10

# Specifying and Analysing Trust for Internet Applications

Tyrone Grandison, Morris Sloman
*Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ*

**Abstract**: The Internet is now being used for commercial, social and educational interactions, which previously relied on direct face-to-face contact in order to establish trust relationships. Thus, there is a need to be able to establish and evaluate trust relationships relying only on electronic interactions over the Internet. A trust framework for Internet applications should incorporate concepts such as experience, reputation and trusting propensity in order to specify and evaluate trust. SULTAN (**S**imple **U**niversal **L**ogic-oriented **T**rust **A**nalysis **N**otation) is an abstract, logic-oriented notation designed to facilitate the specification and analysis of trust relationships. SULTAN seeks to address all the above issues, although this paper focuses on our initial work on trust specification and analysis.

**Keywords**: Trust specification, Trust analysis, Trust management

## 1. MOTIVATION

Trust plays an important role in all E-commerce interactions. Customers must trust that sellers will provide the services they advertise, and will not disclose private customer information. Trust in the supplier's competence and honesty will influence the customer's decision as to which supplier to use. Sellers must trust that the buyer is able to pay for goods or services, is authorised to make purchases on behalf of an organisation or is not underage for accessing service or purchasing certain goods. Thus, trust management has to be an intrinsic part of E-commerce, for it to achieve the same acceptance levels as traditional commerce. However, business transactions span multiple organisations, possibly in different countries and not all of these domains may be trusted to the same extent. A domain may need to

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35617-4_48

J. L. Monteiro et al. (eds.), *Towards the Knowledge Society*

support a range of different trust relationships and hence be capable of supporting different types of security policy (Swarup, Schmidt, 1998). Applications will need to be able to navigate through these possibly inconsistent trust relationships. There is a need for a general-purpose trust management system that supports specification and reasoning about trust and its relationship to risk and experience for Internet applications.

A trust relationship occurs between a *trustor*, the subject that trusts a target entity, called the *trustee*. A trustor or trustee may be a collective entity such as a company, committee or partnership. The essential components of a trust relationship are: the trustor, the trustee, a specific context with associated level of trust, and the conditions under which this relationship becomes active (Grandison, Sloman, 2000).

We define **trust** as a quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context. Trust is not symmetric, so this belief by the trustor does not imply any similar belief by the trustee.

**Distrust** is a quantified belief by a trustor that a trustee is incompetent, dishonest, not secure or not dependable within a specified context. Quantification reflects that a trustor can have various degrees of trust (distrust), which could be expressed as a numerical range or as a discrete classification such as low, medium or high.

The context of a trust relationship is defined as a set of actions with a trust level applying to all the actions, and a set of constraints, which must be evaluated for the trust relationship to apply.

We are developing SULTAN (Simple Universal Logic-oriented Trust Analysis Notation) – a notation with associated tools for specification, analysis and management of trust relationships for Internet applications. We envisage the SULTAN trust management framework being a component within a set of on-line management tools for distributed systems. Most of the current trust management frameworks are aimed at authentication or access control, whereas we are aiming at a more abstract form of trust specification which can then be analysed and evaluated, although it may be used as the basis for generating access control and authentication policies. We eventually hope to use SULTAN trust specifications as a starting point for deriving Ponder security management policies (Damianou et al, 1995) for authorisation and authentication or to make trust-based authorisation decisions.

This paper provides a high-level discussion of the framework, with particular emphasis on the trust specification and analysis approach. Section 2 describes the SULTAN trust management framework followed by details of the specification notation in Section 3. The SULTAN analysis model is outlined in section 4. In section 5 we show a case study that illustrates the SULTAN specification notation and analysis model. In section 6 we discuss how SULTAN fits into our on-line management tools. Section 7 compares SULTAN to related work in the field and we conclude in section 8.

