Available online at www.sciencedirect.com

SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 157 (2006) 95–111

www.elsevier.com/locate/entcs

Trust Evolution Policies for Security in Collaborative Ad Hoc Applications

Elizabeth Gray, Christian Jensen[1,2]
Paul O'Connell, Stefan Weber, Jean-Marc Seigneur, Yong Chen

Department of Computer Science
Trinity College
Dublin, Ireland

Abstract

The vision of pervasive computing has introduced the notion of a vast, networked infrastructure of heterogeneous entities interact through collaborative applications, e.g., playing a multi-player online game on the way to work. This will require interactions between users who may be marginally known or completely unknown to each other, or in situations where complete information is unavailable. This introduces the problem of assigning access rights to such marginally known or unknown entities.
Explicit trust management has emerged as a solution to the problem of dealing with partial information about other users and the context in which the interaction takes place. We have implemented an access control mechanism based on the human notion of trust, where recommendations or initial participation in low risk interactions will allow entities to slowly build trust in each other. As the trust between two entities grows, interactions that entail a higher degree of risk may be allowed to proceed. We have used this mechanism in a simple role-based access control mechanism that uses trust to assign roles to users in a distributed blackjack card game application. This application allows us to experiment with different policies for trust-based admission control and trust evolution. In this paper we present an evaluation of policies specifying trust dynamics, which shows that our prototype reacts appropriately to the behaviour of other users and that the system updates trust values and implements admission policies in a manner similar to what would be expected from human trust assessment. This indicates that trust evolution policies can replace explicit human intervention in application scenarios that are similar to the evaluated prototype.

Keywords: Trust-based access control, trust dynamics, trust evolution.

[1] Contact Email: grayl@cs.tcd.ie
[2] Currently at: Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark. Email:Christian.Jensen@imm.dtu.dk

doi:10.1016/j.entcs.2005.09.038

1 Introduction

The value of the online game industry in Europe is predicted to rise from around $800 million in 2002 to just bellow $7 billion in 2006 [11]. The combination of improved display technologies and short-range wireless technologies, e.g., Bluetooth, being integrated into mobile phone handsets like the Nokia N-Gage [8], means that multi-player online games will soon be available any-time anywhere [23].

Difficulties arise when traditional security mechanisms are applied in a decentralised collaborative ad hoc environment, such as interactive online games, in which entities may organise themselves into groups on a dynamic basis, requiring each entity to determine its own levels of access control per application. For example, suppose Alice takes the same commuter train every weekday morning. To pass the time, she wishes to play an interactive game with other train passengers. 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, who knows about Alice, to determine if Alice should have access rights to participate in the blackjack game, but this is generally not possible in dynamic open systems. Moreover, making the traditional model work in this scenario would require that Alice and Bob shared a common authentication infrastructure, which suffers from problems of scalability and availability (the technology has existed for years, but no generally trusted authentication infrastructure has emerged.)

The uncertainty about other players exhibited in the blackjack game is not unlike situations faced by humans, who must regularly determine, with no assistance from a centralised 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 situations, where complete information is unavailable.

Trust management systems [4,5,10,22] attempt to enforce security policies in large-scale distributed networks through the use of credentials that delegate permissions. However, these systems focus on the static element of trust management and neglect the dynamic component of trust formation. That is, what does trust *really* consist of and how does trust evolve?

It is therefore natural to examine the human notion of trust in order to

derive computational trust models that correspond to human intuition, which will allow ordinary human users to configure and manage systems based on trust. However, human trust is a difficult concept to formalise and evaluate, and many different definitions of trust [18,9,17,3,12,6,7,1,25,21] exist. Moreover, the problem of trust evolution, i.e., determining how the results of current interactions will influence the future trust in the other party, has received little attention and trust evolution algorithms are often proposed with little or no formal justification [20,14].

In this paper, we describe a trust-based access control framework for ad-hoc applications, which has been used to evaluate trust evolution strategies according to the trust evolution framework proposed by Jonker and Treur [16]. We show that the results of applying automatic algorithms for trust evolution are consistent with human intuition, i.e., the results are as expected. This implies that automatic trust evolution may replace human intervention in the configuration and evolution of security policies in open dynamic systems, such as ubiquitous computing or the grid, which may play an important role in the successful uptake of those technologies [26].

