Technical Report

UCAM-CL-TR-643
ISSN 1476-2986

Number 643

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Using trust and risk for access control in Global Computing

Nathan E. Dimmock

August 2005

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted April 2005 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Jesus College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/TechReports/

ISSN 1476-2986

Abstract

Global Computing is a vision of a massively networked infrastructure supporting a large population of diverse but cooperating entities. Similar to *ubiquitous computing*, entities of global computing will operate in environments that are dynamic and unpredictable, requiring them to be capable of dealing with unexpected interactions and previously unknown principals using an unreliable infrastructure.

These properties will pose new security challenges that are not adequately addressed by existing security models and mechanisms. Traditionally privileges are statically encoded as security policy, and while rôle-based access control introduces a layer of abstraction between privilege and identity, rôles, privileges and context must still be known in advance of any interaction taking place.

Human society has developed the mechanism of **trust** to overcome initial suspicion and gradually evolve privileges. Trust successfully enables collaboration amongst human agents — a computational model of trust ought to be able to enable the same in computational agents. Existing research in this area has concentrated on developing trust management systems that permit the encoding of, and reasoning about, trust beliefs, but the relationship between these and privilege is still hard coded. These systems also omit any explicit reasoning about risk, and its relationship to privilege, nor do they permit the automated evolution of trust over time.

This thesis examines the relationship between trust, risk and privilege in an access control system. An outcome-based approach is taken to risk modelling, using explicit costs and benefits to model the relationship between risk and privilege. This is used to develop a novel model of access control — trust-based access control (TBAC) — firstly for the limited domain of collaboration between Personal Digital Assistants (PDAs), and later for more general global computing applications using the SECURE computational trust framework.

This general access control model is also used to extend an existing rôle-based access control system to explicitly reason about trust and risk. A further refinement is the incorporation of the economic theory of decision-making under

uncertainty by expressing costs and benefits as utility, or preference-scaling, functions. It is then shown how Bayesian trust models can be used in the SECURE framework, and how these models enable a better abstraction to be obtained in the access control policy. It is also shown how the access control model can be used to take such decisions as whether the cost of seeking more information about a principal is justified by the risk associated with granting the privilege, and to determine whether a principal should respond to such requests upon receipt. The use of game theory to help in the construction of policies is also briefly considered.

Global computing has many applications, all of which require access control to prevent abuse by malicious principals. This thesis develops three in detail: an information sharing service for PDAs, an identity-based spam detector and a peer-to-peer collaborative spam detection network. Given the emerging nature of computational trust systems, in order to evaluate the effectiveness of the TBAC model, it was first necessary to develop an evaluation methodology. This takes the approach of a threat-based analysis, considering possible attacks at the component and system level, to ensure that components are correctly integrated, and system-level assumptions made by individual components are valid. Applying the methodology to the implementation of the TBAC model demonstrates its effectiveness in the scenarios chosen, with good promise for further, untested, scenarios.

Acknowledgements

Foremost I would like to thank my supervisor, Dr. Jean Bacon, for not only giving me this most wonderful opportunity to study for a PhD, but also for her continuing guidance, support, and encouragement throughout my time as both graduate and undergraduate student in Cambridge. My gratitude must also be extended to Dr. Ken Moody (and his excellent cellar), Dr. David Ingram and Dr. Brian Shand for their advice, collaboration and proof-reading.

Further thanks are due to the "Opera Group Tea-Time Regulars", particularly Peter Pietzuch, Alan Abrahams, Brian Shand, András Belokosztolszki, Dave Eyers, David Ingram and Eiko Yoneki, for so many fascinating and informative discussions. I am also grateful to my colleagues on the SECURE project — especially those here in Cambridge: Brian Shand, David Ingram, Andy Twigg, and Daniel Cvrcek — for the many inspiring and productive meetings which drove my own research forward.

I have been very lucky to be able to travel as widely as I have during the course of my PhD, and therefore I thank the EU for funding this, as well as the EPSRC and Marconi plc for funding my PhD. I should also like to thank those who accompanied me on my travels for making them all the more enjoyable: Brian, Olivia, Dave, Peter, Jenny and Rosie.

Thanks also to Jesus College, and particularly its Graduate Society, Hockey club and Ultimate Frisbee team, for providing with me with such an extensive extra-curricular timetable. Life in Cambridge would also not have been anything like as enjoyable or interesting without the members of the JCN and JDS — thank you, all.

Finally, a huge "thank you" to everyone who proof-read parts of this dissertation: Dave Eyers, Brian Shand, Lauri Pesonen, Jamie McMahon, Stephen Rymill, Elizabeth Scott, Ross Church, Rosie Whitaker, Sean Furey, and Mum and Dad.

List of Publications

- Brian Shand, Nathan Dimmock, and Jean Bacon. Trust for transparent, ubiquitous collaboration. In *First IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 153–160, Dallas-Ft. Worth, TX, USA, March 2003.

- Nathan Dimmock. How much is 'enough'? Risk in trust-based access control. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises — Enterprise Security*, pages 281–282, June 2003.

- Andrew Twigg and Nathan Dimmock. Attack-resistance of computational trust models. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 275–280, June 2003.

- Vinny Cahill, Brian Shand, Elizabeth Gray, Ciarán Bryce, Nathan Dimmock, Andrew Twigg, Jean Bacon, Colin English, Waleed Wagealla, Sotirios Terzis, Paddy Nicon, Giovanna di Marzo Serugendo, Jean-Marc Seigneur, Marco Carbone, Karl Krukow, Christian Jensen, Yong Chen, and Mogens Nielsen. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3):52–61, August 2003.

- Nathan Dimmock, András Belokosztolszki, David Eyers, Jean Bacon, and Ken Moody. Using trust and risk in role-based access control policies. In *SACMAT'04: Proceedings of the ninth ACM Symposium on Access Control Models and Technologies*, pages 156–162. ACM, June 2004.

- Jean-Marc Seigneur, Nathan Dimmock, Ciarán Bryce, and Christian Damsgaard Jensen. Combating Spam with TEA, Trustworthy Email Addresses. In *Proceedings of the Second Annual Conference on Privacy, Security and Trust (PST'04)*, pages 47–58, Fredericton, New Brunswick, Canada, October 2004.

- Nathan Dimmock and Ian Maddison. Peer-to-peer collaborative spam detection. *ACM Crossroads*, 11(2):17–25, Winter 2004.

- Brian Shand, Nathan Dimmock and Jean Bacon. Trust for ubiquitous, transparent collaboration. *Wireless Networks*, 10(6):711–721, Kluwer, 2004.[1]

- Ciarán Bryce, Nathan Dimmock, Karl Krukow, Jean-Marc Seigneur, Vinny Cahill, and Waleed Wagella. Towards an evaluation methodology for computational trust systems. In *Proceedings of the Third Annual Conference on Trust Management*, volume 3477 of *LNCS*, pages 290–305. Springer-Verlag, May 2005.

- Nathan Dimmock, Jean Bacon, David Ingram, and Ken Moody. Risk models for trust-based access control (TBAC). In *Proceedings of the Third Annual Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 1–8. Springer-Verlag, May 2005.

[1]This is a revised and extended edition of the paper, "Trust for transparent, ubiquitous collaboration", published at the *First IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003)*.

Contents

CHAPTER 1

Introduction

As the mobility and communication properties of computing devices increase, users will expect these devices to be able to securely obtain services and share information in a variety of environments. This dissertation shows how such secure collaboration can be enabled and facilitated by using risk to mediate the relationship between trust beliefs and privilege.

The European Union Information Society Technologies Programme (EU-IST) *Global Computing* Initiative foresees a massively networked infrastructure supporting a large population of diverse but cooperating entities. The composition and characteristics of this infrastructure will be dynamic and unpredictable: entities will be both autonomous and mobile, and will have to be capable of dealing with unforeseen circumstances ranging from unexpected interactions with other entities to disconnected operation.

This vision of a massive population of cooperating networked entities is very similar to Mark Weiser's vision of *ubiquitous computing* (ubicomp) [Wei91]. The central idea of ubicomp is that computing technology is so completely integrated into everyday lives that no one even notices its presence. One pre-requisite for this is the ability for entities to continue to operate when encountering new circumstances since failure would remind the human of the computer's presence.

There are also similarities to the concept of *Grid Computing* [App02]. Since Grid computing began as a middleware for sharing resources between well connected computers, initially a lack of mobility distinguished it from ubiquitous computing. However, as Grid computing has evolved, researchers have begun to find synergy with the ubicomp vision and research is now being undertaken into ubiquitous Grid computing [SBD+03]. This new vision of leveraging Grid technology to deliver the ubicomp vision has much in common with global computing.

The properties of global computing present new security challenges that are not adequately addressed by existing security models and mechanisms. The

size of the global computing infrastructure means that security policy must encompass billions of potential collaborators and attackers. Mobility and the potential for disconnection from one's home network requires the ability to make security decisions autonomously in an environment where identity conveys no *a priori* information about the likely behaviour of the principal, precluding the use of many existing access control systems.

These challenges are not unlike those faced by humans when confronted with unexpected interactions, or interactions with unknown persons in foreign or unfamiliar territory. Human society has developed the mechanism of **trust** to overcome initial suspicion and gradually evolve privileges. Trust successfully enables collaboration among human agents, a computational model of trust ought to be able to enable the same in computational agents.

For example, a mobile entity in an unfamiliar and potentially hostile environment will need to obtain services such as network connectivity, printing, use of external displays, and additional computational resources. Alternatively, perhaps after striking up a conversation with fellow commuter Bob, Alice may wish to swap mobile phone ring tones, with him without giving access to her address book. Another scenario might be that having arrived in a new town, Alice wishes to broadcast a request for restaurant recommendations to her fellow train travellers. The responses she receives will undoubtedly be conflicting, but she may find one or two from people she is indirectly acquainted with, or has had dealings with before that would allow her to make a more informed decision.

Such cooperation requires trust between participants. Traditionally trust may be obtained by deferring the security decision to a mutually trusted third-party. Recently, decentralised *trust management systems* have permitted entities to use tokens called certificates to assert their trustworthiness or authorisation credentials even when disconnected, but these still require an existing per-context trust infrastructure that is not guaranteed to exist in global computing.

1.1 The SECURE Project

The EU-IST funded SECURE (Secure Environments for Collaboration among Ubiquitous Roaming Entities) project, was a three year research project (January 2002 – December 2004) that aimed to produce a novel approach to security to meet the challenges outlined above. Five institutions were involved in SECURE: the University of Cambridge, Trinity College Dublin, Basic Research in Computer Science (BRICS) at the University of Aarhus, the University of Geneva, and the University of Strathclyde. The University of Cambridge led the work on *Security Policy*, including the risk and access control aspects of the project, and it is in this area that this thesis makes its contribution.

SECURE Applications

Since SECURE aims to be a general security model for the global computing infrastructure, initial work consisted of analysing a number of possible appli-

cations that the GC infrastructure would enable and capturing their security requirements.

Dynamic Source Routing (DSR) Source routing in mobile ad hoc networks was first described in [JM96]. A node wishes to find other nodes in the network it can trust to correctly route its packets. Since forwarding a packet consumes resources, when accepting a packet to forward on someone else's behalf, it must also trust that the node for which it is routing will reciprocate.

Distributed Gaming [GOJ+02] describes a blackjack gaming system for interacting Personal Digital Assistants (PDAs). Trust can be used to determine whether a player should be permitted to join a game: players who earn a reputation for cheating, not paying when they lose, or perhaps just being too good, would not be permitted to join. If the game protocol allows the dealer additional opportunities for cheating then a distributed trust computation could also be used to determine which player is to be appointed the dealer.

e-purse The e-purse scenario was suggested by Ciarán Bryce of the University of Geneva. It aims to solve the problem of automating small value payment transactions such as paying for newspapers and bus tickets. The e-purse owner must trust that the vendor will charge the correct amount, whilst the vendor must trust that the e-purse user is using *bona fide* e-cash in their e-purse so they will actually receive their payment.

Academic Visitors When a colleague from another SECURE partner institution visits the Cambridge Computer Laboratory giving them access to computing resources such as printing or the Internet is a laborious process. Clearly there is scope for abuse if authorisation is given too easily, but if it were possible to monitor usage and revoke access if there was an indication that they were not trustworthy then the process could be streamlined. Stored experiences could be used to facilitate the enabling of access (or increase restrictions if the visitor proved untrustworthy!) on subsequent visits. Further, trust information could be shared between resource managers to facilitate access to (or prevent abuse of) resources in other Departments or Colleges that the visitor might have cause to visit.

PDA Collaboration The increasing communication capabilities of mobile devices, including PDAs, means there is a desire to facilitate a free flow of information, such as new telephone numbers, photographs, ring tones and games, whilst securing these devices from the viruses and other security problems that plague the average personal computer. Users do not want to be mindful of security, but rather they want it to "just work" when they wish to exchange contact details with everyone in the room, or collaboratively schedule an appointment without revealing the details of their diary.

Distributed Backups/Grid As the world moves into a digital age, storage, and particularly redundant storage, is of increasing importance. Distributed storage systems can also be used for publication, for instance to take advantage of alternative legal jurisdictions, or even just to cache files nearer to the consumers. However, as the saying goes: "On the Internet, nobody knows you're a dog." so users must trust the integrity and reliability of the providers they choose. Similarly, providers must trust that users will honour the terms of the contract, not consume excessive resources, nor do anything illegal.

Collaborative Authorship (Wikis) Slashdot.org has for a long time had a system that allows registered users to rate submissions from other users. Wikipedia.org, a free online encyclopedia, goes further and allows users to edit existing articles as well as add new ones. Readers must trust that contributors are providing accurate information. Similarly to the *Academic Visitors* scenario, trust-based assessment could also be used to streamline the quality assurance process for contributors with established track records, or verified expertise in the relevant area, whilst increasing the level of verification required for less trustworthy or unknown contributors.

Traffic Congestion Reports This application scenario was discussed at the SECURE kick-off meeting and inspired by Trinity College, Dublin's work on traffic monitoring in Dublin. Vehicles would publish information on the congestion status of their current position that other near-by vehicles could then use to optimise their route and avoid congestion. Users must trust that the information supplied is accurate as malicious users might propagate false information in an attempt to divert traffic away from routes they wish to use by indicating that they are heavily congested.

Spam Filtering Email is a good example an application of where the free flow of information is very useful and enables collaborations that were not previously possible. Spam is a good example of what happens when the mechanisms are designed using the assumptions that other principals are, for the most part, trustworthy. Users of email systems must trust that the email they receive is useful, interesting, and relevant to them, and is not unsolicited commercial material or viruses.

Recommendation Systems/Distributed "Sticky" Notes Similar to the *Collaborative Authorship* systems, there exist a number of websites, such as Amazon.com, that will make recommendations about books, music, and films based on previous purchases or listening habits. Another possibility might be the posting of "virtual sticky notes" in an Active City, for instance recommending restaurants or providing information to tourists about sites of interest. With no restrictions on who can post a note, information overload and clutter will require strict filtering of notes, based on recommendations from other inhabitants of the city.

1.2 Research Issues

In summary, the aim of the SECURE project is a general security model for the global computing infrastructure, in which computational entities interact on a basis of mutual trust. This aim shall now be refined and the elements relevant to this thesis highlighted.

Trust is an elusive concept with many competing definitions found in the literature (see chapter 2). While the development of a formal computational model of trust is the domain of other members of the SECURE consortium, a trust-based access control mechanism and security policy must integrate tightly with the trust model in order to be effective.

Intuitively, there is no reason to require trust unless there is something at **risk**. It has long been recognised that the concepts of trust and risk are closely related, and the nature of this relationship will need to be examined. Accordingly, if trust is to become the basis for access control decisions then trust, risk, and privilege must be associated within any computational model.

