perceives that the strength of* (Va^{-}) *to be greater than the strength of* (Va^{+}).
- *If he chooses to take an ambiguous path with such properties, I shall say he makes a trusting choice; if he chooses not to take the path, he makes a distrustful choice.*

The repeated uses of the word 'perceives' in this definition implies that trust is a subjective quality individuals place in one another. This definition also implies that trusting decisions are based on a form of cost-benefit analysis. Different individuals' decisions to trust differ with each individual's perception of the estimated cost (Va^{-}) and benefit (Va^{+}). While Deutsch broke trust down into several different circumstances where such a trusting choice might be made, he concentrated on the fact that trust 'is strongly liked to confidence in, and overall optimism about, desirable events taking place.' [7]

A second definition, with a sociological approach, is given by Niklas Luhmann. [8] Luhmann suggests that the relation of the world as a whole to all of the individual diverse identities within is so complex that individuals need trust as a means to reduce complexity and promote adaptation through increasing the possibility for experience and interaction.

Next, Bernard Barber attempts to solidify a sociological definition of trust. He views trust 'predominantly as a phenomenon of social structural and cultural variables and not...as a function of individual personality variables.' [9] This definition strays from Deutsch's individual, subjective thoughts on trust, and aligns itself more closely to Luhmann's ideas that trust exists at a social level.

Finally, Diego Gambetta [10] amalgamates work from such diverse areas as biology and economics to define trust as 'a particular level of the subjective probability with which an agent will perform a particular action, both before he can monitor such action and in a context in which it affects his own action.' This definition again reinforces the subjective nature of trust. The definition also states that trust is affected by actions that we cannot monitor. This means that in a situation where full information is not available, the quality and meaning of the information one can monitor will have a major effect on his level of trust.

From Deutsch and Gambetta, then, we find definitions for trust as a subjective quality, depending strongly on each individual's views of his environment. From Luhmann and Barber, we add to the subjective nature of trust that trust must exist at a higher level as well, i.e. in the context in which entities are functioning.

2.1.2 Properties of Trust

With the above definitions of trust, we can identify properties of trust. Properties of trust include subjectivity, ability to reduce complexity, non-transitivity, context, and ability to be measured.

The *subjective nature* of trust is the property that creates the single most difficult aspect in creating a trust production element for a computer security model because the parameters used for calculating trust by different individuals varies widely.

On the other hand, we may conjecture that, though this subjective nature of trust leads to increasingly complex scenarios, if we can identify a trust framework to work within, we can decrease this complexity and make use of the property of trust that allows entities to adapt to new contexts and interactions.

It has been shown that trust is *not implicitly transitive*. [11] The following statement gives a good example of this: 'If Ann trusts Bob, and Bob trusts Cathy, then Ann trusts Cathy.' This is in general not true. However, in any large network of diverse entities, it is impossible for one principal to know and trust every other principal. This is the point where a recommendation operation comes into play. Using a recommendation operation, the above statement can be considered true when the following conditions hold:

1) Bob explicitly tells Ann that he trusts Cathy. This is an example of a recommendation.
2) Ann trusts Bob as a recommender. If Ann does not trust Bob as a recommender then she should ignore any information he passes her.
3) Ann has the choice of making a judgement on the quality of Bobs recommendation.

Trust is also *context-specific*. In order to give a recommendation about a principal's trustworthiness, the context in which the recommendation will be used is needed. The following statement is an example of this: 'I trust my brother to drive a car, but not to fly a plane.'

Finally, Dasgupta shows that, while trust has no measurable units, its *value can be measured*. [12] Trust can therefore considered to be a commodity, like information or knowledge. There are problems using explicit values to represent trust. Because trust is subjective, the same value of trust may be associated with different levels of trust by different entities. If we use continuous values to model trust, it reflects the continuous nature of trust and allows for comparison of different trust frameworks. Marsh [5] represents trust as a continuous variable over a range scale [-1,+1]. Marsh states that trust can have threshold values, which vary between people and situations. An entity will have a positive threshold value, above which that trust another entity, and a negative threshold value, below which they will not trust a person. The continuous value system still leaves the problem of subjectivity for receiver of the trust information, if the framework and parameters on which the trust value is calculated are not understood.

2.1.3 Framework for Trust

Although the concept and operations of trust can be well-defined, as above, a framework for quantifying trust is needed so that unambiguous conversation may occur. McKnight and Chervany [13] present a framework that classifies different aspects of trust and provides a system which shows how trust can influence behaviour. This framework is depicted in Figure 1.

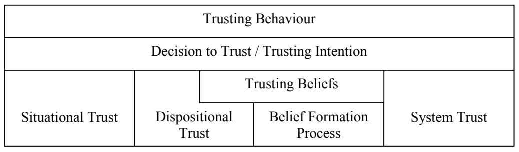


Fig. 1. McKnight and Chervany's Trust Framework

Here we see six integrated components, Situational Trust, Dispositional Trust, Belief Formation Process, System Trust, Trusting Beliefs, and Trusting Intention, feeding different aspects of trust into one actual outcome, Trusting Behaviour.

Situational Trust describes the extent to which one principal intends to depend on another, regardless of who the other is, in a given situation. It means that this principal has formed an intention to trust every time a particular situation arises, irrespective of his beliefs about the attributes of the other party in the situation.

Dispositional Trust is the extent to which a principal consistently trusts across a broad spectrum of situations and parties. This is an example of a cross-situational, cross-personal construct. It is a general intention by a person to believe that people will behave in a certain manner and is valid over a broad spectrum of situations. This value reflects whether an individual is optimistic or pessimistic in their approach to new situations, and is the part of the framework that inputs the subjective nature of trust as defined above.

System Trust represents the extent to which a principal believes that the proper impersonal structures are in place to enable him to act in anticipation of a successful future endeavor, regardless of the other party he must depend on. System trust reflects that safeguards are in place to reduce the amount of risk to which an entity must be exposed. These safeguards may be in the form of regulations, guarantees, or stabilising intermediaries.

The *Belief Formation Process* is the process by which information and experience that have been gathered from the environment are processed to form new trusting beliefs about parties. It is made up of two mechanisms: categorisation mechanisms (unit grouping, reputation categorisation, stereotyping) which organise information into categories, each with an equivalent response; and illusionary mechanisms, which are based on assumptions, emotions, and levels of confidence.

Trusting Beliefs overlaps the Belief Formation Process and may or may not overlap Dispositional Trust. Trusting Beliefs are described as the extent to which one principal believes that another principal is willing and able to act in the trusting partys best interest. The attributes that affect the Trusting Belief include benevolence, honesty, competence and predictability. Assessing these attributes to form a belief is an inherently subjective process.

A *Trusting Intention* is formed when one principal is willing to depend on another party in a given situation with a feeling of relative security, even though negative consequences are possible. A Trusting Intention implies that the entity has made a decision based on the various risks and benefits of trusting.