The structure of the paper is as follows. Section 2 identifies the framework of human trust we use as a basis for trust evaluation. Section 3 specifies the design of the trust evolution function implemented in the admission control system component. This we use a simple distributed blackjack application to illustrate the use of trust in admission control, discussed in Section 4. In Section 5, we present the evaluation of the trust update process and discuss the results of the sample application. Finally, Section 6 gives conclusions and ideas for future work.

2 Framework for Trust

In order that unambiguous conversation may occur, a framework for evaluating trust is needed. McKnight and Chervany [19] define such a framework, as depicted in Figure 1.

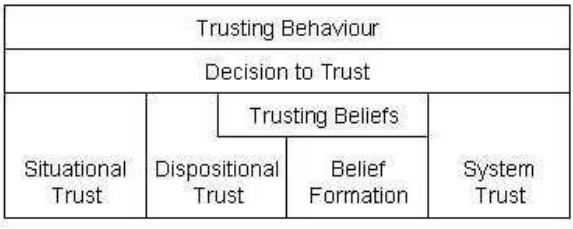

Fig. 1. McKnight and Chervany's Trust Framework

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

Situational Trust means that a 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* incorporates the subjective nature of trust, the extent to which a principal consistently trusts across a broad spectrum of situations and parties. This value reflects whether an individual is optimistic or pessimistic in their approach to new situations. *System Trust* reflects that safeguards are in place to reduce the amount of risk to which an entity must be exposed. 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. *Trusting Beliefs* are the extent to which one principal believes that another principal is willing and able to act in the trusting party's best interest. A *Trusting Intention* is formed when one principal makes a decision to depend on another party in a given situation with a feeling of relative security, even though negative consequences are possible. *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.

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. Overall, the implementation of such a framework will enable the use of trust-based tools such that principals might avoid the problems associated with trust and collaboration in an ad hoc environment.

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 return 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. When we design our trust-based admission control system, we do so based on this comprehensive human trust framework, such that its implementation will show that unknown computational entities can interact and exhibit trust formation.

3 Local Trust-Based Admission Control

This section presents the design of the Local Trust-Based Admission Control (LTBAC) component that was used to evaluate the different trust evolution strategies. [3]

As previously mentioned, the current state-of-the-art in formalising the trust evolution process is too imprecise to enable low-level access control decision-making. Instead, we propose to use trust-based admission control together with standard role-based access control, i.e., trust management is used to decide what role another principal may assume, while traditional role-based access control defines the access rights associated with that role.

Consider principal P_n requesting admission to collaborate in ad hoc group, G, in a particular role. P_n will typically be interested in joining the role with the most access rights. LTBAC is executed by each member of G, P_g, each time a decision is required regarding P_n's request for admission to the group. Each P_g verifies whether or not P_n is trustworthy [4] enough to join G in the requested role, according to his own local policy criteria, and then provides output to the group. This way, we approximate McKnight and Chervany's trust-based decision-making framework as closely as possible, specifically including the subjective and contextual elements of trust formation.

When the LTBAC component is invoked, P_n's identifier and requested role are passed into each P_g's trust engine, which uses local trust-based policies to determine whether or not P_n is trustworthy enough to join G. P_g provides an output to G according to whether or not P_n meets P_g's local policy criteria.

One of the design goals for the LTBAC component is to minimise the amount of configuration that a user must perform. The LTBAC component design therefore includes a number of support components to gather trust-related information, manage policies, and produce trust values. Figure 2 presents an overview of the LTBAC component, illustrating which subcomponents are generic and which are application-specific. The functionality of the LTBAC subcomponents is described in Figure 3.

When P_n is requesting admission to G, his role and identifier are passed into P_g's LTBAC system, commencing the trust evaluation process.

The *Trust System* consists of generic trust-based functions, i.e. functionality that is application-independent. These include the Compliance Checker,

[3] The design and implementation of the complete system and its components is described in full in Gray et al [13].

[4] In this paper, we use the terms trust and trustworthy in their human sense, i.e., an entity is trusted (believed to be trustworthy) if we choose to trust it. This is different from the traditional security literature, where an entity is "trusted" if it can hurt you and if there is no way to mitigate that risk.