Once a principal has established trust and risk parameters for their situation, **security policy** will determine how they proceed with any interaction. Whilst the goal of **access control** is to restrict access to resources and protect their integrity, global computing will only succeed if new relationships can be established and collaborations take place in environments where a principal was previously unknown. The specification and enforcement of security policies based on trust, risk, and privilege — Trust-Based Access Control (TBAC) — will therefore form another major component of this thesis. An important element of this component will be the consideration of previously unknown principals.

User privacy is also an issue for trust-based systems. The effect of sharing trust information too publicly must be considered from the point of view of users' privacy, as well as how that might impact on the effectiveness of the security policy. Requesting or publishing trust information too publicly could allow attackers to infer information about security policy, or the identities of a principal's trusted peers that could perhaps be used to aid their attack. Moreover, security is a **trade-off**: in ubicomp environments resources such as energy for communication and processing are scarce; in any environment there may exist reputation brokers that charge a small fee for their services. It is therefore important for the access control manager to be able to consider the potential cost of the trust calculation when determining access rights — for low value resources the cost of the trust computation could exceed the value of the resource being protected.

1.3 Thesis Contribution

The primary contribution of this thesis is in the specification and prototype implementation of a trust- and risk-based access control system within the SECURE framework. This is realised through a number of research contributions:

- The development of a novel access control model for information sharing

between personal digital assistants using trust and risk.

- The analysis of risk and the development of a formal model of risk for global computing.
- The extension of traditional rôle-based access control models to include reasoning about trust and risk.
- A generalised model of trust- (and risk-) based access control (TBAC) for global computing, that can be used in conjunction with a number of trust models, not just SECURE.
- An architecture for trust- and risk-based decision-making based on the SECURE framework including the risk of seeking recommendations, access control to the evidence store, and the interaction between trust and risk.
- An evaluation methodology for computational trust systems, such as SECURE.
- The implementation of the trust, risk and access control components of the SECURE architecture and an anti-spam application in which TBAC, risk and other aspects of SECURE architecture are validated.

1.4 Thesis Overview

The following chapter reviews the relevant literature on trust management, security policy and access control and self-organising distributed systems, such as those likely to be found in global computing. The aspects of the SECURE framework produced by people other than the author are also detailed here.

Chapter 3 develops a novel access control model for information sharing between PDAs using trust and risk. This work serves as a more detailed analysis of the security requirements of one of the possible application scenarios outlined above.

The subsequent chapter analyses the concept of risk, the relationship between trust, risk and privilege and develops a model of risk for securing the global computing infrastructure. Chapter 5 examines the specification and enforcement of trust- and risk-based security policies. This is done by extending an existing rôle-based access control model and subsequently generalising it.

In light of the experiences of using and evaluating the models described in chapters 4 and 5, and in response to revisions to other parts of the SECURE framework, it was felt necessary to revise the initial model. This new model is presented in chapter 6. Chapter 6 also demonstrates how the new model can integrate with trust models other than the SECURE one, and makes contributions in the areas of the risk of requesting trust information from other principals and controlling access to a principal's evidence/trust information store.

Chapter 7 describes an implementation and validation of the second generation SECURE software architecture. As an emerging field, there was not yet

any standard evaluation methodology that could be used to validate the framework, and therefore it was also necessary to develop one. The methodology developed is described in this chapter, along with two different spam detection applications that were used to validate the aspects of the SECURE framework covered in this thesis.

The final chapter summarises the key results and conclusions of this thesis and outlines potential future work.

CHAPTER 2

Background

This chapter describes the research on which this thesis depends. It begins by surveying access control models for distributed systems, and the development of trust management systems. Next, techniques for managing identity in distributed systems are reviewed, followed by a survey of computational, evidence-based trust systems that extend trust management systems to formally model trust and permit trust assessment to evolve over time. Finally, the SECURE framework is described.

2.1 Access Control in Distributed Systems

[Eye04] defines the term *Access Control* as:

> Access Control involves preventing unauthorised users from interacting with particular resources in certain ways, whilst guaranteeing that authorised users will not be denied their access rights.

Access control models are generally divided into Mandatory (MAC) and Discretionary (DAC). Models of the former type control access to resources via system-wide policy. They tend to be very inflexible, and enforcing system-wide policy is difficult in a widely-distributed system, if such a concept is even meaningful. DAC is more flexible: users may determine the policy used to control access to their own objects or resources.

Early DAC models, such as Access Control Lists (ACLs) and Capabilities, defined policy as mappings from principal identities to privileges. However, these do not scale to the large numbers of principals or resources found in a distributed system as ACLs become inefficiently long with a large number of principals, and capabilities become difficult to manage and administer. Rôle-Based Access Control (RBAC) is a form of DAC that introduces a level of indirection:

principals are assigned to rôles, and rôles are assigned privileges [FSG+01], thus allowing greater scalability.

2.1.1 The Open Architecture for Secure Interworking Services

The *Open Architecture for Secure Interworking Services (OASIS)*[1] [BMY01, BMY02] is a rôle-based access control implementation with a formal, logic-based model and a Horn-clause style policy language.

OASIS has a number of novel features that set it apart from other RBAC systems:

Dynamic Rôle Activation: Rôle certificates are short-lived; that is, they must be *activated* within the context of a *session*, and are revoked when the session concludes. Since a principal need only activate the rôle(s) they need to complete the goal of the current session, the *principle of least privilege* [SS75] is maintained.

Appointments: Persistent credentials, such as professional qualifications and group membership, are supported through the use of appointment certificates. Appointments also permit the delegation of privilege — the delegator may issue an appointment certificate to the delegatee which can be used in the appropriate rôle activation.

Parameterisation: All elements of the OASIS policy language can be parameterised, thus allowing request-dependent information to be included in the access control decisions. These parameters are strongly typed.

Context Dependent Evaluation: Information about the current environment, or *context*, may be assigned to rôle and privilege parameters. OASIS allows environmental conditions to be evaluated through environmental predicates. These predicates may provide two-way communication to components outside the policy-enforcing environment.

Fast Revocation: OASIS is built upon an event architecture that allows rôle assignments to be rapidly and automatically revoked if a precondition ceases to be true.

As noted above, the OASIS policy language uses a Horn-clause form, and there are two forms of rule [BMY02]:

Authorisation rules map rôles to privileges, and follow the structure:

$$r, e_1, ..., e_{n_e} \vdash p$$

According to such rules, the r rôle is assigned the p privilege if all of the n_e number of e_k environmental predicates are satisfied.

[1] The namespace collision with the "OASIS Open" consortium (Organization for the Advancement of Structured Information Standards) is unfortunate. In this thesis, OASIS shall refer to the RBAC system, except where explicitly stated otherwise.

Activation rules have the form:

$$r_1, r_2, ..., r_{n_r}, ac_1, ..., ac_{n_{ac}}, e_1, ..., e_{n_e} \vdash r$$

They may contain any number of prerequisite rôles r_i, appointment certificates ac_j, and environmental predicates e_k. If a principal holds all the prerequisite rôles and appointment certificates prescribed in such an activation rule, and all of the environmental predicates are satisfied, they may enter the target rôle r.

In activation rules certain prerequisites can be marked as membership conditions. Such prerequisites must remain true for the principal to remain active in the rôle, as opposed to only needing to be true at rôle activation time. If the membership condition becomes false (the principal is no longer active in the marked prerequisite rôles, does not hold the marked appointment certificates, or the environmental predicates become false) the target rôle is automatically revoked from the principal.

2.1.2 Access Control Policy

Policy is an amorphous term. [DDLS01] defines a policy as "a rule that defines a choice in the behaviour of a system." In general it is used to refer to behavioural configuration that is independent of the system's implementation, thus allowing the behaviour of the system to be modified without changing the implementation. In this thesis *policy* is used to refer specifically to components of a general framework that determine the behaviour of a particular user or node. This is distinct from behaviour programmed into an application-specific instantiation of the framework. Thus, *access control policy* is the tool by which a resource manager determines which principals should have access to what resources under his or her control.

Policy as a mechanism for dynamically controlling the behaviour of a system is becoming increasingly popular, and there is therefore a plethora of domain-specific languages for specifying policy. Recent research has focused on two areas:

- the development of "general" policy languages that are applicable in multiple domains, thereby allowing re-use of existing tools and knowledge. For example, [BS04a, DDLS01].
- the development of XML-based standardised domain-specific languages, such as [OAS03, Mos04].

Access Control "Aware" Policy Languages

OASIS access control policy consists of a set of Horn-clause style rules of the form described in the previous section.

Ponder is a general, object-oriented, declarative policy language for controlling policy-based networking equipment [DDLS01]. It supports a wide

variety of types of policy, including *authorisation*, *information filtering*, *delegation* and *obligation* as well as a language for specifying the set of conditions under which a policy is valid ("constraints"). Positive and negative policies are permitted which, whilst intuitive, does make enforcement more difficult and can lead to conflicts that must be resolved using static analysis.

Cassandra is a very general policy language with formal semantics that enable efficient evaluation and guaranteed termination of rules [BS04a]. Its expressiveness can be tuned by choosing an appropriate *constraint domain*, and thus instead of using special constructs to express standard policy idioms such as rôles, delegation, and trust management (see section 2.2), it is sufficient to choose a suitable constraint domain. Cassandra permits only authorisation policies, not obligations.

Law Governed Interaction (LGI) is a message exchange mechanism that permits a group of agents to collaborate in a manner *governed* by an explicitly specified policy, called the *law* of that group [AM04]. Similarly to Cassandra, Ao and Minsky propose, in LGI, a general (access control) policy system in which common constructs such as rôles may be defined. They refer to this as *rôle-sensitive* access control. LGI requires each group to have a set of mutually trusted agents to enforce the law.

Rei is another recent general policy language, designed to meet the requirements of ubiquitous computing [KFJ03]. Its semantics are formally grounded (Prolog) and policies written in a standard semantic language (RDF-S). The policies themselves support the deontic logic concepts of rights, prohibitions, obligations and dispensations, but this leads to potential run-time conflicts (such as when an obligated action is also prohibited) and additional meta-policy is required to resolve these.

PERMIS is an RBAC infrastructure based on the X.509 authentication and authorisation standards [CO02]. In contrast to OASIS, rôles are persistent, not parameterised and organised into a hierarchy [BMCO03]. Authorisation policy is specified in a bespoke XML-based format.

SAML stands for Security Assertion Markup Language [OAS03]. It was designed as a method of sending and receiving authentication information via XML, although it also includes some basic authorisation semantics. It has some features, such as an "evidence" field that could make it very useful for evidence-based systems, such as SECURE, but responses are limited to Permit, Deny, and Indeterminate. It has been proposed as the standard authorisation interface for the Grid [Cha03].

XACML is the eXstensible Access Control Markup Language [Mos04]. Like SAML it is an XML-based standard produced by the OASIS Open consortium, but is a language for specifying access control policy rather than just being a protocol for forming access control queries and receiving the

result. It conforms to the ISO standard access control model of separating enforcement and decision functionality [BMCO03] and allows query results to be one of four types: Permit, Deny, Indeterminate (the decision-maker is unable to evaluate the request due to an error, a missing attribute for example) and Not Applicable (the decision-maker does not have any policy that applies to this request). Authorisation and obligation policies are supported and, as the name suggests, XACML is flexible and highly extensible.

Administering Policy

In large-scale systems, the mechanism for decentralised administration of policy (such as the creation and deletion of rôles) is an important consideration. [SBC+97] proposes the use of a second level of RBAC — so called meta-policy — to administer policy, but then meta-meta-policy is needed to administer the meta-policy, and so on. The semi-structured and hierarchical nature of XML introduces new challenges when controlling access to policies written in XML [FM04, WO04].

Creating policies is another open area of policy administration research. The main issue here is automating the process of converting high level strategic policy to low level technical policy and distributing this to all enforcement points [BLMR04].

2.2 Decentralised Trust Management

Trust management is a complementary mechanism for managing access control policy in large-scale, widely-distributed systems. The term was introduced by Blaze *et al.* [BFL96] to describe the inter-related problems of security policies, security credentials and trust relationships in networked systems.

2.2.1 Identity in Distributed Access Control

In traditional, non-distributed systems, the identity of a user making a request is determined by the *uid* of the process. The uid of a process is controlled by the operating system kernel, which is implicitly trusted in such matters. In distributed systems, no such trusted supervisor is available.

Kerberos [And01] solves this problem using a shared authentication server that, upon successful authentication, issues a unique encryption key, tied to the user's identity, that may be used to communicate with authorisation servers for the duration of the user's session. The authentication server shares the user session keys with trusted "ticket-granting servers" that control access to resources. When Alice wishes to access a resource, B, she makes a request to the ticket-granting server, S, using the unique session key obtained from the authentication server. S sends Alice a ticket (an encrypted message that only B can read), along with a new key K_{AB} that is known only to Alice and B. Alice then forwards the ticket to B, along with a message encrypted under the

key K_{AB}. B replies to the message using K_{AB} thereby proving it knows K_{AB} and asserting his identity.

Kerberos relies on a centralised infrastructure to solve this identity problem — the authentication and ticket-granting servers must be available throughout operation (since sessions are short-lived and must be refreshed every few minutes to prevent certain types of attack), and they must share keys with all the resources Alice might wish to access.

Public-key encryption [ACGW99, And01] offers an alternative solution. The principal generates a *key-pair*, one public part that is known to all other principals, and one private part that is kept secret. Messages *signed* with the private part can be verified as originating, untampered with, from a particular principal, using the associated public key. However, this only proves that the message originated from the person who holds the other part of that particular key-pair — it conveys no *a priori* information about the identity of that principal. Thus the binding of a key to a name known to the authorisation server remains a problem. The two best-known solutions (so-called Public Key Infrastructures, PKI) to this problem are the X.509 authentication framework [EFL+99] and the PGP email security system [ACGW99].

X.509 requires a hierarchical infrastructure of *certification authorities* (CAs); when a user creates a new key-pair, he or she must have it certified by a CA which verifies the associated (identity) information. In order for two parties registered with different CAs to interact using public-keys there must be a path from one CA to the other, that is, there must be a "root" CA that is trusted, probably indirectly, by both users.

In contrast, PGP uses an anarchic *web-of-trust* approach where any user may certify the key of another user [ACGW99]. Thus, if Alice has a copy of a public-key that she knows belongs to Bob, and is confident that it has not been tampered with since he generated it, she may sign the key and pass it to Charlie. Hence, Alice acts as an introducer of Bob to Charlie. This scheme has the advantage that unlike X.509 it requires no additional infrastructure and so is completely decentralised. Conversely, the mechanism of finding a "trust chain" from Bob to David in order for them to communicate is much less efficient, and even if a chain is found from Bob to Alice to Charlie to David, Bob must trust Alice's judgement that Charlie is trusted to correctly verify David's identity.

2.2.2 Trust Management Systems

Blaze *et al.* noted the problems in the "trust architecture" of PGP and X.509 and the fact that these systems address only one facet of the overall *trust management problem* [BFL96]. They also observed the issues of policy administration in distributed systems, noted in section 2.1.2, and subsequently proposed a general framework for reasoning about security policies, security credentials and trust relationships [BFL96, BFK98, BFK99].

Traditionally a distributed access control system based on a PKI must determine the answer to two questions:

1. Who is the holder of this public-key?

2. Can this public key be trusted for this purpose? (For example, being granted a requested access right.)

A trust management system generalises this process such that the requester of a service can prove directly that they hold credentials that entitle them to be assigned a particular privilege. Thus, it is easy to write a trust management policy that specifies that a particular cryptographic key may be *trusted* to authorise purchase orders of up to £500, but purchase orders for greater than that value must be authorised by k from n possible keys.