Trusting Behaviour is the outcome of the above components, when one principal depends on another party in a situation with a feeling of relative security, even though negative consequences are possible. This construct effectively describes the act of trusting and implies the acceptance of risk. [14]

Using this trust framework helps us clear conceptual confusion by representing trust as a broad but coherent set of constructs which incorporate the major definitions from research to date. The definition of these trust types is very beneficial, as it enables parties to apply and exchange trust information and to agree on what they mean by a trust value. Using the framework to identify the parameters on which trust is based allows principals to be specific about their trust statements. Overall, the framework enables the use of trust-based tools such that principals might avoid the problems associated with trust and collaboration in an ad hoc environment.

2.2. Policy Specification

Generally, access control policies specify the conditions under which a principal may access a particular role or resource. Using trust concepts as the criteria for access control is a relatively new development in policy specification, and it is an important part of any formal framework that defines trust relationships between entities. In this section, we give an overview of some of the recent research carried out in the area of policy formation for trust-based systems. The requirements for policy languages in trust-based systems are discussed and three policy specification languages are considered.

2.2.1 Language Requirements

A growing body of research in the area of trust management [15], [16], [17], [18], [19] has provided us with several requirements of a policy language to be used in environments where trust must be established between strangers. A summary of these requirements taken from Damianou et al [18] is presented as follows:

- A policy language should support security policies for access control and delegation.
- Structuring techniques to facilitate the specification of policies relating to large systems with millions of objects. This implies the need for policies relating to collections of objects rather than individual ones.
- Composite policies are necessary to allow the grouping of basic security and management policies relating to roles, organisational units, and specific applications. Composite policies become essential in large enterprise information systems which have high levels of complexity in policy administration.
- The language must allow for policy analysis, e.g. identifying conflicts and inconsistencies in the specification or determining which policies apply to an object and vice versa. It is believed that declarative languages make such analysis easier.
- The language must allow extensibility to cater for new policy types. This can be supported by inheritance in an object-oriented language.
- Finally, the language must be easy to understand and use.

2.2.2 Languages

Having determined the requirements of a policy language for trust-based access control, we now give an overview of three existing policy languages that are used in environments where trust establishment between strangers is required: Ponder, RT, and TPL.

Ponder [18] is a declarative, object-oriented language used to provide a standard way of specifying a wide range of policies for management and security in distributed systems. Ponder allows for flexible policies that can map onto various access control implementations. It includes authorisation, delegation, information filtering, and refrain policies that specify access control. There are also event-driven obligation policies to specify system management actions. Because Ponder allows for inheritance, it permits new policy classes to be defined as sub-classes of existing policy classes. Additionally, Ponder uses a subset of the Object Constraint Language (OCL) to specify the set of conditions or constraints under which a policy is valid. This model also allows for translating policies to structured representation languages such as XML, which can then be used for viewing or exchanging policy information. The language supports policy specification for large-scale systems with millions of objects.

RT [20] is a family of role-based trust management languages for specifying policies and credentials for authorisation in distributed systems. Like Ponder, RT makes use of the principles of role-based access control. However, it also combines RBAC principles with those of trust management systems. Using credentials, RT allows localised authority over roles, delegation, linked roles, and parameterised roles. This is the language Winsborough and Li use to specify policies in their framework of practical automated trust negotiation. While they express RT in abstract syntax, in practice it can be represented in various forms, e.g. XML. [21]

The Trust Policy Language (TPL) developed by Herzberg et al [19] is an XML-based language used to define the mapping of strangers to predefined business roles, based on certificates issued by third parties. It extends existing role-based access control mechanisms in an effort to cater for the trust establishment in environments where entities do not necessarily know each other. It uses certificates to convey 'any useful reference about the subject, not necessarily its identity.' Mass and Shehory [22] use TPL as the basis for policy specification their trust establishment framework. This framework provides role-based access control without requiring identities by generating trust amongst distributed agents. Based on Public Key Infrastructure (PKI) technology [3], [23], trust is established based on assessing agents' certificates and determining whether the level of trust in those certificates meets the TPL-specified policy criteria.

2.3. Design-Relevant Trust and Policy Concepts

The aim of this review of trust and policy concepts was to highlight areas that would be relevant to the design and development of a trust-based admission control system and an associated policy specification.

We find that a trust framework exists which captures the major definitions and properties of trust. When we design our trust-based admission control system, then, we do so based on this comprehensive trust framework, such that its implementation will show that unknown entities can interact and exhibit trust production. To enable dynamic trust-based admission control within the system, we must also specify high level policies, keeping in mind the requirements and examples of trust-based policy specification.

3. Design

This section gives a general overview of the collaborative ad hoc application environment, identifying the major processes involved and the requirements of each process.

Additionally, the design of the components that provide the required functionality is presented.

3.1. Collaborative Ad Hoc Application Processes and Requirements

We assume that the users of our collaborative application interact in groups in an environment where an ad hoc networking system is in operation. Each user has a small computer device that is capable of joining the ad hoc network. It is assumed that each device has a screen, a hard disk for storing permanent information, and networking capability. Five major processes are necessary to allow the users to interact: service discovery and connection, admission request, global trust-based admission control, local trust-based admission control, and ad hoc group formation.

This section examines these processes and identifies the requirements of each. Figure 2 depicts a high level view of this environment.

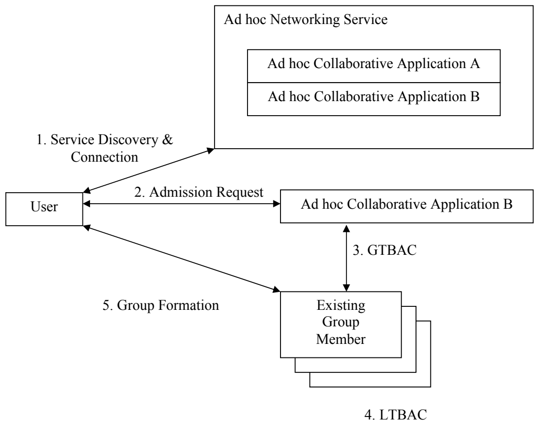

Ad hoc Networking Service
Ad hoc Collaborative Application A
Ad hoc Collaborative Application B
1. Service Discovery & Connection
2. Admission Request
User
Ad hoc Collaborative Application B
3. GTBAC
5. Group Formation
Existing Group Member
4. LTBAC

Fig. 2. High Level View of Main Design Processes

3.1.1 Service Discovery and Connection Process

From a list of collaborative applications available in a given ad hoc network, a user - let's call him new principal, P_n - selects the specific application he wishes to join and downloads a proxy that allows his device to communicate with the application and to initiate the join process. The technical issues associated with application discovery and connection may be handled by any number of service lookup and discovery technologies, such as the JINI lookup service, the CORBA trader service, or JavaSpaces.

3.1.2 Admission Request Process

Having discovered what roles are available for users within a collaborative application, P_n requests admission to group, G, in a particular role. P_n will typically be interested in joining the role with the most access rights. To facilitate this, P_n is allowed to make multiple requests to join G. When the application receives an admission request from a user, it initiates the voting process.