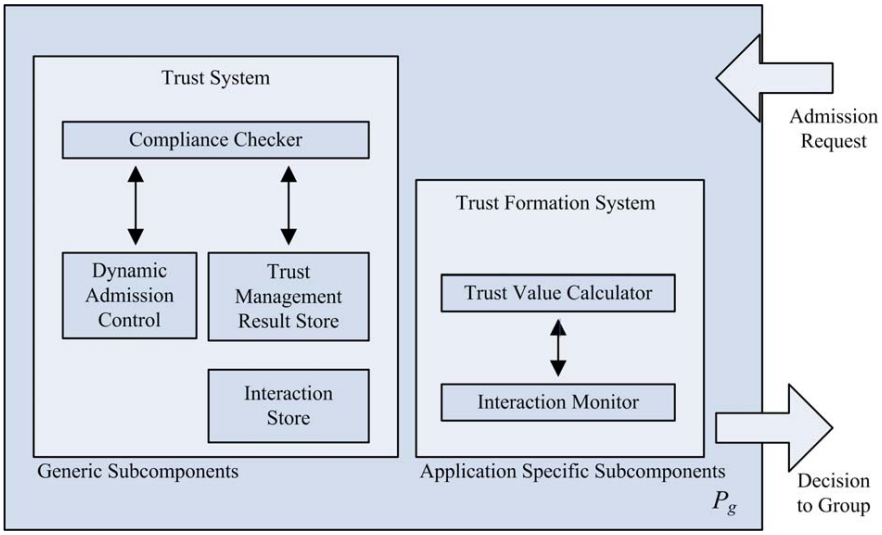

Fig. 2. Local Trust-Based Admission Control (LTBAC)

LTBAC Subcomponent	Functionality
Compliance Checker	Gathers admission control policy statements that are relevant to P_n. Identifies P_n's trust value from the Trust Result Store. Compares 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, informs P_g that P_n's admission request may be accepted.
Dynamic Admission Control	Binds a trust value to the right of admission to a role. Admission control statement includes: • Context, 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; and, • Role, name of the role P_n will be approved for if the trust value meets the applied condition.
Interaction Monitor	Gathers and saves information about principals' interactions. Information is application-specific, based on principals' interactions in a specific context.
Trust Result Store Interaction Store	Store interaction information and trust results. Ensure that the interaction information and trust results are passed in the correct format between the Trust System and the Trust Formation System
Trust Value Calculator	Processes the results of the current interaction to adjust the trust that P_g has in P_n.

Fig. 3. LTBAC Subcomponent Functionality

Dynamic Admission Control, and the Trust Management Result and Interaction Stores.

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

As highlighted in Figure 3, the *Trust Value Calculator* processes the results

of the current interaction to adjust $T(P_g)(P_n)$, the trust that P_g has in P_n. The trust value that a future interaction will depend on is derived from the transaction history $H(P_g)(P_n)$ and the outcome of the current interaction that has been evaluated according to the principal-specified trust evolution policy, $\sigma(P_g)$. This calculation is described by Equation 1.

$$T(P_g)(P_n) = H(P_g)(P_n) + \sigma(P_g) \tag{1}$$

If P_g has had previous interactions with P_n, the historical trust $H(P_g)(P_n)$, i.e. the trust $T(P_g)$ has accrued in P_n to-date, can be accessed from the Trust Management Result Store[5]. The Trust Value Calculator increases or decreases $H(P_g)(P_n)$ using a trust evolution policy, $\sigma(P_g)$. The resulting, updated trust value, $T(P_g)(P_n)$, is expressed as a value in the interval [0,1], where the higher $T(P_g)(P_n)$ is, the more likely it is that the interaction with P_n will have a positive outcome. $T(P_g)(P_n)$ becomes asymptotic as it approaches one or zero, reflecting that a principal can never be completely trusted or distrusted.

If P_g has had no previous interactions with P_n, there needs to be a way of establishing an initial trust level. Referring back to the human notion of trust, this can be done by relying on disposition, that is, the trusting nature of P_g.[6] In our design, P_g may be initially very trusting (allowing all interaction), moderately trusting (allowing some interaction), or very distrusting (allowing no interaction) upon initial interaction with an unknown P_n.

$\sigma(P_g)$, which is used to determine how much to increase or decrease $H(P_g)(P_n)$, is defined by a context-specific trust update function, f_{trust}. f_{trust} is designed such that trust may evolve according to dispositional and situational evaluation of interaction results. The results of the current interaction are passed in and evaluated to produce $\sigma(P_g)$. The trust evolution policy depends on both the context-specific interaction information and the disposition of P_g.