The first trust management system was PolicyMaker [BFL96]. It acted like a database query engine, responding to queries from an application to determine whether a particular request complied with the current policy. A query consists of a set of local policy statements, a collection of credentials and a string describing a proposed trusted action. The returned decision may either be a simple yes/no, or a list of additional restrictions that would make the proposed action acceptable. Certificate management (verification, validation and revocation) is handled by the calling application.

Trust may also be delegated: local policies can be augmented by signed external policies (called certificates), provided there exists a trust chain from the key which signed the certificate back to the local policy. If such a chain exists then, subject to any *filter* conditions imposed by policies in the trust chain, the certificate is treated as part of the security policy when answering queries. Filters may be arbitrarily complex since PolicyMaker policies can be written in any programming language that may be executed in a safe manner (the paper describes a prototype based on a simplified version of AWK). Thus, authorisations may be delegated in a controlled manner where trust assumptions are explicitly modelled.

KeyNote improved on PolicyMaker by standardising the assertion language used to write policies [BFK98]. The KeyNote language is simpler, making it less resource intensive and easier to analyse. This standardisation also made it easier to exchange policy and access control information that would otherwise be stored in application native formats. KeyNote also provided better support to calling applications by, for example, verifying signatures, but like PolicyMaker it lacked any concept of "negative credentials" (such as credential revocation lists).

SPKI (Simple Public Key Infrastructure) describes a standard format for authorisation credentials [EFL+99]. Unlike KeyNote and PolicyMaker, it does not support fully-programmable credentials, hence the model for checking compliance is considerably simpler, but it does include support for revocation lists and validity periods.

Fidelis is a trust management system that distinguishes the concepts of policies and credentials [Yao03]. Credentials are modelled as *trust instances* which may be interpreted by the Fidelis Policy Framework using semantics determined by the local policy. Thus, in contrast to the PolicyMaker/KeyNote approach, *trust* is delegated instead of the authorisation

decision itself. The concept of explicitly modelling the *transfer* of trust information, as opposed to the delegation of trust, is discussed in more detail in section 2.3.

SULTAN is a trust management extension of the Ponder policy language described in section 2.1.2 [GS03]. It adds trust, distrust, recommendation and disrecommendation to Ponder, thus permitting the modelling of trust management systems and also the conveyance of trust, as found in Fidelis. SULTAN also permits *degrees* of trust to be modelled over the range $[-100, 100]$, where -100 is fully distrusted and 100 is fully trusted.

RT is a rôle-based trust-management framework [LMW02]. Trust management is used to control admission to rôles, and determine the mapping of privileges to rôles. Distributed policy management is facilitated by localising the authority of rôles — rôles are defined by the namespace of a particular entity and that entity's policy determines who is permitted to enter that rôle. Hence RT offers a trust management system combined with the advantages of RBAC, such as the ability to selectively activate only the rôles required for the current goal.

Cassandra is an access control policy framework that aims to incorporate all common access control idioms, including trust management [BS04b]. Cassandra was also discussed in section 2.1.2.

Weeks [Wee01] presents a mathematical framework for understanding and comparing trust management systems.

2.2.3 Trust Negotiation

Although trust is not reflexive in nature, some transactions may require that a trust relationship exist in both directions in order for the transaction to take place. For example, when two entities that were previously unknown to each other meet, there may be something of a "Catch-22"[2] situation with neither principal willing to disclose certain credentials, such as membership of a consortium, until the other has proved themselves to be sufficiently trustworthy (for example by being a member of the same consortium). The use of a negotiation protocol to gradually disclose credentials and establish trust is called *trust negotiation* [WL02, Yao02].

2.3 Trust Modelling

While trust management systems allowed trust to be explicitly modelled in access control policy, trust is still a binary concept: an entity is either trusted or untrusted. This is partly because, as noted above, in PolicyMaker/KeyNote delegating trust meant trusting the delegation of an authorisation decision which usually has a binary (grant/deny) outcome.

[2] A set of circumstances in which one requirement is dependent upon another, which is in turn dependent upon the first, from the 1962 novel by Joseph Heller.

Beth, Borcherding and Klein [BBK94] introduce the concept of *degrees* of trust (later incorporated into Trust Management in SULTAN). They also note the need to distinguish between *direct* and *recommended* trust — that is, trust developed as a result of direct interactions with an entity and trust that relies upon third parties (who are themselves trusted to provide such information). However their model takes an entirely statistical approach to trust, using it as a metric of an entity's reliability at performing some action. Maurer similarly uses trust as a metric of confidence (in the interval $[0, 1]$) in a principal's honesty and thoroughness in authenticating another principal's public key before signing it [Mau96].

Abdul-Rahman and Hailes developed a general distributed trust model [ARH97, ARH00] based on the definitions and properties of trust discussed in the social sciences. They follow Gambetta's definition [Gam00] that trust in a principal is a subjective degree of belief about whether that principal will or will not perform that action.

This definition of trust is augmented by defining three types of trust: *interpersonal*, which is the direct trust between two entities; *system*, which is the trust in the perceived properties of a system; and *dispositional*, which models how trusting a principal is in general — for instance, someone with too much dispositional trust might be considered gullible, whilst someone with very little dispositional trust might be considered paranoid. Like Beth, Borcherding and Klein, a distinction is made between direct and recommended trust and interpersonal trust is considered context-specific: Alice may trust Bob to repair her car without trusting him to fly a plane.

[ARH00] defines a model of interpersonal trust with the following characteristics:

- trust is context-dependent;
- there exist positive and negative degrees of trust;
- trust is based on entities' experiences;
- trust information (reputation) may be exchanged with other entities using recommendations, but trust is not transitive;
- trust is subjective;
- trust is dynamic and non-monotonic (that is, further experiences and recommendations may increase or decrease the level of trust in another entity).

Abdul-Rahman and Hailes reject the use of probability as a trust metric as being semantically meaningless, and unsuitably transitive. Instead agents are assigned a trust degree on a four point scale (roughly corresponding to "very untrustworthy", "untrustworthy", "trustworthy", or "very trustworthy"). Recommender trust is evaluated using the concept of *semantic distance*. This is a measure of how different (within a particular context) a principal's recommendations are from the experiences of the principal evaluating the recommendation. For example, if Alice finds that entities which Bob recommends

as "trustworthy" in context c are, by her definition, "very trustworthy" then the semantic distance for Bob in c is $+1$. Thus, distributed trust represents the transfer of trust information that is then interpreted locally, quite different from the notion of delegating trust in PolicyMaker/KeyNote. This notion was later incorporated into trust management systems in Fidelis (see above).

In their conclusion, Abdul-Rahman and Hailes recognise the ad hoc nature of their model, particularly the trust metric. They also leave undefined the specification of contexts, and how to use the results of their model to take trusting decisions/actions.

2.3.1 Formal Modelling of Trust

"Trust" is a complex notion and thus modelling it is not easy. McKnight and Chervany defined the problem space with their typology of trust (first published in 1996 but described in more detail in [MC01]). Although their typology is large, the three types most relevant to models of trust in Computer Science are those used in [ARH00], namely interpersonal, system and dispositional.

Jøsang provides further insights into the properties required of a formal model, noting that trust is a *belief* [Jø96]. For example, when a human states that they trust a *system* they mean they believe it will resist malicious attacks. Similarly when a human trusts another *human* they believe that the person is not malicious. Jøsang also discusses the suitability of using probability to model trust: while objective probability is clearly meaningless, subjective probability does represent a belief and so could be valid. If the knowledge on which trust is based is modelled as objective evidence then subjective probability still fits. However subjective probability must follow the rules and axioms of objective probability and since probability is transitive, this would cause trust to be transitive which intuitively is not the case. Jøsang [Jø96] gives a concrete example:

> If person A has made a good deal with a particular shopkeeper S at the market, and A then recommends the shopkeeper to person B, the probability theory would require B's final trust in the shopkeeper S to be:
>
> $$t(B \rightarrow S) = t(B \rightarrow A) \times t(A \rightarrow S)$$
>
> However it may be that the shopkeeper does not like person B and therefore would cheat him. Anybody with this knowledge [*that S cheats people he/she does not like*] would also know that the formula does not hold.

Jøsang later proposed a Subjective Logic ("Logic for Uncertain Probabilities") based on the Dempster-Schafer theory of evidence as a more accurate model and metric [Jø01]. Dempster-Schafer allows evidence for and against trusting a principal to be formally modelled, and since a lack of belief does not imply disbelief, **uncertainty is modelled explicitly**. The Subjective Logic uses Dempster-Schafer to model trust, and also defines the *discount* and *consensus* operators: $\otimes$ and $\oplus$. Respectively these allow a principal A to adjust (scale)

the *opinion* (trust value) of another principal B according to A's trust in B, and the combination of two opinions into a single aggregate one — such as when including a third party's opinion into an existing one.

Yu and Singh present a belief-based distributed reputation management system, also using the Dempster-Schafer theory of evidence [YS02]. However, unlike Jøsang's model, they use a a metric in the interval $[0, 1]$ for trust and thus do not explicitly model uncertainty.

Bayesian frameworks offer an alternative mechanism for reasoning about belief and combining evidence. As such, trust models based on Bayesian networks are also popular (for example, [BB04, WV03]). Although they do not have the expressiveness of the Subjective Logic they are more tractable for practical applications. Bayesian-based trust models are described in more detail in section 6.5.

2.3.2 The SECURE Trust Model

The SECURE trust model combines the trust management and formal modelling approaches by extending Weeks' mathematical model of trust management systems [Wee01] to include evidence-based systems. Thus, each principal can express their *trust policy*, that is the assignment of trust values to principals, as a mathematical function of other principals' trust policies [CNS03]. These trust policies can then be combined to produce a consistent trust assignment for all principals.

In line with other models of trust, trust values are only valid within a particular context. Thus, the trust model outputs $P \times T_v$, where P is the set of known principals, and T_v is a list of (t_i, c_i) pairs, where t_i is the trust-value assigned to p for trust-context c_i. The domain of t_i in trust-context c_i is the lattice T_i over which there are two orderings defined, trust (indicating increasing/decreasing trustworthiness) and information (how much evidence was used to calculate the trust-value). Thus, trust policies may be combined with the least fixed-point operator, as in Weeks' model, provided all such policies are suitably monotone.

Trust is determined to be based on two factors: (direct) **observations** and (third-party) **recommendations**, but trust values may have any form that satisfies the requirement for two orderings. Shand demonstrates the use of intervals, such as $[0\%, 100\%]$, and (m, n) "evidence counts" where m represents the number of successful transactions and n the number of failed interactions [Sha04]. An example application using this trust model is given in section 5.2.4.

The Revised SECURE Trust Model

Although trust is universally accepted to be in some way *context-dependent*, no researchers have yet addressed how trust-contexts can be formally incorporated into their models.

In light of the work on risk modelling described in chapter 4 of this thesis, SECURE decided that its model of context should incorporate outcomes. Work on context modelling for fields such as ubiquitous computing [WGZP04]

has suggested an ontology-based approach is required, and thus in SECURE context is modelled as a hierarchical *event structure of outcomes* [CDKN04]. For example, figure 2.1 shows an example of an event structure where the following outcomes are possible:

- no outcome yet ($\emptyset$);
- the outcome was the *reject* decision;
- the outcome was the *accept* decision, but the final outcome of *forged* or *verified* has not yet been observed;
- the outcome was *forged* (implying outcome *accept*);
- the outcome was *verified* (implying outcome *accept*).

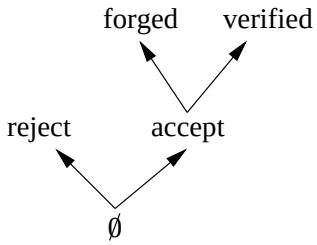


Figure 2.1: Example of an event structure of outcomes.

The "revised" SECURE trust model also standardised the use of (s, i, c) triples as the format of trust values. Since contexts are now outcome-based, s represents the number of pieces of evidence supporting a particular outcome, c the number of pieces of evidence contradicting a particular outcome and i is the number of pieces of evidence that are inconclusive. Using the example of the event structure shown in figure 2.1, an observation of outcome *accept* would lead to the nodes of the event structure for a particular principal being annotated with the trust values shown in the "intermediate" column of table 2.1.

	Trust Value (s, i, c)	
outcome	intermediate	final
reject	$(0, 0, 1)$	$(0, 0, 1)$
accept	$(1, 0, 0)$	$(1, 0, 0)$
forged	$(0, 1, 0)$	$(0, 0, 1)$
verified	$(0, 1, 0)$	$(1, 0, 0)$

Table 2.1: Trust values attached to event structure outcomes after observing an *accept* (intermediate) and *verified* (final).

If it was later observed that the outcome of the same transaction was *verified* then the trust value associated with *forged* would have 1 taken from i and 1 added to c, and similarly for *verified* 1 taken from i and 1 added to s, as shown in the "final" column of table 2.1. Since *verified* is the leaf of the event structure,

no more outcomes from this transaction are possible and the transaction is concluded.

(s, i, c) triples were chosen as they model both uncertainty and information. It was noted that both quantities are important when making decisions based on trust and thus needed to be incorporated into the SECURE model.

Communicating Trust Information: Recommendations

The ability to communicate trust information is necessary if principals are to learn which principals are (un)trustworthy without entering into a potentially loss-causing interaction. In SECURE, trust information is communicated using *recommendations* where a recommendation is defined to be a trust value sent from Witness W about Subject S. When transferring trust information it is important to avoid "double counting" and "chinese whispers". For example, suppose Alice receives some information about a transaction with Eve in a recommendation from Bob which she then incorporates into a recommendation she sends to Charlie. If Charlie also receives the same information about that transaction direct from Bob, his trust computation will incorporate the result of that one transaction twice: once from Alice and once from Bob. To avoid this Alice may only generate recommendations based on her direct observations. Jøsang, Gray and Kinateder consider the problems of transitive trust topologies in more detail [JGK03]. Alice could also forward Bob's recommendation to Charlie, but this has implications for Bob's privacy — what if he does not wish Charlie to know that he performed a transaction with Eve? This problem shall be revisited later in this thesis.

As per Jøsang's Subjective Logic (see above), recommendations must be discounted by a principal's *recommendation integrity*[3] in the witness before being incorporated into a local trust computation. Details of a suitable definition of this operator for the SECURE trust model have yet to be published.

Trust is also often associated with *reputation*. This differs from recommendations in that reputation is a metric of a principal's standing, whereas recommendations are a mechanism for conveying trust information. Reputation is discussed further below.

2.3.3 Time in Trust Models

Time is clearly important in trust. If privilege is gained from being trusted then principals may act in a trustworthy fashion in order to gain privileges which they later abuse. Trusted peers may also be compromised and temporarily controlled by an attacker, and it may be possible to detect untrustworthy principals by analysing patterns of behaviour over time [CM05].