3.1.3 Global Trust-Based Admission Control (GTBAC) Process

The application initiates GTBAC process when P_n requests admission to group, G. The application manager will ask each principal, P_g, that is already an existing member in G to verify that P_n has the right to join G in the requested role. Each P_g votes for or against admission based on the results of the local trust-based admission control process, described in the following section. The application then accepts or rejects P_n's request for admission to G in a particular role based on the results of the GTBAC process.

At the end of the GTBAC process, P_n is informed of the decision, and so is the each P_g. As a back-out mechanism, any P_g can opt to leave the group if he does not agree with the result of the GTBAC process.

If P_n is successful in his request to join G, he will then register officially with the application and be included in future GTBAC processes.

3.1.4 Local Trust-Based Admission Control (LTBAC) Process

LTBAC incorporates trust-based policies to enable decision-making regarding admission requests in the GTBAC process. LTBAC is executed at each P_g each time the collaborative ad hoc application looks for verification of P_n's request for admission to the group.

All P_g are passed P_n's identifier and requested role. This identifier and role are passed into each P_g's trust engine which uses local trust-based policies to determine whether or not P_n is trusted enough to join G. If P_n meets or does not meet P_g's local policy criteria, P_g provides an output to the GTBAC process accordingly.

The development of LTBAC policies for the successful operation of this process is the main focus of this paper.

3.1.5 Group Formation Process

If P_n is successful in his admission request to join G, an application proxy is provided to allow access to all the functionality needed for the application.

The application will decide how the group members will communicate. All communication between the group may be multicast or members may unicast to each other.

3.2. System Components

Having introduced the requirements of the five major processes necessary for the collaborative ad hoc application, we now describe the design of the three major components of the application. This illustrates how the requirements identified above have been integrated into the design of the three system components: application management, admission protocol, and trust-based admission control.

3.2.1 Application Management

The application is designed to have two management components, the *Admission Manager* and the *Group Manager*. The Admission Manager manages the Admission Request, LTBAC, and GTBAC processes. The functionality of the Admission Manager does not change across different application implementations and the design can therefore be reused. The Group Manager, on the other hand, provides context-specific services in managing the Group Formation Process, and therefore must be designed for each application.

3.2.2 Admission Protocol

The Admission Request and GTBAC processes both involve interaction between P_n's device and the application. Both processes are therefore combined in the admission protocol design.

The admission protocol allows users to form into collaborative groups. The protocol is designed so that every P_g gets to verify any new P_n, and each P_g can back out of G at any stage. This ensures that each P_g is able to decline collaboration with a GTBAC-accepted P_n by opting out of the group. The protocol allows a decentralised group decision to be made upon receiving P_n's admission request. The identities of all P_g are kept hidden from the P_n.

Once P_n has selected an application and downloaded a service proxy, the admission protocol will be initiated. The protocol involves six steps which are explained below.

- Get Roles: P_n gets a list of possible roles that the application supports.
- Admission Request: P_n requests admission to G in the highest available role, passing on his identifier and the requested role to the Admission Manager.
- Verify User: When P_n's admission request is accepted by each P_g, P_n verifies the identifiers and current roles of all P_g to confirm that they meet the criteria set in his own admission policies. If P_n's admission request is rejected, he can attempt to join in lower role.
- Officially Join: When P_n has verified the existing group membership, he will then officially join the group.
- Inform All: All P_g are informed that a new user has been granted admission. Any P_g can take this opportunity to exit the group if he disagrees with the GTBAC result.

3.2.3 Local Trust-Based Admission Control (LTBAC)

One of the design goals for the LTBAC component used in the collaborative ad hoc application is to minimise the amount of configuration that a user must perform. The LTBAC component requires a number of support components to gather information, manage policies, and produce trust values. Figure 3 presents an overview of the LTBAC component, illustrating which subcomponents are generic and which are application-specific.

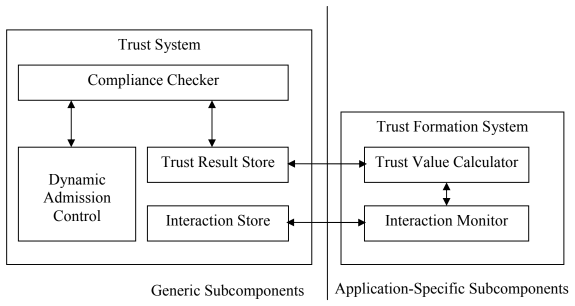


Fig. 3. Local Trust-Based Admission Control (LTBAC)

When P_n is requesting admission to a group, each P_g's *Compliance Checker* passes in P_n's name and requested role. The Compliance Checker then relays the results of the GTBAC process to the Admission Manager, which relays the result to P_n. A compliance check incorporates three main steps:

- Gather admission control policy statements that are relevant to P_n.
- Identify P_n's trust value from the Trust Result Store.
- Compare P_n's trust value and role request to local admission policies.

If P_n's trust value and role request meet P_g's policy criteria, the Compliance Checker informs P_g that P_n's admission request may be accepted.

The *Dynamic Admission Control* subcomponent works with the Compliance Checker by binding a minimum acceptable trust value to the right to access a particular role. It achieves the flexibility and manageability of role-based access control, as earlier discussed, by binding a trust value to right of admission to a role. In this manner, one statement can apply to every potential P_n, thus greatly reducing the number of policies a principal must initially create.

A trust-based admission control statement must contain five pieces of information:

- Context: the name of the application to which the policy is relevant.
- Identifier: name of the principal(s) to which the policy applies.

- Condition: threshold that must be met by P_n's trust value in order for the policy criteria to be met.
- Trust Value: value assigned to how much P_g trusts P_n which is calculated based on interaction information.
- Role: name of the role P_n will be approved for if the trust value meets the applied condition.

The *Trust Formation System* component gathers information on interactions between users and produces trust values based on those application-specific interactions. This component is supported by information processed by the Interaction Monitor and the Trust Value Calculator.

The *Interaction Monitor* gathers and saves information about principals' interactions. The Interaction Monitor saves and updates interaction information on the user disk. This information is application-specific, based on principals' interactions in a specific context.

The *Trust Value Calculator* processes the saved interaction information using linear equations to produce principal-relevant trust values. The trust calculation algorithm is customisable to a certain extent to incorporate the subjective nature of trust. The calculator produces a trust value, expressed as a probability, reflecting that the higher the trust value, the more likely it is that P_n will behave correctly. The trust value becomes asymptotic as it approaches one or zero, reflecting that a principal can never be completely trusted or distrusted.

When the Trust Value Calculator is executed, context-specific trust values are stored in a Trust Results file on the user device. Trust results are broken down into:

- Context: the name of the application to which the result applies.
- Identifier: name of the principal to which the trust value relates, e.g. P_{538}.
- Trust Value.
- Date: on which the trust value was calculated.