## 2. SULTAN TRUST MANAGEMENT

We define trust management as "the activity of collecting, codifying, analysing and evaluating evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships for Internet applications" (adapted from Joesang, 2000). SULTAN's Trust Management consists of the following components:

**Trust establishment** defines the protocols by which the parties negotiate and exchange the evidence and credentials, which are needed for **evaluating** trust in order to define a trust relationship. **Trust analysis** involves checking the semantic properties of the trust and recommendation specifications to determine conflicts and implicit relationships.

A **Specification Server** holds all the trust and recommendation specifications for the administrator's domain. Trust ratings for a service may depend on past usage experience, so a **monitoring service** updates state information related to scenarios being evaluated or from a actual systems, as well as experience information, such as number of successful interactions, which can be used to increase trust level ratings directly. In general, higher risk implies less trust, but risk evaluation may depend on factors such as value of a transaction, who is providing a service etc. The **Risk Evaluation Service** indicates the risk involved in interactions between the trustor and trustee, as a trust specification may depend on risk. We define risk as a probability of a failure with respect to the context of the interaction e.g. non-payment for service, a security failure or service failure.

Figure 1 shows the SULTAN trust management architecture and how the components of our framework interact. Arrowheads denote the direction of information flow. Initially, the administrator may use the specification editor to either create and/or edit specifications. Specifications can be analysed to determine if any conflicts exist in the source or specific scenarios can be analysed taking into consideration state referenced in constraints. The monitoring service updates state information.

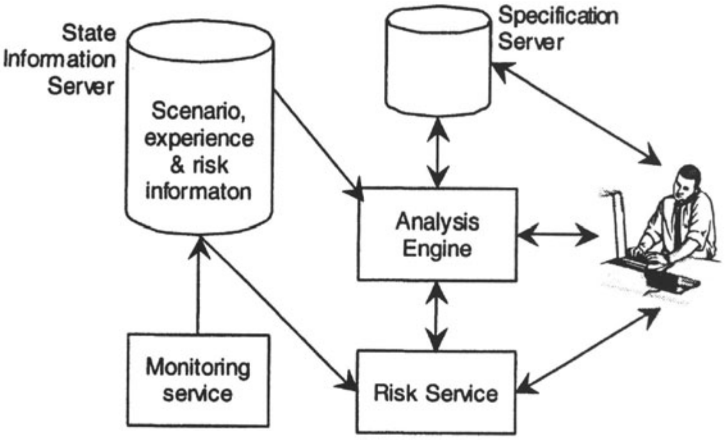

*Figure 1.* SULTAN Trust Management Architecture

We assume that components of the framework can be replicated (for reliability and availability reasons) and that security mechanisms control access to the trust specifications. We turn our focus to the specification of trust relationships in SULTAN.

## 3. TRUST SPECIFICATION CONSTRUCTS

SULTAN has two constructs for specification: the trust construct and the recommend construct.

### 3.1 Trust construct

This is used to specify both a trust and distrust relationship.

> PolicyName : **trust** ( Trustor, Trustee, ActionSet, Level ) ← ConstraintSet;
>
> Trustor trusts/distrusts Trustee for a context specified by ActionSet at trust/distrust Level if ConstraintSet is true; where PolicyName is a unique name for the assertion, Trustor is the entity that is trusting and Trustee is the trusted entity.

ActionSet is a list of colon-delimited actions, which specify the context for the trust rule. The first parameter in an action name specifies where the action is to be performed (whether on the trustor, or the trustee, or one of their resources). The Level of a trust relationship is typically defined by a set of discrete values – low, medium, high. However, there is a need to perform arithmetic operations on levels, for example, to calculate averages. We have arbitrarily decided to represent it as an integer in the range 1 to 100 for trust and -1 to -100 for distrust. The same level applies to all actions in the ActionSet. ConstraintSet is the set of constraints that must be met for this relationship to be established (these can be either user-defined or SULTAN-defined constraints). Constraints must evaluate to a Boolean value.

Examples:

S1: **trust** (Winnifred, Bank, NewAccount(Bank), 50) ← online(Bank);

Winnifred trusts Bank with respect to opening an account at Bank at a medium level (50) if the bank provides online access.