Of the trust models described so far, only [BB04] and [WV03] (both based on Bayesian networks) explicitly incorporate any formal notion of time, and then only that older experiences contribute less to the trust computation than more recent ones. [Mez04] models trust as explicitly diminishing with time using an exponential context-dependent decay factor. This model also neatly

[3] also referred to as *meta-trust*, that is, the principal's trust in the witness as a recommender

models meta-trust in the same manner as normal trust, but the use of values in the interval $[0, 1]$ for trust is not as expressive as other models. It also appears to have the problem of "double counting" indirect trust described above. In summary, the paper defines:

$$\begin{aligned} T &: P \times P \times \Phi \times \tau \mapsto [0,1] \\ I(\alpha,\beta,\phi,t) &= \frac{\sum_{\gamma\in\Gamma} T(\alpha,\gamma,j(\phi),t)R(\gamma,\beta,\phi,t)}{\sum_{\gamma\in\Gamma} T(\alpha,\beta,j(\phi),t)} \text{ where } \Gamma = \text{set of recommenders} \\ R(\alpha,\beta,\phi,t) &= \psi T(\alpha,\beta,\phi,t) + (1-\psi)I(\alpha,\beta,\phi,t) \end{aligned}$$

T is (direct) trust, I is indirect trust, R is reputation, P is the set of principals, Φ is the set of contexts, $j(\phi)$ is the meta-trust context associated with ϕ, τ is time, ψ is the weighting of direct trust to indirect trust in a reputation. Since R incorporates some fraction $(1 - \psi)$ of I, if any γ and α have recommenders in common then the reputation information from those mutual recommenders will be counted more than once (albeit scaled by $(1 - \psi)$).

2.3.4 Global Reputation as a Trust Metric

The unregulated nature of peer-to-peer systems and their recent growth in popularity has given rise to a number of trust metrics designed for such environments. These typically make the assumption that if principals have an incentive to be reputed to be trustworthy — for example, in order to attract more customers — then they will act in a trustworthy fashion in order to maintain and improve their reputation. Thus, systems such as [AD01, XL02, KSGM03] define trust in terms of a global reputation metric. This narrow definition of trust does not conform to the properties of the formal models of trust described in section 2.3 since it represents only one facet of the term, but an overview of three popular "reputation as trust" metrics is given here for completeness.

Aberer and Despotovic propose a system that counts the number of complaints filed by other peers to judge the trustworthiness of a principal [AD01]. Complaints are published and found using their p-grid peer-to-peer architecture, and nodes for whom there are a higher than average number of complaints are considered untrustworthy. However, their metric makes the assumption that all malicious nodes will file complaints about nodes that they cheat in order to hide their deceit, and so a principal may obtain high trust simply by never filing a complaint against another node — even if they have been cheated and have a valid complaint, their own reputation value benefits by not complaining.[4] PeerTrust [XL02] also uses the p-grid architecture and an improved version of the complaint-based metric that takes the number of interactions a principal has performed into account. Both models use thresholds to determine at what level a principal is considered *trustworthy*: Aberer and Despotovic use an ad hoc heuristic; PeerTrust allows the threshold to be determined by the user, modelling their *dispositional* trust. A third metric, eigentrust, defines trust

[4] Alternatively a principal could operate under two separate identities, one for giving services and one for using services.

to be transitive and uses probabilistic methods to determine a trustworthiness ordering over all known peers [KSGM03].

The problem with using global reputation as a trust metric is that it cannot take into account "peer-specific" behaviour, that is, when a peer treats some minority subset of nodes differently from the general populace. For instance, if a malicious auction seller cheats all users from the domain `.co.uk`, but is honest towards all others, they will maintain an acceptable reputation and make a better profit than an honest peer. In addition, since global reputation metrics are public, attackers would be able to adjust their behaviour to optimise the balance between reputation and profit. Global reputation metrics also usually rely on the majority of nodes to be honest, an assumption researchers have found to be doubtful in open systems [Dou02]. A third problem with the specific systems mentioned here is a failure to separate direct and recommended trust. They assume that principals who, for example, give good service, will also be honest when recommending other peers, and hence the metrics offer no defence against those peers that give good service but defame other peers in the system.

XenoTrust demonstrates a novel method of modelling reputation [DKHP03, DHH+03]. Free form reputation statements about other principals are stored in a standard relational database on which users may then perform queries using standard aggregation functions such as min, max and mean. A publish/subscribe event system [EFGK01] is then used to notify users when the aggregated reputation value changes. Although the database is centralised, there is no limit on how many reputation services could operate, or who can operate them. An economic system of earning credit for making honest statements, which is debited for each query [FKÖD04], suggests that the expectation is for there to exist a small number of well-known reputation broker services, the online equivalent of real life credit bureaus. Identities are managed by centralised authorities called XenoCorps to prevent nodes with bad reputations escaping punishment by changing their name.

2.3.5 Summary of Trust Management and Trust Modelling

Trust modelling develops formal models of trust in entities. Trust management models trust relationships between entities, and often also refers to the exchange and dissemination of trust information. Many researchers see the problems as being inter-related and have produced integrated systems such as SECURE [CNS03], that incorporate both modelling and management.

Trust is a subjective belief. Abdul-Rahman and Hailes define "A trusts B" to mean that "A believes B to be trustworthy." Trust beliefs can be formed from direct experiences (observations) and recommendations from other principals trusted to provide honest information. Recommendations can be used to propagate a principal's reputation and thus the need to maintain a good reputation can be an incentive for a principal to behave in a trustworthy manner, but trust is in general not transitive.

Trust is context-dependent. Thus, a server that is trusted to serve web pages is not necessarily trusted to handle email. Similarly, a principal trusted to provide good service may not be trusted to recommend others to provide

the same service, for instance if they have a vested interest in customers not discovering their competitors. Following human intuitions of trust, it is felt that it should be possible to infer trustworthiness in one context from trustworthiness in another, to a limited extent. The SECURE project's use of event structures to model context permits this to be done using the mathematical concept of *morphism*.

Few researchers give much consideration to the use of trust information to take decisions. The usual method, when it is considered at all, is to use a threshold to partition principals into trusted/trustworthy and untrusted/untrustworthy. These thresholds may be configurable by the human user [XL02, YS02], or based on heuristics that look for principals with reputation values outside of a norm [AD01].

2.4 The SECURE Framework

The SECURE trust model described in section 2.3.2 represents only one component of the SECURE framework. An overview of the entire framework is shown in figure 2.2. There are five major component systems: trust, risk, access control, evidence and support.

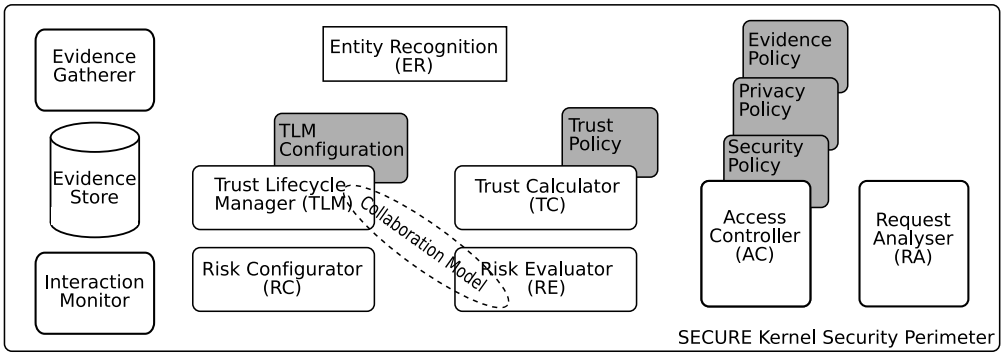


Figure 2.2: Overview of the SECURE model

Trust: The Trust Calculator computes the trustworthiness of a principal using the model described in section 2.3.2. The Trust Life-cycle Manager is responsible for the formation, exploitation and evolution of trust information, over which the TC then performs its calculations.

Risk: A key observation on the use of trust is that *trust is unnecessary unless there is something at risk.* Therefore the Risk Evaluator determines the risk of the current situation using information supplied by the Risk Configurator. The risk elements of SECURE are developed in chapter 4 of this thesis.

Access Control: The Access Controller allows for the specification of access control policies that explicitly reason about trust and risk, and make decisions autonomously. Access control for SECURE is developed in chapters 5 and 6 of this thesis.

Evidence: The Evidence Store holds all the necessary information (details of past experiences and received recommendations) for the TLM and RC to perform their tasks. The Evidence Gatherer is responsible for obtaining recommendations from other principals, and the Interaction Monitor observes the results (where possible) of any interactions that take place.

Support: The Request Analyser encapsulates the external Application Programmer Interface (API) of the SECURE kernel. It processes requests, forwarding them to the various SECURE sub-systems as appropriate, and mediates any trust negotiation. Entity Recognition is responsible for managing the process of identifying principals. Applications may use a variety of different mechanisms, such as public-keys, biometrics, passwords or similar, and since some mechanisms are more capable or secure than others, the ER module outputs a confidence value t_{id} in the recognition of a particular principal [SFJ+03].

This thesis is concerned with the risk, access control and policy aspects of the SECURE framework. A detailed software architecture for SECURE is also developed in chapter 4.

2.5 Chapter Summary and Conclusions

Traditional access control systems do not scale well to the Global Computing infrastructure where there are huge numbers of principals, resources, and policy administrators controlling access to those resources. Trust management is a key tool in decentralising the policy enforcement and administration, therefore allowing access control systems to scale. A formal understanding of trust allows these highly decentralised policies to be better analysed and understood in the context of higher level policies. More recently, evidence-based computational models of trust that permit trust assessments to evolve in response to events have been developed.

However, many researchers focus so intensely on the nuances of the trust modelling itself that some have observed [Lan03] that paper authors do not specify what a working system should look like, hence making evaluation appear impossible. In the case of this thesis, and the SECURE project in general, providing access control for the global computing applications described in chapter 1 is the goal. Having now surveyed the state of the art in distributed access control and trust management it is now possible to articulate a more refined goal. The aim of this thesis shall therefore be that:

> For a given application domain, a computational trust system (such as SECURE), should be able to select an interaction partner (or partners) such that the risk of interacting with that principal is deemed to be acceptable to the decision-maker.

CHAPTER 3

Trust for Ubiquitous, Transparent Collaboration

In this chapter, a simple trust and risk framework limited to facilitating secure collaboration in ubiquitous and pervasive computer systems is presented.[1] It illustrates in more detail one of the application scenarios from section 1.1.

Ubiquitous computing needs trust between participants in order to support collaborative activities, such as arranging meetings, while protecting sensitive information used in the collaboration. At the same time, security measures must be proportional to the risk involved to allow the interaction between devices to be as automated as possible.

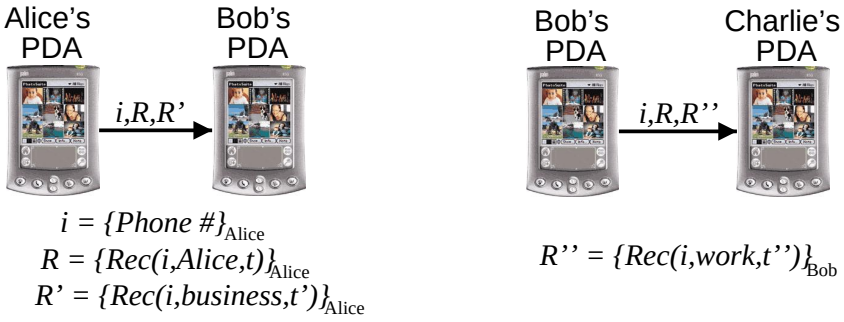
For example, consider a business meeting with representatives from two companies. To schedule a follow-up meeting, the attendees would like to find a time that suits everyone, with the help of electronic diaries and calendars. However, depending on the trust between the companies, they might not want to disclose their detailed movements to each other. Instead, the members of each company might decide to find potential meeting times among themselves, then share only this aggregate information between the companies. This chapter proposes trust and risk models to help automate interactions of this sort, making the computations as unobtrusive as possible while still respecting participants' trust beliefs.

3.1 Trust Infrastructure

Mutual trust is crucial for ubiquitous devices, which must share information and work together to present an unobtrusive interface [GSSS02] to their users.

[1] This PDA scenario and its model of risk extend earlier work with Brian Shand and Jean Bacon, published in a paper at the PerCom 2003 conference [SDB03].

Using a homogeneous recommendation system, a trust framework allows users to share and exchange privileged information. This information can include conventional data such as personal contacts and calendar entries, and also trust beliefs about principals.



(a) Alice sends Bob her work phone number.

(b) Some time later, Bob forwards Alice's number to Charlie.

Figure 3.1: Recommendations in action.

Recommendations can be used to associate data items and explicitly include a degree of confidence (trust) in that association. For example, Alice might give a telephone number to Bob, together with recommendations that it is her telephone number and that it be considered privileged business information. This is illustrated in figure 3.1(a). Alice signs i to certify that she is the origin of the information and also signs her recommendations to allow the recipient to evaluate their relevance using Alice's trust-rating. t represents Alice's *trust* in her recommendation, that is, how much confidence she has in it. Later, Bob forwards Alice's number to Charlie (figure 3.1(b)), along with her recommendation that it is her number (R) and Bob's recommendation that it is her "work" number (R'') in which he has trust t''. He could also forward Alice's original recommendation R' that it is her "business" number if he wished to, but he has chosen not to in this case as the transmission link is expensive and he thinks Charlie will find his recommendation more useful.

Existing trust models for pervasive computing typically represent trust using a security policy which explicitly permits or prohibits actions [FJK+01]. These policies are not well suited to dynamic environments, in which participants have only partial trustworthiness, and trust assessments must constantly change. To avoid this, Abdul-Rahman and others [ARH00] have also proposed explicit recommendation systems, but with only very simple trust values. In the application described in this chapter, recommendations are used to control the flow of information, as well as for access control and the more complex recommendation structure can also be combined consistently, by formally ordering recommendations according to information content [CNS03]. This gives a well-founded approach to trust management decisions, which is suitable for distributed computing applications.

Principals in the framework can also be associated with categories using

the recommendation system, and being a member of a category may confer certain access control capabilities. Bob might send a recommendation $\{Alice, work, t_1\}_{Bob}$ to himself, that recommends Alice as a member of category "work" with trust t_1.

Principals and information are thus associated with categories using the recommendation system. Each item might have more than one recommendation, whether from different principals or for different categories. The trust model assesses the importance of an item (with respect to a category) by combining all the pertinent recommendations.

In the example above, the importance of displaying Alice's telephone number in Charlie's work category would depend on the degree to which Alice recommended the number, Bob recommended the number as a "work" number and Charlie's trust in Bob as a business acquaintance. Furthermore, Bob would not pass on the number automatically to Charlie; his PDA only sends the number because it knows Charlie is a trusted business acquaintance.

In order for a PDA to make these decisions automatically on behalf of its owner, it must understand the dynamics of how the owner themselves would make the decision — usually by assessing the trust and risk in the current context.

The use of categories to assign access privileges is discussed in more detail in section 3.2. First, the following section extends the example to show how recommendations give structure to data.

3.1.1 Phone Book Example

A phone book exchange service illustrates the need for and advantages of trust-based information exchange for ubiquitous computing. Users of handheld computers currently exchange contact details laboriously on a one-to-one basis. Furthermore, there is no associated trust information, so users cannot recommend to whom the information should be redistributed — for example, private and business numbers are usually redistributed together.

In this section, it is shown how the trust and risk framework can make this service more transparent for users and increase automation while preserving the privacy of personal information.

The phone book database consists of many items, each with associated recommendations. These may be signed to prove their authenticity, using a public key infrastructure.

Accessing and displaying information

Each information item has a unique identity, consisting of the author and a secure hash of the contents; any reference to an item uses this identity. As a result, recommendations about an item will cease to apply if the contents are changed. In the case of a phone book, these contents might be a name, a phone number or an address.