To update $H(P_g)(P_n)$ based on results from the current interaction, P_g consults f_{trust}, which is specified by policy such that trust may evolve according to dispositional and situational evaluation of interaction results. The dispositional trust schema we incorporate into the trust evolution function design is based on the framework proposed by Jonker and Treur [16], which defines a

[5] When stored, trust results are broken down into four pieces of information: *Context*, the name of the application to which the result applies; *Identifier*, the name of the principal to which the trust value relates, e.g. P_{538}; *Trust Value*; and *Date* on which the trust value was calculated. It is important to note that identity is not necessarily based on real-world identity, but is assessed based on the recognition of entities, as per the Entity Recognition process described in [24]

[6] Another bootstrapping possibility is for P_g to request recommendations as to the trustworthiness of P_n.

set of six different trust evolution policies: blindly positive, P_g increases trust in P_n unconditionally; blindly negative, P_g decreases trust in P_n unconditionally; slow positive, fast negative, P_g is slow to increase trust in P_n and fast to decrease trust; fast positive, slow negative: P_g is fast to increase trust in P_g and slow to decrease trust; balanced slow: P_g is slow in both increasing and decreasing trust in P_n; balanced fast: P_g is fast in both increasing and decreasing trust in P_n. This defines a generic framework that can be used to define many trust evolution policies, e.g., the Tit-for-Tat [2] strategy, well-known in cooperation evolution studies, can be defined as a balanced fast trust evolution policy. The integration of this generic trust evolution framework into our prototype, allows us to perform a methodical evaluation of the different trust evolution policies.

4 Sample Application

We implemented the blackjack card game as a sample application for evaluating trust update policies and trust-based admission control in ad hoc collaborative environments. 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.

A satisfactory trust value is one that is updated to a level that will be successful in gaining admission to a game when compared to other users' admission policies. The perceived benefit of maintaining a satisfactory trust value in the blackjack application is the ability to continue gaining admission to desirable blackjack "tables." We associate no specific semantics with trust values, they simply allow principals to be compared to each other and to define local admission control policies. [7] However, we generally expect higher trust values to reflect a higher probability that the other party will behave in a competent and benevolent way, which in blackjack includes repaying gambling debts and regularly being available for games.

For boot-strapping purposes, an unknown P_n is assigned an initial trust value, which allows a first interaction with even the most distrusting players. In this implementation, all players are required to connect directly to the user who has the dealer role.

[7] This means that trust values cannot be exchanged directly among principals in the system.

During the game, each P_g monitors the actions of the other group members such that the resultant interaction behaviour data might be used to update trust values accordingly. Some actions that would be typically monitored in casinos include: time, number of interactions, payment record, location, playing strategy, betting strategy, win rate, usual co-players, and level of addiction. In our prototype application, date, number of interactions, and payment record are monitored by the Interaction Monitor and stored in the Interaction Store.

The Trust Value Calculator may then use these partial results to update an overall trust value of any P_n, using Equation 1 above. To determine the trust dynamic, $\sigma(P_g)$, the Trust Value Calculator consults the trust update function f_{trust}. In this example, f_{trust} is calculated based on the product of the three aforementioned context-specific parameters: the number of interactions (T_s), the number of days since the last interaction (T_t), and the payment record (T_d). Each parameter expresses the trustworthiness for that particular type of behaviour. When combined to a trust value based on all monitored behaviours, T_{std}, T_s and T_t each contribute 25% to the result and T_d, the partial result dealing with debt repayment, contributes 50%.

The resultant trust value, T_{std}, is evaluated in light of historical trust, $H(P_g)(P_n)$, to produce σ, which specifies how much or how little the Trust Value Calculator should adjust the overall trust value. f_{trust} increments are currently specified for the blackjack context as shown in Table 1. [8]

	Blind Positive	Fast Positive, Slow Negative	Balanced Fast	Balanced Slow	Slow Positive, Fast Negative	Blind Negative
$T_{std} > H(P_g)(P_n)$	Δ	Δ	Δ	δ	δ	$-\Delta$
$T_{std} < H(P_g)(P_n)$	Δ	$-\delta$	$-\Delta$	$-\delta$	$-\Delta$	$-\Delta$
$T_{std} = H(P_g)(P_n)$	Δ	0	0	0	0	$-\Delta$

Table 1
Trust Dynamic, σ

The Admission Policy Result stores P_g's admission control policies, whch for the purpose of this evaluation are: Blind Trust, High Trust, Medium High Trust, Medium Low Trust, Low Trust, or Blind Distrust. These six policies are based on criteria associated with a range of trust levels, depending on whether P_g is of a disposition to be highly trusting, moderate, or highly distrusting in the context of blackjack.