S2: **trust** (Depositor, NBC, NewAccount(NBC), -100);

Depositor distrusts NBC with respect to opening an account at NBC with a high level.

## 3.2 Recommend Construct

A recommendation can be positive or negative and can be used as the basis to define new trust relationships.

**PolicyName :** recommend **( Recommendor, Recommendee, ActionSet, Level )**
**← ConstraintSet;**

Recommendor recommends/does not recommend Recommendee with a confidence of Level to perform ActionSet if ConstraintSet is true; where PolicyName is the unique name of the rule being defined, Recommendor is the entity making the recommendation and Recommendee is the entity that the recommendation is about.

The attributes of a recommend rule are very similar to those of a trust relationship. ActionSet defines the context. A positive value for Level indicates a recommend while a negative value indicates a non-recommend. ConstraintSet is the set of constraints that must be met for this recommendation to be established (these can be either user-defined or SULTAN-defined constraints).

A recommendation does not imply that the recommendee must be trusted. That is up to the trustor to decided. A recommendation may result in a trust specification or form a constraint within a trust specification, but the trust level need not correspond to the recommendation level.

Examples:

R1: **recommend** (BestShopper, Sainsbury, buy_products(Sainsbury), HIGHREC) ←
BasketCost(Sainsbury) < £40;

BestShopper recommends Sainsbury to provide a buy_products service with high confidence if the cost of a standard basket of products at Sainsbury is less than £40.

R2: **recommend** ( John, Fred, installOS( staffPC), -50);

John does not recommend Fred at medium level to perform installOS on staffPC.

R9: **recommend** (Morris, ICStudents, design_software(ICStudents), HIGH) ←
**trust**(Naranker, ICStudents, program(ICStudents),MEDIUM);

Morris highly recommends ICStudents to provide a design_software service if Naranker reasonably trusts them (at MEDIUM level) to program.

S10: **trust** (John, Fred, installOS(JohnPC), MEDIUM) ←
**recommend** (Peter, Fred, installOS( _ ), HIGHLYREC);

John trusts Fred to perform installOS on his PC at MEDIUM level if Peter highly recommends Fred at to perform installOS on any computer. (A SULTAN variable is zero or more characters (alphabetic or numeric) preceeded by an underscore).

The above examples show that trust specifications can be based on recommendations or recommendations can be based on trust assertions. We now look at the SULTAN analysis model.

## 4. ANALYSIS MODEL

Analysis of the trust and recommendation rules is based on checking whether specified properties hold. The properties can be with respect to the specification source, which is essentially program reasoning. Source analysis ignores the constraints, i.e. assume they are true. The properties can also be with respect to examining trust relationships to identify scenarios of interest. Scenario analysis involves reasoning about the state of the system, and the current state of constraints. For example, in recommendation R1, in section 3.2, the system must have a value for BasketCost(Sainsbury) in order to evaluate the constraint. The monitoring system would be responsible for updating this information.

The prototype for the analysis model is implemented in Prolog. This allows arbitrary application specific analysis to be performed to meet the requirements of a particular organization. To perform an analysis query, we use the rules in the analysis model. A query is a predicate which defines the property that the administrator is interested in, and is used to retrieve a set of results. The format of this construct is:

```
query(Vars, Conds, ResultSet).
```

This finds all Vars that meet condition(s) Conds and stores the result in ResultSet, where Vars are the variables to be collected that must be used in the Conds section; Conds are the conditions to be satisfied and ResultSet is the set of Vars that meet Conds. Note that Conds can be any application-specific predicate defined by the administrator relating to the source rules, scenario data or a system integrity check.