The *Trust Result Store* and the *Interaction Store* ensure that the interaction information and trust results are passed in the correct format between the Trust System and the Trust Formation System. When called, the Interaction Store passes the stored interaction information to the Interaction Monitor in the Trust Formation System. The Interaction Monitor updates this information with current interaction details and passes it back to the Interaction Store to be saved. Similarly, the Trust Result Store passes saved trust results to the Trust Value Calculator in the Trust Formation System. The Trust Value Calculator updates this trust result with a new result based on current interaction information and passes the new trust result back to the Trust Result Store for storage and access by the Trust System.

4. Implementation

Having described the major processes and system components of the collaborative ad hoc application environment and framework, we now delineate the implementation of the framework and a sample application. We first introduce the technologies used in the implementation. Then, we present a high level overview of the interfaces that make up the framework. Finally, we discuss the implementation of the trust-based admission control system (TBAC) and the sample application, the blackjack card game.

4.1. Technologies

The Java programming language is used for the implementation of all components in this system. The implementation used two specialised technologies for certain components, JINI and XML.

4.1.1 JINI

Out of the available service lookup and discovery technologies, we chose JINI [24] because of its known support for collaborative ad hoc applications.[25] JINI is a Java-based distributed system designed for simplicity, flexibility, and federation. The JINI architecture provides a mechanism for devices and programs to federate into groups, thus allowing entities to offer and share resources within a group.

The key concept within the JINI architecture is that of a service. A service can be downloaded and used by any entity in the federated group. Systems provide services to the federation by publishing them on a JINI lookup service, using the Discovery and Join protocols. A user of a JINI system contacts the lookup service and downloads the service he requires. All communication within the JINI architecture employs Java Remote Method Invocation (RMI), allowing data and code to be transferred around the network between objects. The JINI architecture assumes that a network for connecting devices and services is available, and that each device has a Java Virtual Machine installed.

4.1.2 XML

In the design of the trust-based admission control system, we identified the the following requirement: storing on the user's device of admission policies, trust results, and application interaction details. We found several advantages to using the Extensible Mark-up Language (XML) for this purpose. The information being stored on file is already structured a certain way, so by using XML we are able to maintain that structure. The saved files are easier to understand by users because the structure of the information is identified in the file. Additionally, XML files can easily be parsed and converted into Java objects.

Finally, information stored in this manner can be easily extracted for user profiling purposes.

4.2. Framework Interface

During the design phase, we identified the components that must interact according to a standard procedure, regardless of which collaborative ad hoc application is being used. In order to create a standard framework, the interaction between the various components is defined and implemented through a number of interfaces.

4.2.1 Admission Protocol Interfaces

The Admission Protocol involves the Admission Manager and P_n's client software. Each implements a standard interface to allow the other to invoke remote method calls. Using the Ad hoc Service Interface and the Ad hoc Client Interface, the application publishes a service proxy on the lookup service. P_n's software will be pre-programmed to look for that type of service proxy when attempting to discover collaborative applications on the network.

4.2.2 Service Manager Interface

The Service Manager Interface enables the Group Manager to take over control of the application once the Admission Manager has completed the Admission Protocol. It also provides a standard mechanism for P_n to get access to an object that implements the application proxy.

4.2.3 Application Proxy Interfaces

When the application proxy is initialised, the Trust Value Production System connects to the Trust System. The Interaction and Trust Results are accessed and updated. In the sample application described below, the application proxy also implements the graphical user interface.

At this stage, P_n has finished the Admission Request Process and has been admitted to the group. Now considered part of group, G, this principal, P_g, is returned a reference to the application Group Manager. P_g will use this reference to request access to a role-based GUI from the Group Manager. The Service Manager Interface and GUI Interface allow any client to request and be given access to a role-specific GUI.

The GUI provides the Trust Value Production system to P_g. There are two possible configurations that allow P_g to graphically interface with the application. In the first configuration, GUI code would be stored on P_g's device and initialised when P_g is admitted to the application. In the second scenario, GUI code would be stored with the application and downloaded by P_g whenever the application is used.

We have implemented the second design option in our system, as it allows the application developer to make upgrades to the GUI and still ensure that all users have the most current version. However, the main issue with this option is that the code must be downloaded every time a user gains admittance to the application. We foresee that as mobile networking technology advances, download delays will significantly decrease.

Accessing the GUI entails the following steps:

- P_n completes the admission request and is given a reference to the Group manager. Having successfully joined group G, P_n is now a member of G, thus, P_g.
- P_g calls on the Group Manager to provide a GUI object and downloads the GUI Factory.
- P_g makes a call to the GUI Factory, passing his role, a reference to the local trust system, and a reference to the remote Group Manager.
- The GUI Factory produces a role-specific GUI for P_g.
- The role-specific GUI contacts the Group Manager.

4.2.4 Trust and Policy Management Interfaces

The Trust and Policy Management Interfaces are implemented by P_g's trust system. They allow the application-based Trust Value Production System to access and store the relevant trust and policy files on P_g's machine. As a session proceeds, the Interaction Monitor updates these files. At the end of a session, the Interaction Monitor writes the details to P_g's files after calling the Trust Value Calculator to update Trust Management Result Store.

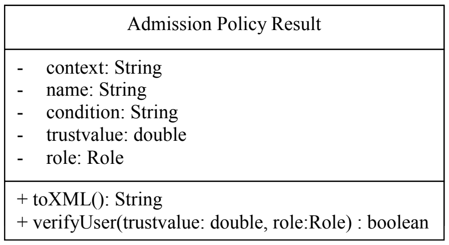
4.3. Local Trust-Based Admission Control (LT-BAC) Implementation

This section explores the implementation of the LT-BAC system, concentrating on admission policy, trust management, and the client-side trust system. The design of this system requires that the Admission Policy Result and Trust Management Result statements should be uniform for all principals and applications.

4.3.1 Admission Policy Result Object

In the Dynamic Admission Control segment of the design phase, we identified five key pieces of information to be stored in the admission control statement: context, identifier, condition, trust value, and role. This information and its associated functionality is implemented in the Admission Policy Result object, as seen in Figure 4.

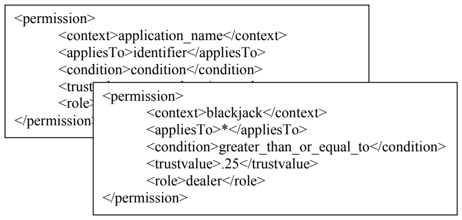
Initially, the implementation of this object in the system allowed P_g to choose between two approaches. In the first approach, P_g could put in a specific identifier, thus binding an admission policy directly to a particular identity, similar to the mechanism used in traditional access control lists. Alternatively, a second approach is to state that the policy applies to all principals, as noted by inputting an asterisk in the identifier field. For the purposes of producing useful correlations between admission policies and trust values, we concentrated on the second, more generic approach. Thus, in the current implementation, P_g may choose only between three specified generic policy pairs based on the nature of trust: High Trust, Medium Trust, Low Trust.



```
Admission Policy Result
- context: String
- name: String
- condition: String
- trustvalue: double
- role: Role
+ toXML(): String
+ verifyUser(trustvalue: double, role:Role) : boolean
```

Fig. 4. Admission Policy Result Object

The Admission Policy Result object allows P_g to get and set the data held in the five data fields based on criteria associated with a high, medium, or low level of trust. The object also implements a method that determines if P_n's trust value and requested role meet the conditions specified in the admission policy. Moreover, the object implements an XML conversion function to support the storage of the admission policy information on P_g's disk. Figure 5 presents the generic XML storage format overlaid by a specific case.