The Trust Management Result stores blackjack-specific trust values for principals. The Interaction Result stores the monitored results to keep a record of how many sessions have been played, the date of the last interaction,

[8] Δ = large increment. δ = small increment

and the amount of debt owing. Finally, the Trust Dynamic Result allows P_g to tailor the trust update function according to a selection between the six trust dynamics specified above.

5 Evaluation

Our assessment of this system is performed through a controlled sequence of interactions in a blackjack game, wherein we evaluate both LTBAC and f_{trust} policies, as specified above.

5.1 Test Parameters

In order to evaluate trust evolution policies, we need to consider how each policy reacts to different types of behaviour by other players and, ultimately, whether the decisions by the trust-based admission control mechanism corresponds to the decisions achieved through human intervention. [9]

In each collaborative group, or gaming table, there are two potential roles, one dealer and one player. The principal in the role of dealer, P_d, interacts with the principal in the role of player, P_p, for 100 blackjack games. In our evaluation, P_p loses every game, which allows us to focus on the effects of the two primary parameters: frequency of interaction and payment of debts. Initially, unknown P_p is permitted access to a first interaction with an initial trust value of .50.

Each principal is evaluated with a LTBAC policy of Blind Trust, High Trust, Medium High Trust, Medium Low Trust, Low Trust, or Blind Distrust; Figure 4 shows the values currently specified by these six policies. Moreover, each principal is evaluated with a trust evolution policy, f_{trust}, from Blind Positive, Fast Positive Slow Negative, Balanced Fast, Balanced Slow, Slow Positive Fast Negative, and Blind Negative. Δ is specified as .05, and δ is .005.

We evaluate trust evolution and admission control over 100 interactions and in three debt repayment cases. The debt repayment strategies are based on concepts from Axelrod's [2] cooperation studies. Having lost a game, a principal will Always Pay (AP), Never Pay (NP), or Randomly Pay (RP). The randomness of payment/non-payment for the RP case was generated with a random numbers generator [15].

As each game is played, we record $T(P_d)(P_p)$ based on the three parameters specified earlier: number of interactions, number of days since the last interaction, and payment record. All games are played on the same day, so as

[9] This may be considered a simple example of a Turing test.

Admission Policy	Context	Applies To	Condition	Trust Value for Role
Blind Trust	Blackjack	*	≥	.01 dealer .01 player
High Trust	Blackjack	*	≥	.25 dealer .10 player
Medium High Trust	Blackjack	*	≥	.50 dealer .25 player
Medium Low Trust	Blackjack	*	≥	.75 dealer .50 player
Low Trust	Blackjack	*	≥	.90 dealer .75 player
Blind Distrust	Blackjack	*	≥	.99 dealer .99 player

Fig. 4. LTBAC Policy Criteria for Blackjack

to better isolate the monitored effect of number of interactions and payment record, i.e., the parameter T_t is fixed within this experiment.

We define *correct behaviour* on the part of P_p as consistent interaction and payment in full of debts. Correct system behaviour refers to the ability of the LTBAC system to form and evolve trust, as well as accept and reject admission requests, in a manner corresponding to human intuition, i.e. such that a human user would feel safe in delegating these actions to the system.

5.2 Results and Analysis

The following figures show how $T(P_d)(P_p)$ increases and decreases over a series of interactions. For each of these cases, we discuss the trust evolution as well as the usefulness of the LTBAC policies specified in Figure 4.

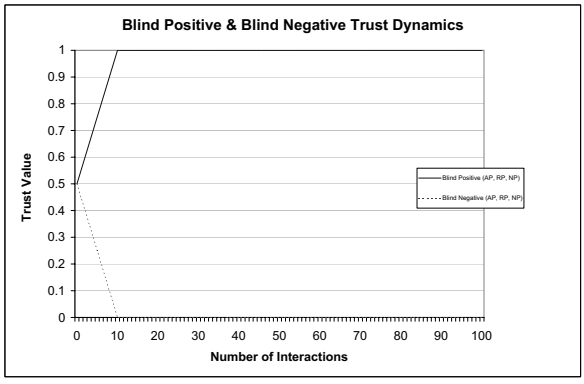

Fig. 5. Trust Value Evolution Using Blind Positive or Blind Negative Trust Dynamic

First, we treat the Blind Trust and Blind Distrust dynamics as a special case. We see in Figure 5 that, regardless of number of interactions or whether

or not debt was paid, a blindly trusting P_d evolves trust increasingly by 5% per interaction to the upper limit, and P_d who is blindly distrusting does the opposite. These two trust evolution dynamics are unique in that $T(P_d)(P_p)$ reaches and rests at the limits. Moreover, the recorded trust levels successfully emulate the very gullible and very cynical extremes of human behaviour.