Examples:

```
query( [T,D], ( p_pos_trust(T), p_neg_trust(D), p_trustor(Tr,T), p_trustor(Tr, D),
                p_trustee(Te,T), p_trustee(Te, D), p_actionset(ACT, T),
                p_actionset(ACT, D) ), Result).
```

Is there a trust rule and a distrust rule (in my specification source) concerning the same trustor, trustee and actionset? (This is an example of source code analysis).

```
query( [T, NR], ( pos_trust(T), neg_rec(NR), subject(Rr,T), subject(Rr, NR),
                  target(Re,T),  target(Re, NR), actionset(ACT, T),
                  actionset(ACT, NR) ), Result).
```

Is there a scenario in which there is a trust relationship and a recommendation that concern the same subject, target and actionset? (This is an actual scenario analysis)

```
query( [T1,T2], ( pos_trust(T1), pos_trust(T2), T1 \== T2, trustee(Te,T1),
                  trustee(Te, T2), actionset(A, T1), actionset(A, T2) ), Result).
```

Is there a scenario in which there are two trust relationships, which involve the same trustee and actionset? (this is one way of identifying a conflict of interest scenario).

We now show the use of the SULTAN specification and analysis in a small case study.

## 5. CASE STUDY

Bob's Music Warehouse (BMW) (Viega et al, 2001) is an Internet music sales site. It consists of: a web browser, a client application, a front-end server, a content database and a credit card server. The web browser and client application are run from the users' computer, while the other components are run and maintained by Bob. A user uses the web browser to access the front-end server and buy music. The browser communicates with the front-end server using cryptography. The client application is used to play the user's purchased titles. The content database contains all the titles that can be bought at Bob's site, and this database can be linked with third-party databases. To prevent illegal replication of purchased music, the purchased titles remain encrypted on the users' computers and only the client application can decrypt and play the purchased titles. Also, in order to prevent a user from giving encrypted titles to a friend with a copy of the client application, the titles are cryptographically bound to the user. Bob has established strategic business alliances with Pete's Music Warehouse (PMW) and with ProvE, a provider of music titles. The client applications from BMW and PMW are able to interact, and Bob uses the content database from ProvE to augment his product base.

For simplicity, we will only focus on BMW's trust assumptions, which are:

i) BMW only trusts the client application to decrypt songs.
ii) The front-end server trusts the client application to play purchased titles.
iii) The front-end server trusts the web browser to access the music database and buy music
iv) BMW trusts PMW's client application to access its music database.
v) The front-end server trusts the credit card server to verify and store credit card information.
vi) The front-end server trusts the content database to encrypt and to provide titles.
vii) The front-end server trusts the ProvE content database to provide titles.

For convenience, we will use the abstractions shown in table 1.

| Symbol | Meaning |
|---|---|
| BMW | Bob's Music Warehouse |
| ClientApp | BMW's client application |
| FrontEnd | BMW's front-end server |
| WebBrowser | The client's web browser |
| PMWClientApp | PMW's client application |
| CreditServer | BMW's Credit Card Server |
| ContentBase | BMW's content database |
| ProvE | ProvE's content database |
| decrypt(Entity, Title, Decrypted) | decrypts Title to produce Decrypted |
| encrypt(Entity, Title, Encrypted) | encrypts Title resulting in Encrypted |
| play(Entity, Title) | plays Title |
| AccessMusic(Entity) | accesses music database on Entity |
| BuyMusic(Entity, Title) | allows one to purchase Title |
| VerifyCreditInfo(Entity, CreditDetails) | verifies CreditDetails |
| StoreCreditInfo(Entity, CreditDetails) | stores CreditDetails in secure form |
| ProvideMusic(Entity, Titles) | retrieves music titles. |

*Table 1.* Abstractions for BMW

We assume that the system administrator has specified the following in the SULTAN specification notation for BMW:

i1: **trust**(BMW, ClientApp, decrypt(ClientApp, TitleName, Decrypted), 100);

i2: **trust**(BMW, _Y, decrypt(_Y, TitleName, Decrypted), -100);

ii : **trust**(FrontEnd, ClientApp, play(ClientApp, TitleName), 100) ← decrypt(ClientApp, TitleName, DecryptedFile);

iii : **trust**(FrontEnd, WebBrowser, AccessMusic(ContentBase): BuyMusic(FrontEnd, Title), 100);

iv : **trust**(BMW, PMWClientApp, AccessMusic(ContentBase), 100);

v : **trust**(FrontEnd, CreditServer, VerifyCreditInfo(CreditServer, CreditDetails): StoreCreditInfo(CreditServer, CreditDetails), 100);

vi : **trust**(FrontEnd, ContentBase, encrypt(ContentBase, Title, EncryptedFile): ProvideMusic(ContentBase, Title), 100);

vii : **trust**(FrontEnd, ProvE, ProvideMusic(ContentBase, NewTitles), 100);