Figure 3.2 illustrates how Charlie uses the trust model to display Alice's phone book information in the example above. When Charlie searches his phone

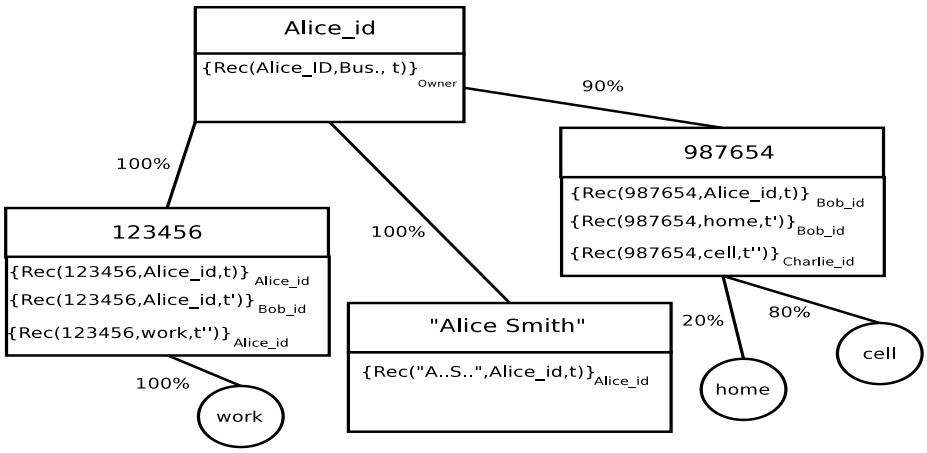


Figure 3.2: Each piece of information is stored separately and links between them are determined by the trust model.

book for "Alice", he finds the entry for Alice's name, ranked according to the strength of its recommendations. If he views that entry, he is presented with the linked information too, again weighted by importance. Very unimportant entries might not be displayed at all, according to a threshold set by Charlie.

In contrast, consider David, Charlie's colleague who is allowed to view business information in Charlie's phone book, but nothing personal. If David views Alice's information there, he is presented with the restricted view shown in figure 3.3. Furthermore, the importance of links might be different, if David had other knowledge of Alice, such as an old work number that is now out of date, as illustrated here.

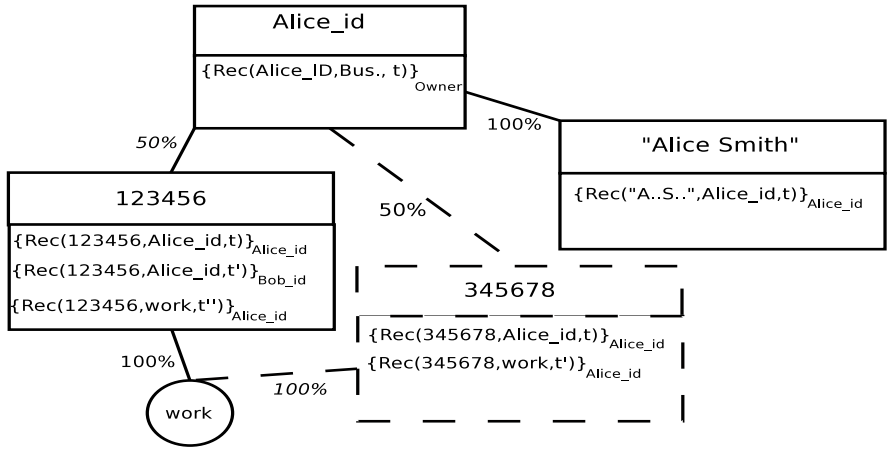


Figure 3.3: David receives a different view on Charlie's information about Alice, plus additional information from his own database (shown as dashed lines).

3.1.2 User Privacy

When Alice gives her phone number to Bob, she trusts him not to redistribute it to people she would not want to know her telephone number. However, Bob must then realise that Alice has given him her direct line number instead of the switchboard and not pass it on indiscriminately. In our example in figure 3.1, Alice's recommendation R states that she recommends that Bob treat this as business information and not a public number.

Security expert Bruce Schneier asserts that security is a process of trade-offs [Sch03] and that every "security" measure must be evaluated as to whether the security gained is worth the cost [Sch02]. Sometimes the cost trade-off is subtle and indirect — enforcing strong password policies can make it more difficult for users to remember their password and thus they take steps, such as writing it down or reusing it in other domains, that actually reduce the security of the password.

In a pervasive computing environment the cost of security is an inability to collaborate with other ubiquitous devices, or lost productivity as the human owner must explicitly set security policy for each new device he or she encounters. Accordingly, the aim of this work is to create a security mechanism suitable for use in the pervasive computing environment where human intervention is a valuable resource [GSSS02], yet due to the nature of the data involved, security remains important.

The following section shows how rôles and categories can be structured to preserve the *meaning* of recommendations. This ensures that user privacy is better protected in automated information transfers, by unifying trust assessments with access control.

3.2 Categories and Rôles

Information exchange is restricted with the help of categories, arranged in a partial order. Each PDA owner can have their own set of categories and associated partial ordering. These categories restrict the distribution of information, and the actions of principals, and are analogous to rôles in a Rôle-Based Access Control system [BMY02].

This model extends traditional RBAC rôles by associating a trust assessment with each category assignment. Users of the system can then combine a risk assessment, together with their trust in the information, to decide whether or not it should be used or displayed. For example, the risk of displaying an incorrect telephone number might depend on the cost of the user's time when attempting to use it. Conversely, if the number is not displayed (or is shown as less important), the risk is that the user might not find it, even though it is correct.

Each category has a list of privileges associated with it; these are action and category pairs which can be used by principals associated with the category. The overall trust assessment of an entity is thus a mapping from action and category pairs to primitive trust values. These trust assessments and the contribution of other recommendations are expressed formally as a local policy

function, defined in [SDB03], analogously to Weeks' proposal for formalising access control system policies [Wee01].

Categories are arranged in a natural privilege hierarchy: when category c_0 extends the privileges of another c_1, write $c_0 \supset c_1$. Figure 3.4 illustrates a typical hierarchy, where the top category $\top$ contains the owner of the PDA, c_n represent user defined categories such as immediate family, business colleagues, business contacts, friends and relatives. A are acquaintances, people known to the owner, but not categorised, and S are strangers.

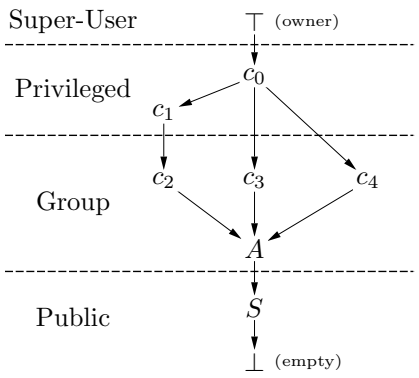


Figure 3.4: Example of a category hierarchy.

Band	
Super-User	As Privileged, but may also delete information.
↑	
Privileged	As Group, but can also read own category.
↑	
Group	As Public, but can also write to own category and those below.
↑	
Public	Read: Categories lower in graph. Write: None.

Table 3.1: Privileges are conferred on a category by membership of a *Privilege Band*.

The diagram also shows another important feature of the framework, from a human interaction perspective. Categories are divided into bands (see table 3.1); and these bands then dictate the extra privileges granted to categories within them. This makes it far more convenient for users to manage their trust policies, simply by moving categories within their trust lattice. For example, categories in the "Group" band can by convention recommend that members may write data to their own categories and those below them, and read data below them.

The banding of categories allows user privileges to be easily and intuitively assigned. However, if necessary the banding may be overridden, by explicitly associating extra permissions with categories or users.

Formally, these bands are a partition of the privileges of the system. For example, if $p(c)$ represents the privileges of category or band c, then $p(\text{Group}) = (p(c_2) \cup p(c_3) \cup p(c_4)) \setminus p(S)$, the marginal privileges accrued by categories within the band.

The category bands also have a second function: they facilitate information exchange, by providing a common framework for expressing category meaning between devices, even devices with otherwise different categories.

This allows recommendations between devices to be made in terms of category bands. As long as users attribute similar meanings to these bands — encouraged by band privileges — then information transferred between devices will automatically be restricted to the appropriate band, unless there is an explicit user override. This is particularly useful in avoiding sensitive information

being leaked by different collaborating users assigning different meaning to categories of the same name or placing them in different bands. For example, suppose Alice has two categories, *business* and *business contacts*, the former in the *Privileged* band and the latter in *Group* and she gives her *business* number to colleague Bob, whose *business* category is in *Group*. However, if Alice sends an additional recommendation that the number is *Privileged* then Bob's address book should respect this when calculating access rights.

3.2.1 Categories in Calendars

The same framework can also be used for calendar information. In the phone book there were read and write capabilities for viewing, inserting and updating phone numbers. When a principal attempts to read a particular time-slot, the information returned will depend on their location in the hierarchy of categories in relation to the category of the appointment. All appointments have a projection into categories lower in the hierarchy, although not into $\bot$. This has the effect that a principal who does not have read permission for an appointment sees the lower category projection that the time is busy, tentative or free but not the details of any appointments. Because appointments are not projected into $\bot$, principals in sufficiently low categories (such as S in figure 3.4) do not see anything at all and can learn no information about the owner's schedule.

Write permission to a category is the ability to make an appointment in that category, and categories could be used to determine the default response to an appointment made in a free slot (for example: "automatically accept all appointments made by principals who are members of category PhD-Supervisor").

3.3 Implementation Issues

Trust management through recommendations is well suited to mobile and distributed applications, since recommendations conveniently factorise and encapsulate trust policy (section 2.3.2).

This is particularly important for vulnerably connected nodes such as PDAs, which must store the relevant components of others' policy locally, for use when disconnected. Transferring only a few recommendations from a trust policy is justified in our application, since extra recommendations correspond to additional trust information in our partial order. Therefore using a subset of a policy corresponds to weaker policy assertions, and the resulting trust decision will also be weaker.

However, locally-cached policies must be kept up to date in order to be used appropriately. It is therefore proposed to assign time stamps and validity periods to recommendations which are then refreshed automatically each time devices interact; this removes any burden on the owner to ensure their local cache is not about to expire before embarking on a period of extended disconnection. If a recommendation does expire, using out of date policy may be preferable to no knowledge at all and so the weight of expired recommendations is scaled down rather than being discarded. However, the principal danger of outdated information is that a person may no longer deserve the privileges

that they once had, for example someone who has been fired from the company. Therefore old trust policy that says something negative about a principal cannot cause security to be compromised and so this scaling is only applied to expired positive recommendations.

Recommendation systems often suffer from issues of long trust chains, because the meaning of "trust" changes with depth in the chain. By analogy to human notions of trust, trusting someone to be a good car mechanic is different from trusting someone to recommend a good car mechanic [JGK03]. However, in this model the problem is turned on its head by using the implicit trust assumptions found in the organisation of the user's phone book to determine privilege assignments — people categorise the people they know according to the type of trust they place in them: close friends are clearly highly trusted; "business colleagues" do not try to sabotage each other's list of contacts but would not usually have access to personal numbers; and so on. People within a single category may have different levels of trust placed in them, but partial belief in category membership caters for this. When necessary, exceptions can also be made; the owner of the PDA can fine tune their policies, via the recommendation system, to customise individual users' permissions. By explicitly considering these implicit human intuitions of trust (including the overloading of the meaning of the term), the framework is made powerful while still being easy to use, although this correspondence is clearly not true in the general case. The applicability of this assumption shall be considered in more detail later.

Another common problem with recommendation systems is that the provenance of information can become distorted so that it is no longer possible to tell whether information from different principals is truly independent or in fact based on data from a single principal further up the chain. To avoid this, only recommendations based on first-hand experience may be transmitted by a PDA — the passing-on of information from another principal is done by forwarding the relevant recommendations from that principal, but this leads to privacy implications that will be explored later in this thesis.

3.3.1 Trust Values

Although a number of trust value formats have been proposed (section 2.3), the trust framework in this application uses (*belief*, *disbelief*) pairs, representing the weight of evidence for and against a particular trust assignment, with $belief + disbelief \leq 1$. This can be compared to Jøsang's logic of uncertain probabilities, based on the Dempster-Shafer theory of evidence [Jø01].

No information is represented by (0,0), while (1,0) and (0,1) represent certain belief and disbelief respectively. These trust values are ordered according to trustworthiness by defining $(b_1, d_1) \preceq (b_2, d_2)$ iff $(b_1 \leq b_2)$ and $(d_2 \leq d_1)$, which forms a lattice on the trust domain T_b.

However, there is also a second natural ordering, according to information, where $(b_1, d_1) \sqsubseteq (b_2, d_2)$ iff $(b_1 \leq b_2)$ and $(d_1 \leq d_2)$, which is used in combining recommendations, as described in [SDB03].

3.4 Risk Assessment and Decision Making

As noted above, security measures must be proportional and appropriate for the risk involved: a user may happily distribute a business card to strangers to advertise their business, but may be quite careful as to whom they give their mobile phone number.

In the same way that a principal's position in the category hierarchy (figure 3.4) assigns it a permission, the position of a piece of data implicitly gives it a value that can be used to assess the risk of an operation involving it: the higher in the hierarchy, the greater the value. The risk of an operation is defined as being the sum of the risks of all the possible outcomes of that operation. The risk of an outcome is a function of the likelihood and impact of that outcome. This is in line with existing literature on risk management, such as [SGF02] and the impact of an outcome is taken as the worst-case cost to the user should that outcome occur. This cost will be a combination of two factors: the seriousness of the outcome itself and the value of the data involved.

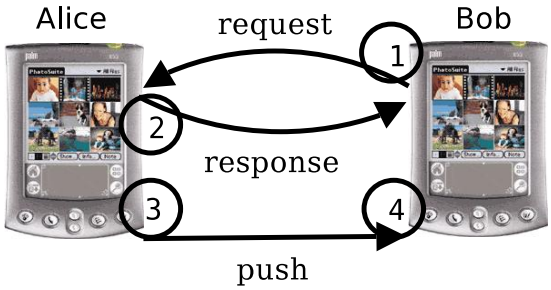


Figure 3.5: Possible interactions between two PDA users.

In the address book scenario, two users may interact in two different ways as shown in figure 3.5. Either Bob may request a number from Alice, or she may try to send Bob information, unsolicited. Before either side takes part in an interaction, there is a decision to be made (shown as the numbers 1 to 4 in figure 3.5). Those decisions are as follows.

1. **Request**: Bob wishes to ask Alice's PDA for a telephone number. As far as Bob is concerned, the possible outcomes from interacting with Alice are (in increasing order of impact):

   - he obtains the number he wanted and it is correct;
   - he obtains the number he wanted but it is incorrect (e.g. out of date);
   - he does not obtain the number he wanted.

2. **Response**: Alice receives Bob's request and must decide what access to her address book she is prepared to give him. From Alice's point of view, the possible outcomes of giving Bob access to an entry in her address book are:

- Bob obtains the number he wanted;
- Bob obtains the number, but misinterprets or ignores the attached recommendations and redistributes it indiscriminately.

3. **Push-Number**: Alice wishes to automatically send her number to certain PDAs with which she comes into contact. For example she may have changed her home telephone number recently and wish to inform all the friends she meets, but not business colleagues. Her PDA must decide whether to automatically send the number to Bob; the possible outcomes for Alice are (again in increasing order of impact):

   - Bob stores the number and respects Alice's accompanying recommendations on redistribution;
   - Bob discards the number;
   - Bob stores the number but ignores the accompanying recommendations on redistribution by, for example, storing the number in a publicly accessible area of his address book when Alice recommended that it was privileged business information.

4. **Receive-Number**: Alice wishes to send Bob some information. Bob must decide what to do with the received information. The possible outcomes from his point of view are:

   - Bob finds it useful;
   - Bob finds the information unhelpful or incorrect;
   - Alice attempts a denial of service attack against Bob's PDA by sending many numbers, aiming to fill its storage space or saturate its connectivity.

As stated above, the risk of an outcome is a function of the worst case cost in the event of the outcome occurring, and the probability that the outcome will occur, which is solely dependent on the principal(s) involved. Using the idea that trust is a measure of how well an actor is known, it is possible to assign a probability to each outcome.