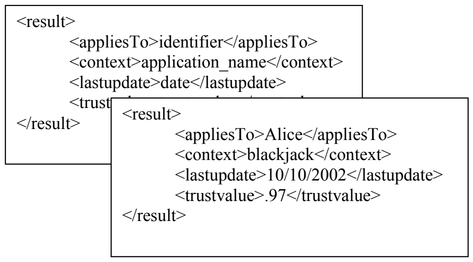
```
<permission>
        <context>application_name</context>
        <appliesTo>identifier</appliesTo>
        <condition>condition</condition>
        <trust
        <role
</permission
```

```
<permission>
        <context>blackjack</context>
        <appliesTo>*</appliesTo>
        <condition>greater_than_or_equal_to</condition>
        <trustvalue>.25</trustvalue>
        <role>dealer</role>
</permission>
```

Fig. 5. Admission Policy Result stored as XML

4.3.2 Trust Management Result Object

Similarly, the Trust Management Result object is used to store trust and interaction information about other principals. The Trust Management Result object has four fields that can be accessed and modified. The object will also produce an XML string for the purposes of storage. We see the generic XML storage format overlaid by a specific case in Figure 6.



```
<result>
        <appliesTo>identifier</appliesTo>
        <context>application_name</context>
        <lastupdate>date</lastupdate>
        <trust
</result>
```

```
<result>
        <appliesTo>Alice</appliesTo>
        <context>blackjack</context>
        <lastupdate>10/10/2002</lastupdate>
        <trustvalue>.97</trustvalue>
</result>
```

Fig. 6. Trust Management Result stored as XML

4.3.3 Client-side Trust System Implementation

The client-side trust system design proposed above is implemented in the client-side software. An overview of this system is shown in Figure 7. When P_g's software is initialised, the Trust Management Result and Admission Policy Result are parsed and their respective objects are stored in the Trust Management Result Store and Admission Policy Store. Should P_g need to decide on the validity of P_n's admission request, his software would call the Compliance Checker. The Compliance Checker calls the two stores to gather the relevant policy pairs and trust values. A trust value for the P_n requesting admission to the application is, along with the P_n's requested role, compared to P_g's admission policy pair to see if the admission request should be confirmed or rejected. It is important to note here that a principal can never be fully trusted or untrusted. Therefore the trust value upper and lower bounds are .99 and .01 respectively.

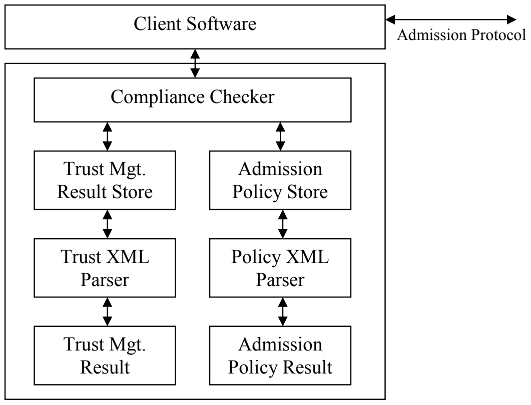


Fig. 7. Trust System Implementation

4.4. Sample Application

We implemented the blackjack card game as a sample application to demonstrate trust-based admission control in ad hoc collaborative environments. Blackjack is a popular card game in which players gamble with a dealer over the value of a set of cards. This application was chosen due to the ingrained use of trust in the way players join the game, thus allowing for a comparison between the human and computer trust models.

In blackjack, there are two distinct roles: dealer and player. The role a principal has in the game has a major impact on the willingness of others to join the game. Because in blackjack, the dealer's odds of winning are more favorable than the odds of the player, the principal holding the dealer role must be considered trustworthy by other players. Looking at this from another angle, the right to assume the advantageous dealer role can be seen as a privilege earned through fair, trustworthy playing of the game.

The trustworthiness of any P_n is calculated based on three parameters: the number of sessions played, the number of days since the last session of interaction, and the amount of cumulative debt owing. The perceived benefit of maintaining a satisfactory trust value (one that is high enough to gain admission to a game when compared to other users' admission policies) is the ability to continue gaining admission to desirable ad hoc blackjack ”tables”.

4.4.1 Blackjack Group Formation

When P_n has successfully been admitted to a group in a requested role, the group formation process begins. In this implementation, all players are required to connect directly to the user who has the dealer role.

When the Blackjack Dealer GUI is initialised, it will register with the Group Manager, which implements the Blackjack Manager Interface. The Blackjack Manager Interface allows the dealer to register his proxy in central well-known location. Players can then ask the Group Manager to pass them a reference to the dealer. When the Blackjack Player GUI is initialised, it connects to the Group Manager, gets a reference to the dealer of the group, and registers himself to start a game.

Two further interfaces enable the smooth running of the game. The Blackjack Dealer Interface implements two methods allowing players to join and leave a blackjack game and remaining methods for game execution. The Blackjack Player Interface allows a player to receive updates and commands about the game from the dealer.

4.4.2 Blackjack Local Trust-Based Admission Control (LTBAC)

The LTBAC system for the collaborative ad hoc blackjack application implements some specific functionality for the purposes of evaluating admission control and trust management.

In this implementation of the blackjack application, the Admission Policy Result object allows P_g to generate one of three generalised policy pairs: High Trust, Medium Trust, Low Trust. These three policy pairs are based on criteria associated with a high, medium, or low level of trust, depending on whether P_g is of a disposition to be highly trusting, moderate, or highly distrusting in the context of blackjack. An example of a High Trust Policy pair follows:

- High Trust Policy Pair
  - Context: blackjack
    Applies To: $*$
    Condition: $\geq$
    Trust Value: .25
    Role: dealer
  - Context: blackjack
    Applies To: $*$
    Condition: $\geq$
    Trust Value: .10
    Role: player

From this, we see that the only variants in admission control are associated with a trust value and role request.

The Trust Management Result object is used to store trust and interaction information about other principals, as explained in the LTBAC implementation section above.

After each session, the Interaction Monitor updates the values of the monitored parameters associated with a principal. In blackjack, these parameters are: Session Number[1], Days Since Last Session, and Cumulative Debt. The Trust Calculator uses linear equations within the trust engine code to calculate these partial results so as to produce an overall trust value. As with the overall trust value, each partial result has an upper bound of .99 and a lower bound of .01. The value of each of the partial trust values is measured against a limit set by the player. The Session Number is compared to the Session Limit variable. If the Session Limit is 10, it will take 10 times of playing for a partial trust level to reach its maximum of .99. Therefore the session-related trust value increases as the Session Number value increases. The other two partial trust levels are calculated with inverse proportions, i.e. as Cumulative Debt or Days Since Last Session increase with respect to set limits, their respective partial trust results decrease.