All specifications are translated into Prolog for analysis. For this study, we will only look at source analysis. To determine if there are any positive trust policies that have webbrowser as their trustee, we write:

**query**(X, (**p_pos_trust**(X), **p_target**(webbrowser, X)), Answer).

If BMW decides that no entity should be trusted to both verify and store credit card information (as this would be a conflict of interest), then the following query would be used:

**query**(X, (**p_actions**([verifycreditinfo(_,_)], P), **p_actions**([storecreditinfo(_,_)], Q), **p_target**(X,P), **p_target**(X, Q)), Answer).

To determine if there is a trust and distrust policy in our specification that relate to the same entities and the same action, we specify:

**query**([X , Y], (**p_pos_trust**(X), **p_neg_trust**(Y), **p_subject**(A, X), **p_subject**(A,Y), **p_target**(B,X), **p_target**(B,X), **p_actionset**(Act, X), **p_actionset**(Act, Y)), Answer).

The queries are meant to be simple and demonstrative, in order to keep the discussion high-level. More involved and complex queries can be formulated to meet BMW's requirements. We now outline how the SULTAN view of trust can be used in the field of Internet Commerce.

# 6. USING A TRUST MANAGEMENT FRAMEWORK

The Trust Management Framework will be used both as a decision support tool to aid human managers or automated manager agents and to support on-line trust queries for policy decisions relating to access control or which security mechanisms to use. We use the case study to show how the SULTAN Framework can be used.

## 6.1 The Negotiation Process

The negotiation process between a client and a producer determines whether or not a business relationship should be set up. This process governs the exchange of credentials (Winsboroough, Seamons, Jones, 1999), bargaining over price and possibly quality of service. BMW might be approached by another content provider MusicStore. Initially, MusicStore would only has a low default trust level, so BMW would take an overly cautious approach. However, as part of the credential exchange, MusicStore provides a signed recommendation from PMW. This recommendation is inserted into BMWs trust spec and now there is a match with a trust spec based on a PMW recommendation, so the trust level of MusicStore is now medium.

## 6.2 The Contract Evaluation Process

Eventually BMW has 2 potential contracts to choose from but decides it really only needs one more content supplier. When faced with a choice between a group of potential E-commerce partners, the SULTAN trust framework can be used to help in selecting the appropriate partner. AllMP3 has revealed that part of its content is outsourced from DodgyStores and BMW has experience information about DodgyStores.

The manager of BMW queries the information store and specification server to give all recommendations, trust rules, experience and risk information relating to an entity and then inserts a new trust rule which effectively gives greater weight to bank recommendations than client recommendations.

## 6.3 The Recommendation Formation Process

It is possible to explicitly specify recommendations in the SULTAN notation. However, humans may want to generate recommendations about an entity based on current trust rating for that entity either within the same context or even with respect to a different context i.e. there are no existing trust rules for that specific context. For example, BMW gets a request for a recommendation of PMW as video content supplier but only has a trust rule related to music supply. BMW is not quite sure about PMW's ability to provide the data rate needed for video so only gives a low rating, which is inserted in the specification server.

## 6.4 Infrastructural Security

Trust-based decision-making can be used to configure the security of infrastructure. BMW sets up a contract with a local radio station to give unlimited access to all its music sources for a fixed monthly charge. A VPN connection is used to link the Radio station network and BMW, so the encryption can be disabled on sending music to the Radio station as the VPN provides a secure channel.