The *Response* trust-decision will now be considered in more detail.

3.4.1 Deciding Whether to Participate

When Bob asks Alice for a number from her address book, in access control terms, she must decide whether to grant him read permission on that number or not. The aim of the model is to make this decision as automatic as possible, but in situations where the correct response is unclear the PDA may then attract Alice's attention and ask for her guidance. However, the cost of Alice's time to give that guidance must also be factored into the decision, so the cost-benefit analysis must take into account the benefit from helping someone by giving them a number, the worst-case cost of giving a number to an inappropriate person and the cost of asking the owner for guidance.

Formulae will now be derived for calculating the benefit of each of the three possible courses of action. Since principals and data may be filed under multiple categories in an address book, for example colleagues who are also considered friends, all pairs of categories, (c_p, c_d), must be considered where there is a recommendation (or recommendations) with $b > d$ that permit c_p to read c_d. c_p is the principal's category and c_d is the category of the data item they wish to read.

There is clearly a benefit to not giving out a number if the PDA has no trust in that person's right to that number, that is, if it does not believe them to be a member of c_p. Define:

$$\text{Benefit}_{no}(b{-}d{,}val_{c_p}) = -val_{c_p}.(b{-}d)$$

where (b, d) is Alice's belief and disbelief in Bob's membership of c_p, as computed by the trust engine described in [SDB03]. val_{c_p} is the value Alice has associated with category c_p — the more valuable the category and the greater Alice's distrust in Bob's membership, the greater the benefit of saying, "no".

For calculating Benefit_{yes}, in addition to considering how strongly Bob is associated with c_p and the expected benefit of that association ($val_{c_p} \times (b{-}d)$), the PDA must also consider the relative importance of Bob himself (inferred from the value of the category of which he is a member) compared to the importance of the data he is trying to read, which encodes the potential cost of Bob ignoring Alice's recommendations and redistributing the number indiscriminately. The function must represent the fact that there is benefit in helping someone who is potentially a close friend to read a low value number, while greater assurance is required to allow access to a more valuable number.

It is also necessary to allow the user to configure their disposition to trust [MC01], and it may be useful to take any available contextual information (such as location or the status of the owner) into account. For this purpose, the variable val_{read} is introduced which represents the importance of generally being a helpful source of information.

$$\text{Benefit}_{yes}(b{-}d{,}val_{c_p}{,}val_{c_d}) = val_{c_p}.(b{-}d) - \max\left(val_{c_d}{-}val_{read}, 0\right)$$

This definition balances the expected benefit of assisting Bob, $val_{c_p}.(b - d) + val_{read}$, against the value of the data being read, val_{c_d}. The maximum function prevents the benefit of giving out information from being greater than the expected benefit of interacting with Bob, even if the value of val_{read} is greater than the value of the data involved. Section 3.4.2 discusses the val_{read} variable further.

If a positive trust-relationship between Bob and the requested data does exist, but is sufficiently tenuous that there is no clear benefit to granting the request, then it may be worth asking Alice for guidance on the decision. To represent the cost of Alice's time in having to focus on her PDA and input the correct decision, a second user-configurable and context-dependent variable is introduced, val_{time}.

$$\text{Benefit}_{ask}(b{-}d, val_{c_p}) = val_{c_p}.(b{-}d){-}val_{time}{+}val_{read}$$

This equation simply compares the expected benefit of helping Bob (and also the potential cost if in the case of an erroneous negative decision) to the cost of Alice's time that is taken up making the decision. val_{time} is discussed further in section 3.4.2.

It follows that Alice's response to Bob's request can now be determined by:

$$\begin{aligned} \text{Answer} = \; & \texttt{if Benefit}_{no} \geq 0 \texttt{ then } \text{“No”} \\ & \texttt{else if Benefit}_{yes} > 0 \texttt{ then } \text{“Yes”} \\ & \texttt{else if Benefit}_{ask} > 0 \texttt{ then } \text{“Ask”} \\ & \texttt{else } \text{“No”} \end{aligned} \tag{3.1}$$

Intuitively, these equations determine whether there exists c_p of which Bob is a sufficiently strong member to be able to read c_d. For simplicity $(b - d)$ is used as a measure of the strength of a principal's membership of a category. Since there are three possible responses to the request — *Yes*, *No*, and *Ask owner for guidance* — the range of $b-d$, the interval $[-1, 1]$, can be seen as being divided into three corresponding regions, as illustrated in figure 3.6.

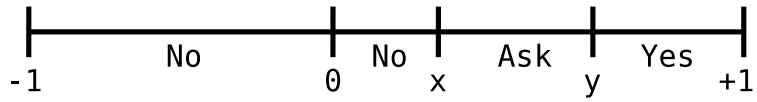


Figure 3.6: Number line showing how partitions of $(b - d)$ in membership of a category lead to a decision.

The positions of x and y may be determined by setting Benefit_{yes} and Benefit_{ask} to the threshold at which the PDA decides there is sufficient benefit to take that course of action — 0 in equation (3.1). Re-arranging to find the corresponding value of $b-d$ gives:

$$y = \max\left(\frac{val_{c_d} - val_{read}}{val_{c_p}}, 0\right) \tag{3.2}$$

$$x = \min\left(\frac{val_{time} - val_{read}}{val_{c_p}}, y\right) \tag{3.3}$$

The region $[0, x)$ is when Bob is a member of c_p, but there is insufficient benefit in saying "yes" or asking the owner to grant the request, so the answer must be negative. The region $[-1, 0)$ is also logically a negative response as there is no trust at all in Bob's membership of c_p.

3.4.2 Fine-tuning Policy

The variables val_{read} and val_{time} used in the benefit equations can be tuned by the user to give them fine-grained control over their policy and take any available contextual information into account. For example, the owner may be able to place the PDA into a "Do-Not-Disturb" mode that would scale the

value of val_{time} to infinity. The significance of the values of these variables on the decision making process can be seen by considering the effect they have on the thresholds x and y.

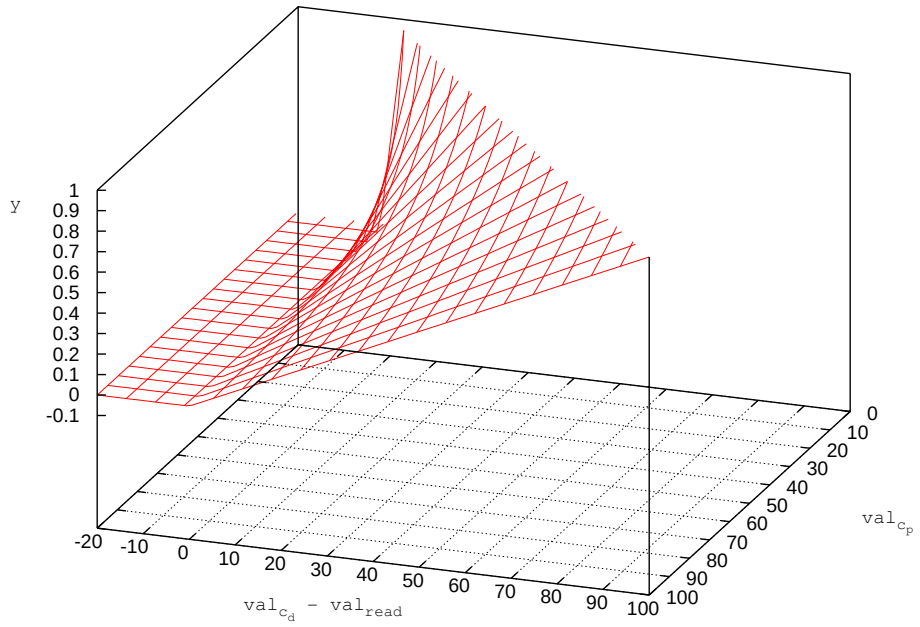


Figure 3.7: Plot showing how the amount of trust required, y, varies with the values of val_{c_p} and $val_{c_d} - val_{read}$.

Figure 3.7 shows how the values of c_p and c_d affect the threshold y, for constant val_{read}. It demonstrates that the Benefit$_{yes}$ equation has the desired property that as val_{c_d} increases for constant val_{c_p} a greater amount of trust in membership of c_p is required to grant the request, and as the right-hand back corner of the graph shows, not even a "fully" trusted principal may read a number which is of greater value than themselves. Conversely, a principal of much greater value than val_{c_d} needs a much smaller amount of trust to read the number, and if $val_{read} > val_{c_d}$ then only the most tenuous connection with a category is required to be granted the privilege. This leads to the conclusion that a good choice of value for val_{read} is the value of the category for which a principal should only need the smallest amount of belief in membership in order to be able to read it.

The Benefit$_{ask}$ equation shows that the choice of val_{time} is dependent on the choice of val_{read} since if $val_{time} \leq val_{read}$ then the possible benefit of being helpful always outweighs the cost of the time involved. The result is that the PDA will always ask the user if $b \geq d$, unless the answer is obviously "Yes".

From equation (3.3) it can be seen that if $(val_{time} - val_{read})$ is a constant, the amount of trust required in the membership of category c_p is inversely proportional to the value of the category. Accordingly, the lower the potential value of the other principal, the greater the trust required for it to be worth

disturbing the PDA owner. In practical terms, this indicates that a good choice of value for val_{time} is the value of the lowest category with which the user wishes to be consulted when a principal with strong membership (high value of $b-d$) of that category attempts to read a number of equal value.

3.4.3 Other Trusting Decisions

The cost-benefit analysis of the other trust-decisions shown in figure 3.5 will now be considered.

Request

The PDA owner enters the name of the person about whom they require information and the category under which they intend to file the data. The address book application now computes the expected benefit of asking each person within communication range and then polls each one in turn until one returns an answer. If the PDA is unable to obtain the number then it can store the request and ask other PDAs that it encounters in the future.

The expected benefit of asking Alice for a number depends on the relationship between Alice and the number. If the owner is looking for Alice's number then she is clearly the best person to ask, but if Bob knows Alice as a friend and he is looking for the number of a colleague then she is unlikely to be able to help. In access control terms this is represented as the ability of Alice to write to the category under which the owner intends to file the number and to link numbers with the principal that Bob is looking for. This is an analogous decision to the one made in section 3.4.1 when the PDA must make a decision based on the read permissions held by the other principal.

Therefore, assuming that Bob is looking for the number belonging to someone other than Alice, the expected benefit of asking Alice for a number is:

$$\text{Benefit}_{yes}(b-d,val_{c_p},val_{c_d}) = val_{c_p}.(b-d) - \max\left(val_{c_d}-val_{write},0\right)$$

c_p is a category which has write permission on the target category c_d supplied by the owner of the PDA and of which Alice is a member. val_{c_p} and val_{c_d} are the values of c_p and c_d respectively and hence this equation is analogous to the benefit equations used to determine whether a principal may read a number in a category c_d. val_{write}, like val_{read}, is a tunable parameter that represents how aggressively the PDA should search for a number. This parameter is much more dynamic than val_{read}: it can be influenced by the user when initiating a search via an "urgency" rating on the search screen, but it also takes environmental factors into account such as remaining battery power versus the cost of communication.

However this equation does not take into account that Alice may be associated with multiple categories c_p that have write permission on c_d, and since membership of one category may imply membership of another, summing the benefits for all c_p that can read c_d would lead to an inflated result. Using the category structure shown in figure 3.8 as an example, suppose Alice is a member of c_a with trust $(0.7, 0.1)$, and Charlie is a member of c'_{ch} with trust

$(0.8, 0.2)$. Since the strength of membership $(b{-}d)$ of their respective categories is equal and the value of c'_{ch} is greater than that of c_a (12 compared to 10) there should be greater benefit in asking Charlie than Alice. However, by default all members of c_a inherit the permissions of c'_a so the naïve solution would give the benefit of asking Alice as 15. Simply using the category with the greatest value would also lead to errors: if some weaker recommendations also place Charlie in category c_{ch} with trust $(0.2, 0.1)$, asking Alice would again be assigned a higher benefit than asking Charlie, despite the greater value of c_{ch}.

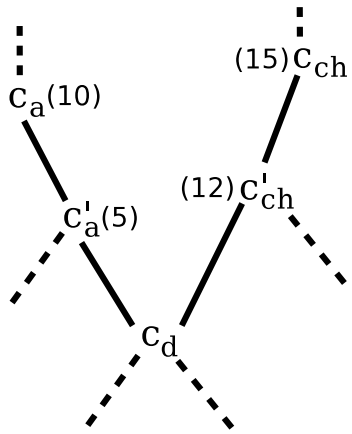


Figure 3.8: Part of a category hierarchy. Category values are shown in parentheses.

The solution employed is thus: for a particular traversal of the lattice, consider only the category which maximises the principal's expected benefit (that is, $val_c \times (b-d)$). Since a principal may appear in multiple paths through the category lattice (the previously mentioned example of colleagues who are also friends), define a set, C, of these categories. The benefit of asking Alice for the required number is therefore:

$$\sum_{c_p \in C} val_{c_p}.(b-d) - \max\left(val_{c_d} - val_{write}, 0\right)$$

Suppose Bob is searching for Charlie's number. If Charlie is within communication range, his PDA will ask Charlie's first, but if Charlie is unwilling to give his number to Bob (perhaps their PDAs have yet to be "introduced") or if Charlie is not contactable, then Bob's PDA will contact the other available PDAs in decreasing order of the expected benefit of asking them for the number, until the number is successfully obtained.

Push-Number

It is likely that Bob wishes to distribute his details to those people who would normally be able to read his number if they were to request it. Therefore, a number should only be pushed to those users who have read permission on the category in which it belongs: for each of the principals within communication

range the PDA evaluates the Benefit_{yes} function described in section 3.4.1, and if it is positive, the number is sent to that principal. The val_{read} variable may be used to determine how aggressively the PDA pushes its number to potential receivers.

Receive-Number

To avoid wasting resources, the decision as to whether to receive a number from another PDA should be taken as early as possible in the interaction. Unfortunately the current prototype implementation does not have access to the lower levels of the network stack and can only make a decision whether to accept a piece of data once it has been received. Since storage is cheap, the approach taken is that the PDA will accept all recommendations and compute their usefulness upon demand. This also means that should the PDA later obtain further recommendations that render the received recommendations more useful then they will be treated as such, instead of potentially being discarded when they are first received.

The size of each recommendation depends on the size of the cryptographic keys used in the implementation but, with a key length of 1024 bits, each recommendation would be approximately 550 bytes, plus any associated meta-data. With gigabyte flash memory storage units already appearing on the market, storing large numbers of these recommendations on a PDA is not a problem although this could leave the PDA vulnerable to a denial of service attack.

3.4.4 Deciding What to Display

There is one other operation, not shown in figure 3.5, where the trust model is invoked. This is in choosing what to display to the owner of the PDA when he or she wishes to view some information. Suppose Alice wishes to view Bob's number. She searches for his name and the PDA finds ten telephone numbers that are linked to him with varying degrees of strength. Since ten numbers will not fit onto the PDA display at one time, they are displayed in an order given by the product of the strength of the trust-model's belief they belong in a category (b–d), and the value of that category — the expected benefit of that number. The interface is designed to allow the user to give feedback on which number they were looking for and how successful they were at using it. This means that if Alice tries to use a number that is, for example out of date, she can click a button next to it and the system takes this to be a recommendation from her (which is implicitly highly trusted) that this number is not Bob's and updates its trust values accordingly.

Alternatively, Alice might browse entries by address book category. These could be ordered either conventionally (alphabetically), or by the degree of category membership. Again, the interface allows feedback for incorrect entries, in the form of extra recommendations.