The same set of partial trust parameters is monitored for each principal, although, as we have just seen, scaling is possible within the context of the application by setting the parameter limits at different levels. One implementation of this blackjack application allows P_g to customise his trust calculator by defining his own limits for each

[1]Session Number: number of interactions one principal has had with another.

parameter, thus demonstrating the subjective nature of trust, as P_{g1} might put more emphasis on debt while P_{g2} considers session number to be more important to trust. A second blackjack implementation pegged parameter values to the admission control trust dispositions (high, medium, or low). For the purposes of this discussion, however, we have set the scaling factors to the same limits for all principals, such that data resulting from testing is uniform for our evaluations.

When a trust value is required for P_n, the trust level of each stored interaction parameter is calculated. The three partial trust levels are then fed into the trust engine to output a final trust value. This value is then stored as a Trust Management Result.

5. Evaluation

Our main goal is to assess the TBAC system to find out if it produces usable, relevant trust values in a way that can be used for role-based admission control in a collaborative ad hoc application environment. Our assessment of this system is performed through a controlled sequence of blackjack outcomes, whereby we can evaluate both admission policy strategies as well as trust production parameters. First, we delineate the test parameters. Then we provide the test results and analysis.

5.1. Test Parameters

To keep the gaming environment controlled, we kept it as simple as possible. In each group, or gaming table, there are two roles, one dealer and one player. Each principal sets his trust level to high, medium, or low, depending on his trusting disposition in the context of blackjack. This affects his admission control policy. If the trust level is set to high (H), the principal uses a policy pair whereby admission to the role of dealer requires a trust value, T, of .25 and admission to the role of player requires T = .10. Alternatively, if the trust level is set to medium (M), the principal uses a policy pair whereby admission to the role of dealer requires T = .50 and admission to the role of player requires T = .25. Finally, if the trust level is set to low (L), the principal uses a policy pair whereby admission to the role of dealer requires T = .75 and admission to the role of player requires T = .50. Admission policies are not enforced the first time a principal requests admission to the group, i.e. before interaction behaviour can be evaluated T_{P_n} is assumed to be .50, which is high enough to enter as a player in any game.

Additionally, each principal employs a debt repayment strategy, based on concepts from Axelrod's *Evolution of Cooperation*. [26] A principal will Always Pay (AP), Never Pay (NP), or Random Pay (RP). The randomness of payment/non-payment for the RP strategy was generated with a random numbers generator. [random.org]

Combining admission control policy with debt repayment strategy gives us a set of principals whereby each principal $\in$ {HAP, HNP, HRP, MAP, MNP, MRP, LAP, LNP, LRP}. Each principal in the set is tested in both the role of dealer and the role of player.

The principal in the role of player loses \$1 per game for 20 games, so that we can assess that principal's trustworthiness after each game, based on trust value and debt repayment behaviour. The trust level parameter limits for all principals in all games are static: Session Limit = 10, Day Limit = 1, and Debt Limit = 10. These values were chosen at the more restrictive end of the scale so as to promote small steps in trust increments/decrements.

5.2. Results and Analysis

Every principal in the role of dealer, P_d, was matched against every other principal in the role of player, P_p, for 20 blackjack games, with P_p losing each match. As each game was played, we recorded P_d's Trust Management Result for P_p. Because P_d wins each game, his debt repayment strategy is not relevant. Therefore, the effects on trust are generalised by grouping together all possible P_d per trust category, high, medium, or low: Hxx = {HAP, HNP, HRP}; Mxx = {MAP, MNP, MRP}; Lxx = {LAP, LNP, LRP}.

Similarly, the admission control policy of P_p is not important in our trials since the dealer, P_d, is the only principal performing admission control. Therefore, we generalised all P_p per debt repayment strategy: xAP = {HAP, MAP, LAP}; xNP = {HNP, MNP, LNP}; xRP = {HRP, MRP, LRP}.

By performing all rounds of games on the same day, we were able to remove the effect of the Days Since Last Played parameter, so as to isolate the effect of Cumulative Debt and Session Number on overall trust values.

The value of Session Number in each round moves in one-step increments from 1 - 20. Because the Session Limit is set at 10, we note that the Session Number partial trust result visibly affects overall trust in games 1 - 10 as Session Number is advancing towards Session Limit, after which the Session Number partial trust result levels off at 0.99. Therefore, we can separate out two types of results: (1) the effect of Cumulative Debt on T_{P_p} in games 1 - 10 when Cumulative Debt and Session Number both affect overall trust; as distinct from (2) the effect of Cumulative Debt in games 11 - 20 when Cumulative Debt may or may not have reached the Debt Limit and Session Number is no longer affecting a rise or decline in T_{P_p}. Once this is done, we describe how trust-based admission control policies are affected.

5.2.1 Effects of Debt and Session Number on Trust

Figure 8 depicts how trust is generally affected by Session Number and Cumulative Debt, without specific reference to admission control policies. We break our analysis down into two segments: effects on T_{P_p} in games 1 - 10, and effects on T_{P_p} in games 11 - 20. In the

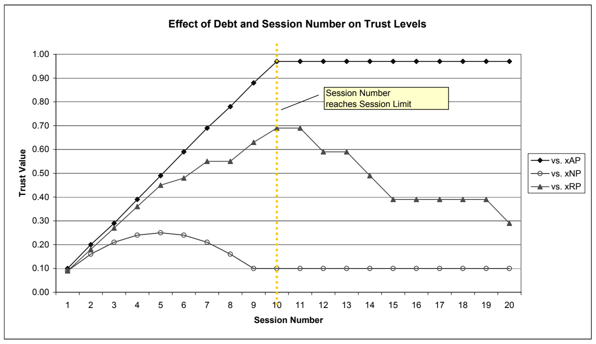


Fig. 8. Effects of Debt and Session Number on Trust Levels

first 10 games, the Always Pay debt repayment strategy (xAP) results in $T_{P_{p(xAP)}}$ increasing in sharp even steps due to the fact that the Session Number is increasing consistently towards the Session Limit and no debt is being accumulated. After the Session Limit is reached in game 10, $T_{P_{p(xAP)}}$ plateaus at 0.97.[2] From this point on, a principal who has reached the Session Limit and continues to pay all debt in full will maintain this high level of trust.[3]

The line plotting trust value for a principal using the Never Pay debt repayment strategy (xNP) curves gradually up and then down during the the first 10 games. The increase in $T_{P_{p(xNP)}}$ here is due to the fact that Cumulative Debt is relatively small in the first 5 games, while the Session Number is steadily increasing towards the Session Limit. By game 6, however, Cumulative Debt is so big (and still increasing) that $T_{P_{p(xNP)}}$ starts to gradually fall. Once the Session Number reaches the Session Limit, this parameter can no longer mitigate the damage to trust being done by an ever-increasing Cumulative Debt. In fact, by game 10, Cumulative Debt has reached the Debt Limit, and levels off at a partial trust result of 0.10. This is reflected by $T_{P_{p(xNP)}}$ in games 10 - 20, which plateaus at 0.10, a very low level of trust.