The LTBAC policies work appropriately in both the Blind Trust and Blind Distrust cases. If P_d implements the Blind Trust admission policy, he allows access to all roles once $T(P_d)(P_p) \geq .01$. Therefore, P_d might implement a Blind Trust evolution policy with any other admission policy, even that of Blind Distrust, resulting in all P_p gaining admission to all roles with $T(P_d)(P_p) = 1.0$. Fittingly, P_d implementing a Blind Distrust evolution policy with a Blind Distrust admission policy allows no further interaction at all.

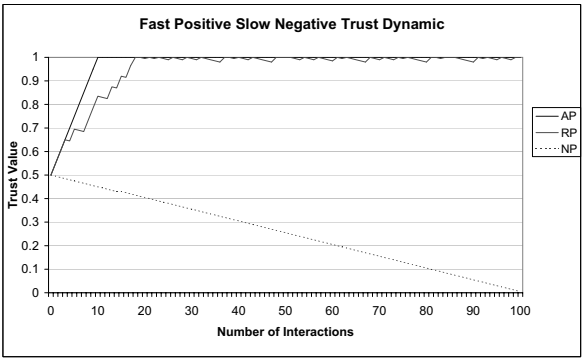

Fig. 6. Trust Value Evolution Using Fast Positive Slow Negative Trust Dynamic

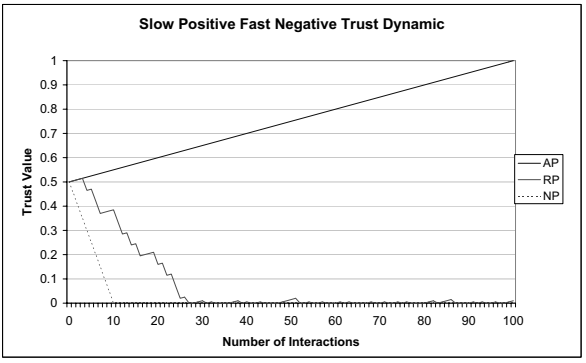

Fig. 7. Trust Value Evolution Using Slow Positive Fast Negative Trust Dynamic

In Figure 6, we see that $T(P_d)(P_p)$ advances rapidly when P_p always exhibits correct behaviour and declines slowly when incorrect behaviour consistently occurs. When cumulative debt is paid in full on some occasions but not others, $T(P_d)(P_p)$ rises rapidly with payment and drops slowly with

non-payment. Because P_d is dispositionally positive in his trusting of P_p, random non-payment does not keep $T(P_d)(P_p)$ from rising to the upper limit and fluctuating around that limit.

The opposite is true in trust evolution using the Slow Positive Fast Negative Trust Dynamic, displayed in Figure 7. In this case, $T(P_d)(P_p)$ advances slowly when P_p always exhibits correct behaviour and declines rapidly when incorrect behaviour consistently occurs. Thus, when cumulative debt is paid at random, $T(P_d)(P_p)$ falls sharply with incorrect behaviour and rises slowly with correct behaviour. Because P_d is dispositionally negative in his trusting of P_p, random payment is not enough for $T(P_d)(P_p)$ to rise out of a rapid decline to the lower bound.