3.5 Conclusions

This chapter outlined a framework for an unobtrusive and mostly automated security model for ubiquitous devices, using a system of trust-evaluated recommendations combined with an explicit risk analysis. This model is useful for a wide range of pervasive and ubiquitous computing applications, in which the user's time is a valuable resource and transparent interaction is needed wherever possible. The models have been tested by applying them to the prototypical examples of a phone book and appointment diary, and it is believed that they would be equally applicable to other ad hoc interactions with principals found in the owner's phone book — playing (electronic) games and sharing music, for example. However, it seems of limited practicality in applications that require interaction with principals unlikely to feature in the owner's address book, such as an e-purse [CSG+03]. Use in unattended environments may also be difficult as it is not clear whether paradoxical chains can automatically be resolved consistently with human intuition.

Another limitation of this recommendation-based trust model is that it makes no provision for the automatic detection of untrustworthy principals, such as those that disobey the requested dissemination policy for a user's phone number. It is possible for the PDA to spot some infringements by intermittently requesting its owner's personal information from other PDAs it encounters. Since information from third-parties may only be transferred by forwarding the original recommendations unchanged (see section 3.3), it is possible to examine the received recommendations for evidence of misbehaviour but this ad hoc detection cannot be considered to be an effective enforcement of the data-owner's policy.

This framework also has a limited cost and risk model since the focus was on usability, and the example application involves data with a poorly defined, and highly subjective, notion of "value". Later chapters will show how this model can be extended to more expressive application domains and explore suitable access control policies for autonomous trust- and risk-based decision-making frameworks.

CHAPTER 4

Risk in Global Computing

In the previous chapter, trust and risk were used to develop a novel access control model for information sharing in a ubiquitous computing scenario. This work highlighted the ability of trust to automate the making of low-risk security decisions. The importance of considering risk when assigning privilege was also highlighted: it was noted that a user may be more than happy to distribute a business card to complete strangers if it means free advertising for their business, but may be quite careful to whom they give their mobile phone number. Hence, a trust-based security paradigm must assess the risks involved in any operation as part of the process of making a decision — an area not explicitly considered in the existing trust systems described in chapter 2.

This chapter begins with a detailed exploration of the concept of risk, surveying usage and definition of the term in other fields, then using this research to define a risk model for use in global computing. This model is illustrated using two of the example applications from chapter 1. Finally, using the SECURE framework described in section 2.4 as a basis for the instantiation, a software architecture for SECURE is presented.

4.1 What is Risk?

John Adams gives a detailed analysis of the term **risk** in [Ada95], drawing upon all definitions and usages of the word. To disambiguate the various uses he defines the following two terms:

Risk is the probability of an adverse event in a stated time period. This is the definition of common parlance.

Detriment is the expected harm or loss associated with an adverse event. The integrated product of risk and harm is the **definition of risk** in the risk

and safety literature including, as will be seen later, the safety-critical programming field of computer science. This product is often expressed as a cost, such as pounds or loss in expected years.

Adams also states in [Ada95] that social scientists see risk as being an inherently subjective quantity — people see the world differently and react differently, so it is hard to measure objective risk because behaviour is modified in response to perceived risk. A classic example is when roads that are felt to be highly dangerous by local residents have statistically fewer accidents because people are more vigilant. However, in a computational decision-making framework, this distinction of *perceived* and *objective* risk is irrelevant. If risk can be perceived then, in a computational world, it must be quantifiable. The economist, Frank H. Knight [Kni21] arrived at a similar definition for risk, defining it as being a special type of *uncertainty*.

Risk applies to situations when one is unsure of the outcome, but the odds are known.

Uncertainty applies to situations when one is unsure of the outcome and the odds are unknown.

Within the insurance industry, risk is defined as being the potential variation from the expected outcome, which can be measured using the **standard deviation** of the outcome distribution [AB00].

Risk Models in Computer Science

Within the safety-critical programming industry, the following definition of risk seems to be widely, although not universally, accepted [Lev95]:

A **hazard** is defined as a state that together with other conditions in the environment will lead inevitably to an accident. It has two components:

1. severity (the worst possible accident that could result from the hazard given the environment in its unfavourable state);
2. likelihood of occurrence.

An accident is the result of a number of hazard states existing simultaneously. Therefore the risk of an accident is the probability of some combination of hazard states occurring concurrently. **Risk** is the *hazard* combined with:

1. the likelihood that the hazard will lead to an accident;
2. hazard exposure or duration (latency).

4.2 A Risk Model for Global Computing

Of the models and definitions described above, the insurance industry notion of risk seems the most appropriate for an access control system: when the system

grants privileges to a principal it is with the expectation that they will be used in a particular manner (for example, to update patient records to record the treatment that was administered), but there is also the possibility that the principal will deviate from this expected behaviour. The combined likelihood and severity (detriment in Adams' terms) of that variation is the risk of granting those privileges to the principal.

The proposed risk model is that each principal, before making a decision on whether to interact with another principal, analyses all the possible *outcomes* from that interaction and assesses the possible variation from the expected one. This will be done by calculating the likelihood and maximum potential cost or benefit[1] of each outcome.

This risk model builds upon the simple model developed for the PDA application described in chapter 3. That application had a weak notion of detriment, which is extended here to be modelled explicitly as monetary costs and benefits. The use of a tangible asset allows a market mechanism to be used to quantify preferences and priorities that may then be distilled into policy. This addresses one of the important open problems in many policy systems, namely the process of converting high-level strategic policy to low-level technical policy [BLMR04].

It is likely that in many applications, some outcomes will actually be distributed over a space that has a range of potential costs with corresponding probabilities. For example in a distributed file system, one outcome might be that part of a file is lost. However, depending on the data stored in the file, there may be a non-linear relationship between the cost and which blocks are lost. This could be modelled as one outcome for every possible combination of data blocks lost, but this would quickly lead to an unmanageably large number of very similar potential outcomes. The proposed solution is to allow each outcome to have a probabilistic variable cost, represented as a cost-Probability Density Function — a probability density function with cost on the x-axis, or *cost-PDF*.

4.2.1 Deriving Costs and Probabilities for an Outcome

For many application scenarios the outcomes and associated costs will be predetermined statically by the user, effectively encapsulating part of their security policy. For long-running systems, costs may change over time and new outcomes may be detected that were previously not expected. This is discussed further in section 4.2.3.

The objective theory of probability states that the likelihood of each outcome can be determined by dividing the number of occurrences of the outcome by the total number of events observed [Ram31]. However, what if the user has no previous observations to draw upon, or a change in context invalidates some or all of any previous observations?

It was noted above that risk is the quantification of uncertainty, and that when interacting with autonomous agents in a global computing environment

[1] For simplicity, "benefit" shall be defined as the negative side of the cost axis. All future references to cost may be taken to also include any potential benefit.

the risk lies with the uncertainty in the future behaviour of the other agents. Accordingly, Gambetta proposes the following definition in [Gam00]:

> trust (or, symmetrically, distrust) is a particular level of subjective probability with which an agent assesses that another agent (or group of agents) will perform a particular action, both *before* he can monitor such an action ... *and* in a context in which it affects *his own* action.

This suggests that the likelihood of each outcome should be a function of the trustworthiness of the requesting principal, and any past experiences of this situation. However, since past experiences are also used to calculate trust, the trust value already encapsulates the objective probability and so the likelihood shall depend only on the trustworthiness.

4.2.2 Combining Trust and Risk

To recap from chapter 2, in SECURE *trust* in a principal is computed by examining *evidence* (direct observations and third-party recommendation) relevant to the current context. The output of the trust calculator is t, a list of (t_i, c_i) pairs, where t_i is the trust-value assigned to p for trust-context c_i. The domain of a trust-value t_i in trust-context c_i is the lattice T_i over which there are two orderings defined, trust (indicating increasing/decreasing trustworthiness) and information (how much evidence was used to calculate the trust-value) [CNS03].

Figure 4.1 illustrates a user (the *decision-maker*) contemplating entering into parameterised interaction with a principal, p. For each possible outcome the decision-maker has a parameterised cost-PDF (that is, a family of cost-PDFs) which represent the range of possible costs and benefits that may be incurred by the user should this outcome occur. The name of the principal and any parameters associated with the request are passed to the trust-model which returns some trust-information t as described above. The function shown as $\star$ in Figure 4.1 then uses t to determine the values of the parameters of the cost-PDF and therefore selects which of the family of cost-PDFs is most appropriate in this situation.

For example, suppose t is a vector consisting of t_1 and t_2. For this particular outcome, the cost-PDF might be a Gaussian distribution with mean determined by t_1 and variance determined by $t_1 \oplus t_2$.

Once a cost-PDF for each outcome has been determined, the cost-PDFs for all the possible outcomes are combined and analysed according to the decision-maker's security policy — a process that will be described further in chapter 5 — and a decision made as to whether or not to go ahead with this interaction.

4.2.3 The Trust Life-Cycle and Trust-Contexts

The dashed lines in Figure 4.1 represent the second half of the "trust life-cycle." Once the decision has been made, some interaction may or may not take place and sometime later there will be feedback from that interaction, which is analysed by the $\diamond$ function and the trust-policy updated appropriately.

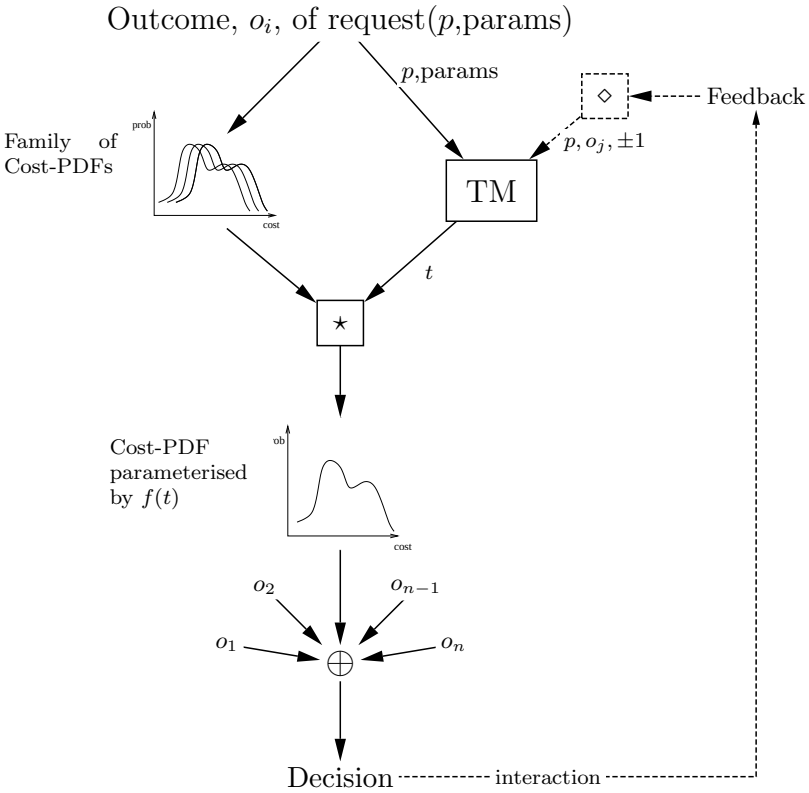


Figure 4.1: The interface between trust and risk.

In some applications the request and receipt of feedback are not only separated by a time delay but may not directly correspond at all. The outcome may not be observable, or might be a poor indicator of behaviour — in a distributed backup system, for example, it is better to test hosts' responses methodically than to rely on the presumably rare events that correspond to actual file recovery operations.

In order to function, both the $\star$ and $\diamond$ boxes in figure 4.1 require a mapping between outcomes and trust-contexts. The former uses such a mapping to correctly parameterise the cost-PDFs, the latter to update the principal's trust information in the correct trust-context in light of the actual outcome. As described in section 2.3.2, the SECURE project has chosen to unify the concepts of trust-contexts and outcomes by using an event structure of outcomes as trust-contexts. This neatly removes the need for the policy writer to laboriously define a trust-contexts/outcomes bijection, and facilitates the easy marriage of trust and risk components. An illustration of using event structures for outcomes is given in section 4.3.1. Using outcomes in this way also integrates well with the feedback system as the "Feedback" input shown in figure 4.1 becomes simply the outcome(s) that actually occur.

An important criterion for global computing is the ability to handle new situations — that is, the acknowledgement that all the information cannot be known in advance. A powerful feature of the event structure is that it is extensible, so new outcomes may be added as they become known, thus allowing the risk model to evolve in a similar manner to the way trust assessments change in response to principal behaviour. Cvrcek and Moody outline a framework for a model of *dynamic risk* in SECURE [CM05].

4.3 Application Examples

4.3.1 An e-Purse Scenario

An e-purse is a communications device that contains a smart-card on which electronic tokens with monetary value may be stored and used to pass this *e-cash* to other similarly equipped devices [SB02]. E-cash is useful in situations where the payments are too small for it to be cost-effective to use a credit-card and the use of real cash undesirable. Bus tickets, for example, usually cost much less than the smallest denomination of bank note, and having drivers deal with coins is inefficient as it takes time for them to find the passenger's change, plus there is the risk of robbery and loss.

Figure 4.2 illustrates how an e-purse could be used to pay for a bus journey. There are four trust-mediated decisions to be taken and the dashed line represents possible trust-information exchanged between the bus company and the bank. Each of these four decisions, and how they might be taken using the SECURE framework, will now be described in more detail.

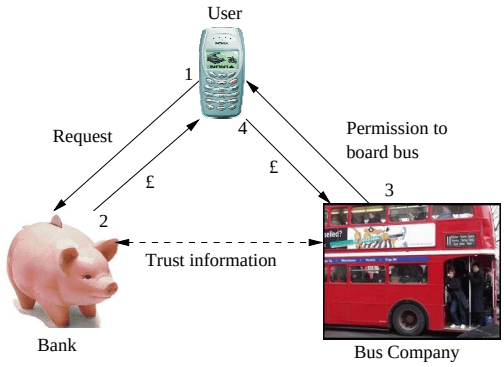


Figure 4.2: Overview of operation of e-Purse

1. *User request to buy e-cash from a bank.* The user contacts an agent of their bank and requests the transfer of e-cash to their mobile phone. It is envisaged that the user would prefer to deal with their own bank directly, but may be forced to deal with an unknown agent, for example if they are in a foreign country. Possible outcomes are:

   - The bank does not trust the user sufficiently to send them any e-cash and the transfer is refused.

- The bank sends the requested amount of e-cash.
- The bank sends a different amount of e-cash.
- The bank sends forged[2] e-cash.
- The bank deducts the correct amount of money from the user's account.
- The bank deducts an incorrect amount of money from the user's account.

2. *Bank agent sends e-cash to the user*. The bank has to decide how much e-cash to transfer to the user. Possible outcomes are:

   - User receives their cash and spends it.
   - The user is not the owner of the account to be accessed, bank is forced to cover the real owner's loss.

3. *User requests permission to board bus*. Many public transportation systems use a zone-based charging system, where journeys between zones incur a higher charge than within a zone. To automate the charging process, the bus registers the user when they board, and charges the passenger's e-purse when they disembark. When a person attempts to board a bus, their e-purse registers with the bus e-purse via a bluetooth connection and the bus must then make a decision as to whether to allow them to board, based on factors such as the maximum cost of a journey on the bus's current route. Possible outcomes are:

   - User pays the correct amount when they disembark the bus.
   - User has insufficient funds to cover the cost of the journey.
   - User pays with forged e-cash that is rejected by the bank when the bus company tries to redeem it at the end of the day.

4. *Bus requests payment for a ticket*. Possible outcomes are:

   - The bus requests the correct amount for the journey.
   - The bus overcharges for journey taken.
   - Something other than the bus the user is on is trying to charge the user's e-purse, or the user is not actually on the bus but is perhaps just near a bus.