## 6.5 Access Control Decisions

The trust level of a trustee can be the basis of an authorisation decision for access to a trustor's resources or services by a trustee. A new customer of BMW is covered by the following rule, which allocates a default low trust level of 10.

```
DefCust: trust (BMW, _newCustomer, AccessMusic(ContentDatabase), 10);
```

A couple of Ponder authorisation policies can be used to give access to all music if the customer trust level is high but only to a subset if the trust level is low. A Ponder authorisation policy specifies access control for security. Positive authorisation policies specify the actions that a subject is permitted to perform on a target object, while negative authorisation policies specify the actions that a subject is forbidden from performing on the target object.

```
type auth+ Access ( domain SegmentofContentBase, string TrustValue){
   subject Client; target SegmentofContentBase; action AccessMusic();
   when trust+(FrontEnd, ClientApp,  AccessMusic(ContentDatabase), TrustValue )
   };
inst auth+ AccessHigh = Access(/BMW/ContentBase, HighTrust);
inst auth+ AccessLow = Access(/BMW/ContentBase/Restricted, LowTrust);
```

This defines a Ponder policy type called Access with 2 parameters – the segment of the music to which access is to be permitted and a trust value. the constraint to the policy is in the form of a query to the SULTAN framework to determine whether the subject is trusted at the required level. Two policy instances are then created to cater for the high and low trust situations.

This shows that it is possible to use SULTAN queries as Ponder constraints to help in the access control decision.

## 6.6 Resource Allocation Process

BMW performs an audit of its records relating to customer use every six months. Based on the experience information on the customer and the trust level (and the changes in the trust level) of the customer, BMW decides whether to upgrade the customer's status and give the customer further discounts on all purchases.

This completes our discussion on the application of the trust management framework. We now focus on related work.

# 7. RELATED WORK

The concept of the trust specification and analysis aspects of the SULTAN framework is derived from related work on logic-based formalisms of trust and on trust management.

All the logic-based frameworks that have attempted to deal with the issue of trust (namely, Joesang's subjective logic (Joesang, 1997, Joesang, 1998, Joesang, 1996) Jones and Firozabadi's model (Jones, Firozabadi), Rangan's model (Rangan, 1988), etc) suffer from one or more of the three basic problems: 1) their underlying assumptions make them non-applicable to the distributed framework, 2) there is no associated tool support, or 3) they address a subsection of the trust management problem. SULTAN addresses a large subset of the trust problem, makes no assumptions that may render it unusable and comes with software support.

The dominant view of trust management has not changed considerably since 1996. The term was defined in the context of public key cryptography. A trust problem was identified due to the deployment of large-scale public key infrastructures (PKIs). The solutions developed using this view of trust management (namely, AT&T's PolicyMaker and KeyNote, REFEREE, IBM Role-based Access Control Model IBM, Trustbuilder (Winsborough et al, 1999) and Poblano (Chen, Yeager, 2000)) have the following problems: 1) the responsibility of checking the conditions of the establishment of a trust relationship is entirely the application developer's concern, 2) the relationships are assumed to be monotonic 3) these solutions do not learn from the information available to them and 4) the emphasis is on access control decisions rather than general analysis of trust, whereas SULTAN (with Ponder) can cater for both.

## 8. FUTURE WORK & CONCLUSIONS

Current work on the SULTAN framework is focused on implementing the necessary tools for specification and analysis. The main thrust of SULTAN is not to offer a concrete, exhaustive, logic theoretic framework for trust. The primary purpose is to identify and analyse the effects of changing specifications on a business and to utilize these specifications to augment the security of Internet commerce. By using symbolic names for trustees and trustors, we ensure that identity of entities is a non-issue. Unknown transaction partners can be referred to by some arbitrary mnemonic. SULTAN allows for the separation of trust management code from application code to support scalable Internet based applications.

Current solutions focus on authentication or access control, leave the responsibility of checking the conditions of the establishment of trust relationship entirely up to the application developer; they assume that relationships are monotonic and they do not learn from experience to dynamically adjust trust levels.