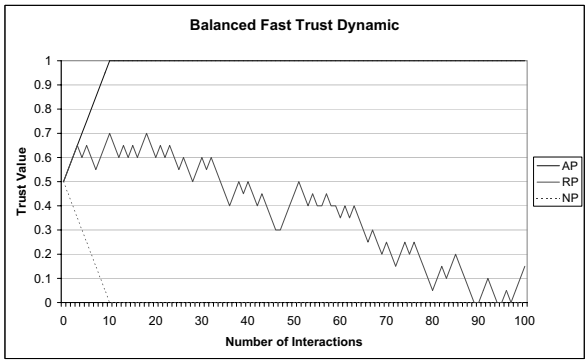

Fig. 8. Trust Value Evolution Using Balanced Fast Trust Dynamic

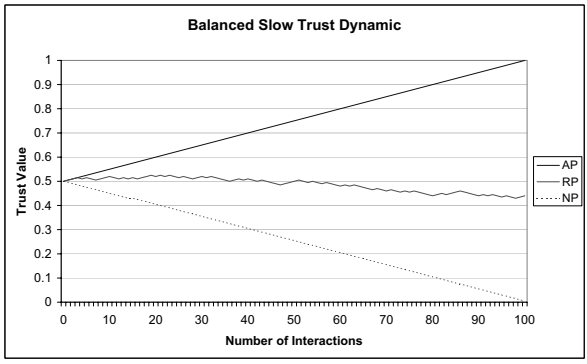

Fig. 9. Trust Value Evolution Using Balanced Slow Trust Dynamic

In the case of Balanced Fast trust evolution, Figure 8, and Balanced Slow trust evolution, 9, we find that trust evolves the most naturally of all cases. In the Balanced Fast case, $T(P_d)(P_p)$ advances rapidly in face of correct behaviour and declines equally rapidly when debt is not paid. When cumulative debt is paid at random, $T(P_d)(P_p)$ rises and falls sharply according to which

type of behaviour is displayed. Over time, however, $T(P_d)(P_p)$ evolves rapidly downward as incorrect behaviour is continuously intermittently displayed. The same results occur in the Balanced Slow case, only much more gradually, with no sharp increases or decreases. Because P_p is rewarded only slightly for good behaviour and punished only slightly for bad behaviour, $T(P_d)(P_p)$ hovers around the moderate range until over time it commences a very slow decline as incorrect behaviour is continuously intermittently displayed. Overall, the Balanced Slow dynamic shows the smoothest trust evolution, due to the dispositional lack of large jumps in response to behaviour that the other trust dynamics specify.

In Figures 6 to 9, we begin to see increases and decreases in trust in a collaborative ad hoc application environment that follow closely the way human trust rises and falls based on disposition, interactions, and resultant behaviour. When interactions increase, trust grows steadily, unless the interactions become increasingly tainted by incorrect behaviour, i.e. not paying one's debts. Trust evolution in these four cases varies dispositionally appropriately according to the selected trust dynamic.

We find overall that the LTBAC policies are appropriate in all evolution cases, allowing and rejecting admission based on trust values that are more or less sharply evolving per dynamic. Additionally, each admission policy follows closely to the human decision-making process as to whether or not another entity is trustworthy enough to participate in a future interaction in a given context. Some policies, like some humans, are more restrictive in allowing interaction than others. With correct behaviour, $T(P_d)(P_p)$ rises, and P_p may be admitted to games in which P_d has a more restrictive admission policy. As incorrect behaviour increases, more restrictive admission policies ensure that admission requests are rejected.

Analysis of the LTBAC system illustrates that trust may be formed and evolved automatically by principals in a collaborative ad hoc application in a manner very similar to human trust. Additionally, we find that the LTBAC system behaves correctly in permitting or denying access to resources using a range of human-like restrictions, i.e. by adjusting trust value and implementing LTBAC policies. The system and policies match very closely with the human trust model, although we believe more complexity must be added.

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 and evolving trust in open distributed systems. The success of a trust-based security mechanisms depends on its ability to automatically make decisions that correspond to human what a human would have done in the same situation, i.e. ordinary people should be able to configure and operate the system so that decisions made automatically by the system corresponds to the decisions they would have made themselves,

Having adopted the McKnight and Chervany human trust framework as the basis for our trust definitions, we then designed and developed a prototype application and trust-based admission control system for the purposes of testing trust-based admission control.

We have shown that a policy-based approach, allows us to specify high-level trust-based trust evolution and admission control policies that are simple to use and have predictable results. Based on our experiments, we see that the trust-based admission control system reacts correctly to changes in a principal's behaviour, i.e. evolves trust values and implements admission control policies in a manner similar to that of humans. In particular, we used an existing trust evolution framework to show that automatic trust evolution does indeed deliver results that correspond to the human intuitions about trust. This means that simple dispositional policies and automatic trust evolution may provide an answer to the problem of developping "calm" security technologies that do not require constant human intervention.