Finally, $T_{P_{p(xRP)}}$, where a random debt repayment strategy is used, increases in steps throughout games 1 - 10, as Cumulative Debt never grows high enough to counterbalance the increase in the session-related partial trust value. When debt is unpaid, trust does not decrease, but the line does not advance as steeply as when debt is paid. After game 10, though, a descent in trust commences because there is no longer an increase in the session-related partial trust value to keep debt-related trust from pulling down the overall trust value. When dept is unpaid, trust drops. Even when debt is paid, the overall trust value plateaus, as shown on the graph at games 10 - 11, 12 - 13, and 15 - 19. Eventually, when Cumulative Debt reaches the Debt Limit, the line plotting trust value would fall to the lowest possible level, 0.10.

These results show increases and decreases in trust in a collaborative ad hoc application environment that follow closely the way human trust rises and falls based on number of interactions and debt repayment. When interactions increase, trust grows, unless the interactions become increasingly tainted by incorrect behaviour, i.e. not paying one's debts.

5.2.2 Effects of Debt and Session Number on Trust-Based Admission Control Policies

Figure 9 again depicts the rise and fall of trust due to Debt and Session Number. This chart also highlights the trust-based admission control policy pair values, i.e. the minimum amount a principal must be trusted in order to enter the role of dealer or player when high, medium, and low trust-based admission control policies are enforced.

As mentioned above, admission policies are not enforced the first time a principal requests admission to the group because T_{P_p} is assumed to be .50, which is high enough to enter as a player in any game and as a dealer in groups where P_d employs high or medium trust-based admission control policies. After completing one interaction, however, P_p's trust value is calculated as a product of the three partial trust results. With the current, restrictive Session Limit, Date Limit, and Debt Limit settings, T_{P_p} remains low throughout the first three interactions, regardless of debt repayment strategy. Any P_p looking to join a group after having had only one interaction with P_d would only ever be admitted in the role of player by P_d employing a high trust admission policy.

Once interactions begin to accrue, though, we find that the LTBAC policies are appropriate. For $P_{p(xAP)}$ who consistently pays all debt over time, $T_{P_{p(xAP)}}$ remains at a level where P_p can always enter in the highest role to collaborate with an entity employing the most restrictive of LTBAC policies. A $P_{p(xRP)}$ that sometimes pays debt gets multiple chances to increase $T_{P_{p(xRP)}}$. While this $P_{p(xRP)}$ is barred from accessing the highest role in a game against a principal invoking the most restrictive trust

[2]All trust values are rounded to 2 decimal points.

[3]As earlier mentioned, a trust value is also dependent on a third parameter, Days Since Last Session, which is not measured in these trials. Should the trials be performed over a number of days, however, the Days Since Last Session parameter would affect trust behaviour: as Days Since Last Session increases with respect to set limits, the associated partial trust results decrease, thus affecting overall the overall trust value.

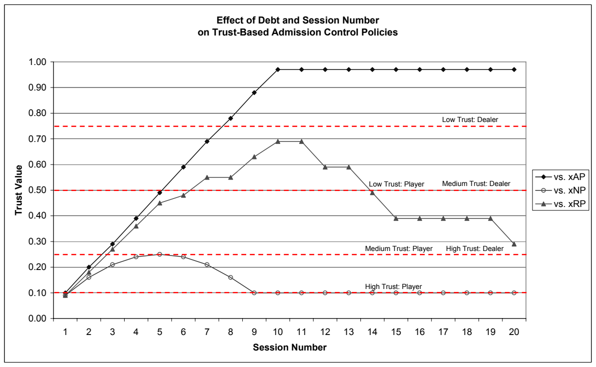


Fig. 9. Effects of Debt and Session Number on LTBAC Policies

policy, he is nevertheless able to enter into a lower role in the most restrictive environment. Still following a parallel with the human trust model, once debt accrues to a point where $T_{P_{p(xRP)}}$ slowly decreases, $P_{p(xRP)}$ is allowed only into games where P_d uses medium and high trust policies. Unless Cumulative Debt is paid off, $P_{p(xRP)}$ is relegated to requesting admission to the lowest role against only high trust principals. Finally, principal $P_{p(xNP)}$ who never pays debt can only ever enter games in lowest role against principals who have high trust policies. This illustrates that our application not only produces trust results in a manner very similar to human trust calculation, but also that the admission control policy pairs permit access to resources using a range of human-like restrictions.

5.2.3 Implementation of Trust Framework and LTBAC Policies

Based on the above experiments, we see that we have only partially implemented McKnight and Chervany's trust framework:

- The *Situational Trust* aspect is implemented by including the concept of contexts for trust production.
- Similarly, the subjectivity of *Dispositional Trust* is present by allowing a principal to shift his admission criteria between different trust-based policy pairs as well as to set the limit parameters according to which parameter matters most to him.
- The act of *Belief Formation* is only partially replicated in our implementation, in as much as an entity observes trust production and forms 'beliefs', or applies policy, accordingly. In order to extend this, we feel that incorporating a recommendation or reputation component is necessary, and we will explore this area in future work.
- *System Trust* is represented in our implementation through the belief of a given user that the system is working to ensure that only trusted entities are admitted to the collaborative application, while entities that behave incorrectly are punished by denial of access.
- Trusting Beliefs are made generic by specifying only three policy levels. However, in our implementation, *Trusting Beliefs* include the aspect of context rather than remaining apart from it. In order to simulate a human trust framework more closely, it may be necessary to extend the policy specification and separate out the element of context.
- The *Trusting Intention* is represented by the process of comparing Trust Management Results with LTBAC Policies.
- Finally, *Trusting Behaviour* is exhibited when P_g admits P_n to the group based on the result of the GTBAC process. However, we mentioned above that Trusting Behaviour also implies the acceptance of risk. In this implementation of the framework, we do not analyse the element of risk, which should feed into Trusting Behaviour. With future work we hope to incorporate a risk-analysis component such that overall risk may be assumed or rejected, which will bear on the final decision to perform Trusting Behaviour or not.

Taking all of this on board, the framework we have implemented as compared to McKnight's is depicted in Figure 10, with the areas requiring further work shaded in grey.