The case study demonstrates that systems can be specified and analysed in SULTAN quite easily. This document outlined the basics of the framework and did not delve into the intricacies of the system.

## 9. ACKNOWLEDGEMENTS

We gratefully acknowledge our colleagues in the Policy Group at Imperial College who provided comments on this work, namely, Naranker Dulay, Emil Lupu, Alessandra Russo and Marek Sergot. We also acknowledge funding of a studentship from Microsoft, British Telecom and the Bank of Nova Scotia.

## 10. REFERENCES

Swarup and C. Schmidt, *"Interoperating between Security Domains"*, European Conference on Object-Oriented Programming (ECOOP) Workshop on Distributed Object Security, Brussels, Belgium, 1998, Springer LNCS, pp. 283.

T.Grandison and M. Sloman, *"A Survey of Trust in Internet Applications"*, IEEE Communications Surveys and Tutorials, Vol. 4, No. 4, pp. 2-16, 2000.

N. Damianou, N. Dulay, E. Lupu, and M. Sloman, *"The Ponder Specification Language"*, Workshop on Policies for Distributed Systems and Networks, HP Labs, Bristol, 2001, Springer LNCS, No. 1995, pp. 18-37.

A.Joesang, *"Trust Management for E-commerce"*, 2000, *http://citeseer.nj.nec.com/375908.html.*

J. Viega, T. Kohno, and B. Potter, *"Trust (and Mistrust) in Secure Applications"*, Communications of the ACM, Vol. 44, No. 2, Feb. 2001, pp. 31-36.

W. Winsborough, K. Seamons, and V. Jones, *"Automated Trust Negotiation: Managing Disclosure of Sensitive Credentials"*, IBM Transarc Research White Paper, 1999.

A. Joesang, *"Artificial Reasoning with Subjective Logic"*, Proc. 2nd Australian Workshop on Commonsense Reasoning, Perth, Australia, 1997.

A. Joesang, *"Prospectives for Modelling Trust in Information Security"*, Australasian Conference on Information Security and Privacy, Sydney, NSW, Australia, 1997, Springer LNCS, No. 1270, pp. 2-13.

A. Joesang, *"A Subjective Metric of Authentication"*, 5th European Symposium on Research in Computer Security (ESORICS'98), Louvain-la-Neuve, Belgium, 1998, Springer LNCS, No. 1485, pp. 329-344.

A. Joesang, *"The right type of trust for distributed systems"*, ACM New Security Paradigms Workshop, Lake Arrowhead, California, 1996, pp. 119- 131.

A. Jones, J. I. and B. S. Firozabadi, *"On the Characterisation of a Trusting Agent - Aspects of a Formal Approach"*, in Trust and Deception in Virtual Societies, Cristiano Castelfranchi et al., eds., Kluwer, Holland, pp. 163-174

P.V. Rangan, *"An Axiomatic Basis of Trust in Distributed Systems"*, Proc. IEEE Symposium on Research in Security and Privacy, Washington, DC, 1988, IEEE, pp. 204-211.

M. Blaze, J. Feigenbaum, and J. Lacy, *"Decentralized Trust Management"*, IEEE Symposium on Security and Privacy, Oakland, California, USA, 1996, pp. 164-173. *http://www.crypto.com/papers/policymaker.pdf.*

Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, *"REFEREE: Trust Management for Web Applications"*, 1997, *http://www.farcaster.com/papers/www6-referee/.*

Y.-H. Chu, *"Trust Management for the World Wide Web"*, MEng Thesis, Massachusetts Institute of Technology, 1997, *http://www.w3.org/1997/Theses/YanghuaChu/.*

IBM, *"IBM Trust Establishment Policy Language"*, *http://www.haifa.il.ibm.com/projects/software/e-Business/TrustManager/index.html.*

R. Chen and W. Yeager, *"Poblano: A Distributed Trust Model for Peer-to-Peer Networks"*, Sun Microsystems Technical Paper, 2000, *http://www.sun.com/software/jxta/poblano.pdf*