Finally, we foresee many promising future developments of this work. First of all, we wish to experiment with large scale deployment of this technology, e.g., through the use of Bluetooth enabled PDAs or mobile phones. The parameters considered for trust evolution in this paper are very important, but they are also very simple. Therefore we wish to examine more complex trust evolution strategies, e.g. a strategy that attempts to determine the other player's strategy and rates the consistency with which that strategy is followed. Finally, future work in the area of group behaviour, i.e., extending trust evolution dynamics to situations in which more than two principals are interacting, is of interest.

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] Abdul-Rahman, A. and S. Hailes, *Supporting trust in virtual communities*, in: *Proceedings Hawaii International Conference on System Sciences (HICSS-33)*, Maui, Hawaii, U.S.A., 2000.

[2] Axelrod, R., "The Evolution of Cooperation," Basic Books, New York, 1984.

[3] Barber, B., "Logic and Limits of Trust," Rutgers University Press, New Jersey, 1983.

[4] Blaze, M., J. Feigenbaum, J. Ioannidis and A. Keromytis, *The KeyNote Trust-management System, version 2*, RFC 2704, IETF (1999).

[5] Blaze, M., J. Feigenbaum and J. Lacy, *Decentralized Trust Management*, in: *IEEE Symposium on Security and Privacy*, 1996, pp. 164–173.

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

[7] Dasgupta, P., "Trust as a Commodity," 2000 .

[8] Deck, N. N.-G. G., *Tech spex*, Technical report, http://www.n-gage.com/en-R1/gamedeckngage_qd/, visited 20 August, 2005.

[9] Deutsch, M., *Cooperation and Trust: Some Theoretical Notes*, in: M. Jones, editor, *Nebraska Symposium on Motivation* (1962).

[10] Ellison, C., B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen, *SPKI Certificate Theory*, RFC 2693, IETF (1999).

[11] Frost & Sullivan, *European mobile gaming market*, Market Research Report, R1-2527 (2003).

[12] Gambetta, D., "Can We Trust Trust?" Dept. of Sociology, University of Oxford, 2000 pp. 213–237.

[13] Gray, E., P. O'Connell, C. Jensen, S. Weber, J.-M. Seigneur and C. Yong, *Towards a Framework for Assessing Trust-Based Admission Control in Collaborative Ad Hoc Applications*, Technical Report 66, Dept. of Computer Science, Trinity College Dublin (2002).

[14] Griffiths, N. and K.-M. Chao, *Experience-based trust: Enabling effective resource selection in a grid environment*, in: *Proceedings of the Third International Conference on Trust Management*, Paris, France, 2005.

[15] Haahr, M., *random.org*, http://www.random.org, visited 22 Augist 2005.

[16] Jonker, C. M. and J. Treur, *Formal analysis of models for the dynamics of trust based on experiences*, in: F. J. Garijo and M. Boman, editors, *Multi-Agent System Engineering (MAAMAW-99)*, Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World **1647** (1999), pp. 221–231.

[17] Luhman, N., "Trust and Power," Wiley, 1979.

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

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

[20] Michalakopoulos, M. and M. Fasli, *On deciding to trust*, in: *Proceedings of the Third International Conference on Trust Management*, Paris, France, 2005, pp. 61–76.

[21] Nielsen, M. and K. Krukow, *On the formal modelling of trust in reputation-based systems*, in: G. P. J. Karhumki, H. Maurer and G. Rozenberg, editors, *Theory is Forever: Essays Dedicated to Arto Salomaa*, number 3113 in Lecture Notes in Computer Science, Springer Verlag, 2004 pp. 192–204.

[22] Rivest, R. and B. Lampson, *SDSI - A SimpleDistributed Security Infrastructure*, Presented at CRYPTO'96 Rumpsession (1996).

[23] Saltzman, M., *Bluetooth on the go*, N-Gage website (2004), http://www.n-gage.com/R1/en/newsevents/articles/newsevents_article_020204.htm, visited 19 August, 2005.

[24] Seigneur, J.-M., S. Farrell, C. Jensen, E. Gray and Y. Chen, *End-to-end trust in pervasive computing starts with recognition*, in: *Proceedings of the First International Conference on Security in Pervasive Computing*, Boppard, Germany, 2003.

[25] Weeks, S., *Understanding trust management systems*, in: *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, U.S.A., 2001, pp. 94–105.

[26] Weiser, M. and J. S. Brown, *Designing calm technology* (1995), http://www.ubiq.com/hypertext/weiser/calmtech/calmtech.htm, visited 20 August 2005.