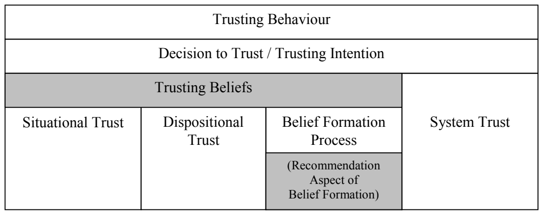


Fig. 10. Implementation of Trust Framework

Policy specification within the TBAC framework is promising, considering that even a simple XML-based approach showed good results. More complicated LTBAC policies, particularly those that allow for more complex interaction, such as passing recommendation certificates, will most certainly prove too difficult to specify directly in XML. Therefore, we foresee the requirement of a more complex policy specification in future implementations.

5.3. Summary

Testing and analysis of the trust-based admission control system prove that it reacts correctly to changes in a principal's behaviour, i.e. adjusts trust value and implements admission control policies. The system and policies match very closely with the human trust model, although we believe there is still a need to change the system to better emulate the human model. In this regard, we will start by changing the LTBAC system so that the partial trust results are combined to produce some trust element that is added or subtracted from T_{P_n}. By doing so, T_{P_n} will rise and fall more naturally from a start value of 0.50, rather than having to crawl its way up from 0.10, as it did in the presented trials.

Another interesting result is that once trust slips downwards, it becomes impossible to regain unless one of two events occur: 1) P_n pays all Cumulative Debt that is accrued, or 2) P_g cancels any debt owing, wiping the slate clean for P_n. In both scenarios, Current Debt is eliminated, which, in the current environment, would push T_{P_n} right up to the highest possible admission rights. We plan to do further research into this area, examining what human nature would be in these scenarios, e.g. how much debt is one willing to eliminate and how debt cancellation might affect future interactions. Moreover, a further issue arises of how much information must be stored and updated by each principal so that admission policies may be refined correctly on a dynamic basis. Also along these lines, we plan to integrate more debt repayment strategies to see if trust production follows Axelrod's cooperation production trials.

Finally, we were able to implement the majority of the trust framework described in Section 2. With future work, the remaining components of this framework can be implemented, which may require the specification of a complete certificate-based policy language.

6. Conclusions and Future Work

We identified that principals wishing to join ad hoc collaborative applications currently have no way of directly establishing trust in one another. There is a reliance on perceived trust or recommendations, which is too vague when security issues are being considered. Thus, we suggested that a trust-based admission control system using high-level trust-based security policies be implemented to provide a solution to the problem of establishing trust in open, diverse systems.

Our survey into trust and policy formation provided us with a framework for trust-based admission control as well as identifying areas of research into trust-based policy specification. Having investigated the state-of-the-art, we then designed and developed a collaborative ad hoc application framework and trust-based admission control system for the purposes of testing trust-based admission control. We have shown that a policy-based approach, centred around simple XML specifications, allows us to specify high-level trust-based admission control policies that are simple to use and have predictable results.

Based on our trials, we see that the trust-based admission control system reacts correctly to changes in a principal's behaviour, i.e. adjusts trust value and implements admission policies. We have noted areas of the framework that are not fully implemented and are subject to future work, i.e. a recommendation component and a risk component. We also found three issues with the current LTBAC system that we intend to develop further: reworking the trust engine to calculate a trust element that can be added or subtracted from the overall trust value; exploring more debt repayment strategies and appropriate policy responses to these strategies; and using high level policies to specify the linear equations used to calculate trust values rather than implementing these equations within the program code.

Acknowledgements

The authors wish to acknowledge the support provided by the SECURE project (IST-2001-32486), a part of the EU FET Global Computing initiative.

References

[1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote Trust-management System, version 2," IETF, RFC 2704, September 1999.

[2] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," in *IEEE Symposium on Security and Privacy*. Los Alamos: IEEE Computer Society Press, 1996.

[3] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," IETF, RFC 2693, September 1999.

[4] R. Rivest and B. Lampson, "SDSI - A SimpleDistributed Security Infrastructure," October 1996.

[5] S. Marsh, "Formalising Trust as a Computational Concept," Ph.D. dissertation, University of Stirling, Department of Computer Science and Mathematics, 1994.

[6] M. Deutsch, "Cooperation and Trust: Some theoretical notes," in *Nebraska Symposium on Motivation*, M. R. Jones, Ed. Nebrask University Press, 1962.

[7] R. Golembiewski and M. McConkie, *The Centrality of Interpersonal Trust in Group Processes*. Wiley, 1975, ch. 7, pp. 131–185.

[8] N. Luhman, *Trust and Power*. Wiley, 1979.

[9] B. Barber, *Logic and Limits of Trust*. New Jersey: Rutgers University Press, 1983.

[10] D. Gambetta, *Can We Trust Trust?* Dept. of Sociology, University of Oxford, 2000, ch. 13, pp. 213–237.

[11] B. Christianson and W. Harbison, "Why Isn't Trust Transitive?" in *Security Protocols Workshop*, 1996, pp. 171–176.

[12] P. Dasgupta, *Trust as a Commodity*. Dept. of Sociology, University of Oxford, 2000, ch. 4, pp. 49–72.

[13] D. McKnight and N. Chervany, "The Meanings of Trust," University of Minnesota, Management Informations Systems Research Center, University of Minnesota, MISRC 96-04, 1996.

[14] D. Povey, "Developing Electronic Trust Policies using a Risk Management model," in *CQRE [Secure] Congress*, November 1999.

[15] K. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation," in *Proceedings of the Symposium on Network and Distributed System Security*, February 2001.

[16] W. Winsborough, K. Seamons, and V. Jones, "Automated Trust Negotiation," in *DARPA Information Survivability Conference and Exposition*. IEEE Computer Society Press, January 2000.

[17] M. W. T. Yu and K. Seamons, "Interoperable Strategies in Automated Trust Negotiation," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, November 2001.

[18] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," in *Proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY)*, ser. LNCS 1995. Bristol, UK: Springer-Verlag, January 2001.

[19] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access Control meets Public Key Infrastructure, or: Assigning Roles to Strangers," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2000, pp. 2–14.

[20] N. Li, J. Mitchell, and W. Winsborough, "Design of a Role-based Trust-management Framework," in *IEEE Symposium on Security and Privacy*, Oakland, May 2002.

[21] W. Winsborough and N. Li, "Towards Practical Automated Trust Negotiation," in *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*. IEEE Computer Society Press, June 2002.

[22] Y. Mass and O. Shehory, "Distributed Trust in open Multi-agent systems," in *Trust in Cyber-societies*, ser. Lecture Notes in Computer Science, M. S. R. Falcone and Y.-H. Tan, Eds., vol. 2246. Springer, 2001.

[23] X. P. K. I. documents, http://www.ietf.org/ids.by.wg/pkix.html.

[24] S. Microsystems, *JINI Architecture Specification, Revision 1.0*.

[25] C. Jensen and M. Haahr, "Towards a Security Framework for Ad Hoc Applications," in *Proc. of the 2nd Workshop in Distributed Object Security, In Association with the OOPSLA'99 Conference on Object-Oriented Programming, Applications and Systems*, Denver, Colorado.

[26] R. Axelrod, *The Evolution of Cooperation*. New York: Basic Books, 1984.