It is noted that for decisions 1 and 2 the model is similar to that already employed by international credit card networks, such as VISA. The level of verification (that is, trust required) is dependent on the risk associated with the transaction. A more novel use of the SECURE model is for decision 3, which will now be discussed in more detail.

[2] *i.e.*, e-cash that appears genuine under the verification procedure available to e-purses, but is rejected when redeemed at a bank

Boarding the Bus

Factors that will influence the bus's decision to let a person aboard are:

- user's history with this bus company;
- recommendations from banks;
- recommendations from other bus companies;
- maximum cost of ticket on current route.

The costs and benefits of each outcome are related to the fare the passenger is expected to pay. It is assumed that for a particular context (such as time, day of week, location), the number of zones travelled by a passenger in a single journey is randomly and independently distributed. This can be modelled using a Poisson distribution, $P(\lambda)$, where λ is the average number of zones travelled through on a single journey.

$$P(\lambda) : f(x) = \frac{e^{-\lambda}\lambda^x}{x!}$$

This leads to the following costs and benefits for each outcome, expressed as probability-density functions.

- *User pays the correct amount when they disembark the bus.* Benefit = $P(\mathit{fare}(\lambda))$ where *fare(n)* is the function that determines the fare for travelling through n zones.
- *User has insufficient funds to cover the cost of the journey.* If this is an honest mistake then the amount of money that the passenger is short of the fare will be a random number between one pence and the full fare.[3] Therefore, the cost and benefit will be uniformly distributed between the full fare (minus one pence) and the cost of transporting a non-paying passenger; that is between $P(\mathit{fare}(\lambda))$ and $P(\mathit{cost}(\lambda))$ where *cost(n)* is the function that determines the cost-price of a fare for travelling n number of zones. This is illustrated in figure 4.3.
- *User pays with forged e-cash that is rejected by the bank when the bus company tries to redeem it at the end of the day.* Cost = $P(\mathit{cost}(\lambda))$, where *cost(n)* is the same function as in the previous outcome.

In all three outcomes, λ can either be based on any history about the passenger stored in the trust model, or statistically determined if the person is unknown. In either case the system would take into account contextual information such as time-of-day.

[3] A dishonest passenger could attempt to persistently avoid paying the fare by always having too little e-cash in their e-purse, but an intelligent adversary would arrange to always pay small but random amounts to avoid creating too much suspicion.

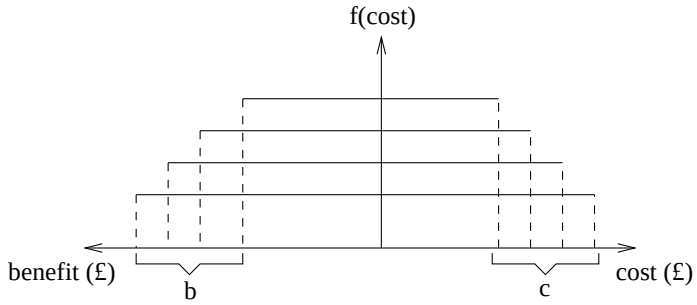


Figure 4.3: The family of cost probability density functions for the outcome that the passenger has insufficient funds to pay the fare. Both b and c are Poisson-distributed by the length of the journey.

Integration with SECURE Trust Model

An example event structure for the e-purse scenario is given in [CDKN04] and expanded in figure 4.4 to include the additional outcomes considered in this example. The following outcomes are possible:

- Nothing may have been observed ($\emptyset$);
- *reject* may have been observed;
- *accept* may have been observed, but it is not yet known whether this is of subtype *forged*, *insufficient funds*, or *paid in full*;
- *forged* may have been observed (implying observation of *accept* as well);
- *insufficient funds* may have been observed (implying observation of *accept*);
- *paid in full* may have been observed (implying observation of *accept*);

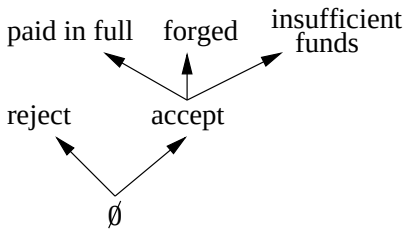


Figure 4.4: Example of an event structure of outcomes for the e-purse scenario.

The SECURE trust-engine can compute a trust-value of the form, (s, i, c) for each of the possible outcomes, where s is the number of supporting observations, i is the number of inconclusive observations and c is the number of contradictory observations. Given this information, it is trivial to compute a probability for each of the possible outcomes, as required by this risk model.

4.3.2 Spam Filtering

A second on-going example application used in the SECURE project is that of detecting unwanted email ("spam"). This uses a simpler risk model than the e-purse scenario, thereby making it easier to evaluate other aspects of the SECURE framework. The SECURE project has developed a number of different applications for detecting spam; the risk model used in all of them is described below.[4]

Requests and Responses: There is one possible request, namely to accept an email. The decision-maker can either "pass" or "mark" messages. Passed messages are delivered normally. Marked messages are taken out of the normal stream, for example they may be placed in a special "suspected spam" folder, which is checked less frequently.

Outcomes: The message might be real or it might be spam, so there are two outcomes. The user of the system will eventually discover which occurred in either case, thereby providing feedback. The event structure of these outcomes is shown in figure 4.5. There are two cost-PDFs representing the two possible outcomes, Spam and NotSpam, and each must be able to encode the possibility of the email being marked or passed, as shown in figure 4.6. Note that since the costs chosen below are easy to model using discrete values, cost-probability-functions have been used here, rather than probability density functions.

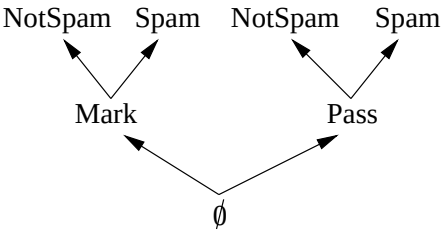


Figure 4.5: Event Structure for spam filtering.

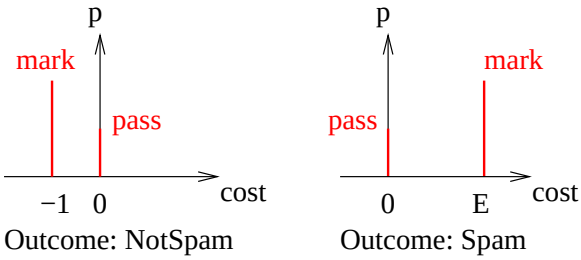


Figure 4.6: Probability cost-functions for spam filtering.

Costs

Costs are expressed relative to what would be incurred without the spam filter in place since this avoids difficulties in assessing exactly how much email is "worth." Table 4.1 is the cost matrix for the two outcomes with respect to each of the two members of the decision set.

Note that passing a message always costs zero, since that is what would happen without the filter in place. Marking a message provides a benefit (cost

[4]This application example risk model was developed in collaboration with David Ingram and published in [DM04] and [DBE+03].

	Pass	Mark
Spam message	0	-1
Real message	0	E

Table 4.1: Outcome cost matrix for the spam filtering application

of -1) if it is spam, equivalent to the time saved and the value of not interrupting the user. This is arbitrarily set to be the unit of cost in this application. Marking a real message has a positive cost of E (the false-positive error cost). E is likely to be considerably more than 1, and is configured by the user based on the average severity of the consequences of missing a valid email relative to the cost of their time.

Trust Values and Outcome Likelihoods

There are a number of ways to determine the likelihood of the outcomes in this scenario. The simplest method is based on the identity of the sender of the email — effectively asking the trust engine how much the sender is trusted not to be a spammer. An event structure for this is shown in figure 4.5. However, a lack of authentication in the Simple Mail Transfer Protocol (SMTP) [Kle01] means that this approach is not particularly effective in practice and alternative methods will be discussed in chapter 7.

4.4 The SECURE Architecture

Figure 2.2 gave an outline of the SECURE framework. The interaction between trust and risk within the framework was shown in more detail in figure 4.1; how this integrates with the rest of the framework is shown in figure 4.7.

Interactions are initiated by a *request* relating to a principal p being submitted to the Request Analyser. The entity recognition module then determines whether p has been seen before, and how much trust there is in the recognition process used to authenticate p, therefore translating p into a Recognised Principal (RP). The request is then submitted to the Access Control Manager, which invokes the trust and risk components to determine the trustworthiness of RP and the risk associated with the request. The Trust Calculator computes the trustworthiness (T_v) of RP according to the trust policy [CNS03] and returns that value to the Access Control Manager. The Risk Evaluator determines the appropriate family of cost-PDFs for the request (a "risk metric"), as described earlier in this chapter.

The Access Control Manager combines the trust value and risk metric (shown as $\star$ in figure 4.1) and evaluates the user's risk-based security policy to determine whether the request should be granted — the $\oplus$ function in figure 4.1. A key benefit of the SECURE approach is that the explicit modelling of uncertainty is carried all the way through the decision-making process, facilitating the making of decisions based on metrics of the quantity and quality of information available, rather than just the information itself. In chapter 3

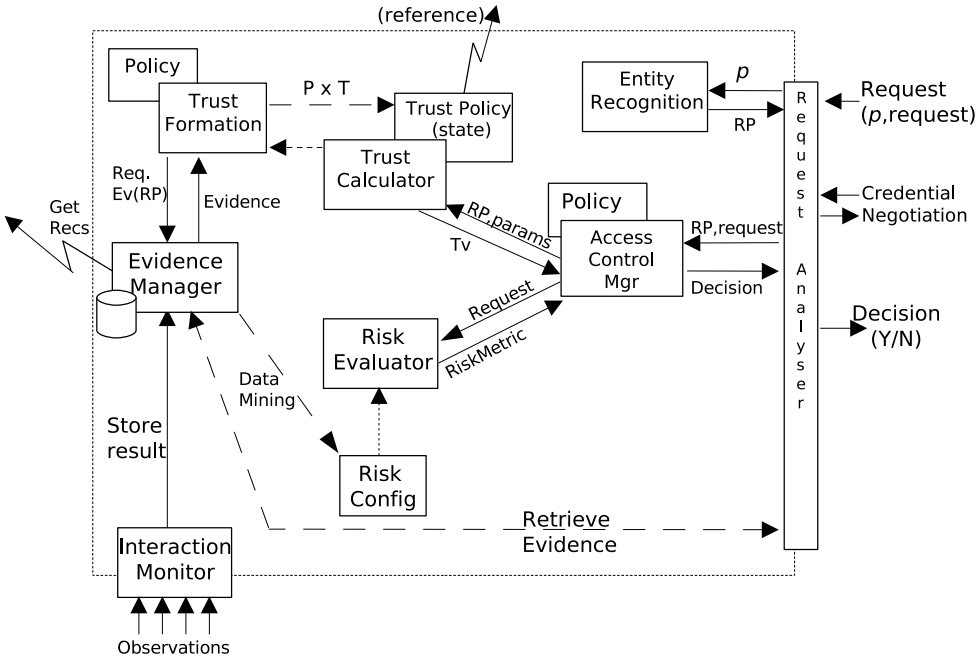


Figure 4.7: A Software Architecture for SECURE

it was shown that, as a result of this uncertainty, a richer set of possible responses was required than simply "yes" or "no" and therefore a requirement of the SECURE security policy language is that it must be possible to return a set of possible responses. For example, one of these responses may be that more information is required, perhaps in the form of additional credentials that might lead to a phase of credential discovery or negotiation by the Request Analyser. For a variety of reasons, such as privacy or the principle of least privilege, p may not have submitted all of the credentials he or she holds in the initial request, but may be willing to do so if the the decision-maker can produce some verification of their own authenticity. To ensure the generality of the architecture, all decisions can be mapped to a binary yes/no value in the event that the calling application does not understand the semantics of the actual decision.

Once an interaction has completed, the Interaction Monitor observes the outcome and gathers any other relevant feedback before submitting it to the Evidence Manager (the function represented by $\diamond$ in figure 4.1). The Evidence Manager is responsible for the collection, aggregation and storage of evidence concerning other principals, in the form of direct observations or recommendations from others. The Evidence Manager may also issue recommendations about principals for use by others, but since there are also privacy implications in doing so, access is controlled using SECURE. A request from another principal for trust information about a third party is evaluated using the Access Control manager in the same way as any other request, and the Request Analyser then only retrieves the requested evidence from the Evidence Manager if the decision is positive. Access control to the evidence store will be discussed

further in section 6.7.2.

There are two other major components in figure 4.7, Risk Configurator and Trust Formation, that have not yet been described. These are not normally invoked while evaluating a request, but perform background processing on the evidence to provide pre-computed information which the other components then use during evaluation. The Risk Configurator performs data mining on the evidence store to update the risk information, as described in section 4.2.3. The Trust Formation component (also known as the Trust Life-cycle Manager) is responsible for instantiating and updating *trust policy*. Where a trust policy contains a reference to a principal who is currently unreachable, the Trust Formation module may also be able to provide a cached value. The Trust Formation module may be invoked at a variety of times, for example when sufficient new evidence has accumulated in the evidence store, when access control policy calls for trust values more recent than those cached in the trust policy, or as a background process whenever the system has resources available.

4.5 Conclusions

This chapter has reviewed the concept of **risk** in a variety of fields, including computer science and the insurance industry. The results of this review lead to the development of a formal and explicit model of risk for use in securing global computing applications which was subsequently illustrated using the examples of spam detection and an e-purse scenario. A software architecture for the SECURE framework was then presented, based on, and extending, the work of the SECURE project consortium.

The next chapter will build upon the risk model by considering the requirements for specifying and enforcing security policy in the SECURE Access Control manager.

CHAPTER 5

Policy Languages for Trust-Based Access Control

The previous chapter defined a model of risk for use in global computing, and in particular trust-based access control. In contrast to trust, which may be considered to be globally computed, risk evaluation is a local process in which a principal must determine their personal exposure that will result from trusting another principal. In this way, a principal may make a local evaluation of global information ("trust") to ensure that the views of others cannot contravene their own security policy, or force them to take a decision that they would not otherwise wish to make.

This chapter will study the problem of writing and evaluating security policies incorporating reasoning about trust and risk; that is, the rôle of the Access Control (AC) manager in the SECURE architecture (figure 4.7).

5.1 Making Trust-based Decisions

Intuitively a high-risk action requires greater trust in its participants, and the lower the risk the less worthwhile it is to expend resources to establish a high level of trust. The majority of computational trust systems, such as [ARH00] and [AD01], concentrate on aspects relating to assigning a trust-value to a principal; they do not consider policy-driven decision-making using trust.

[YS02] and [XL02] make use of thresholding in their policy languages — the former checks that the trust-value is greater than a scalar, while in the latter a principal must have performed at least a certain number of interactions as well as having a trust value greater than a predefined level. However, all of these thresholds are statically determined by the policy author and there is no run-time evaluation of risk.

In SECURE, an explicit cost-benefit analysis is used to determine how much trust is required to offset any given risk. This can take two possible forms:

Risk-based thresholding: the AC manager checks rules that compare a trust-value to a cost-metric. This is similar to [YS02], except that the threshold is dynamically set by the level of risk, so the greater the risk, the greater the trust required. This was also the approach taken by the application in chapter 3.

Risk metrics: the AC manager checks that the risk of the request does not exceed thresholds determined by the policy author. As noted in section 4.1, risk can be measured by a number of statistical measures such as expectation, variance and minimum/maximum.

Another requirement for the policy language is the ability to choose from a number of possible responses, such as request additional credentials, or (in an interactive system) ask the user for guidance. These responses must be parameterisable to allow the system to communicate, for instance, what additional credentials are required.

5.2 Trust-reasoning in the OASIS Policy Language

Both OASIS and the SECURE AC manager have the concept of principals making requests which are then granted or denied. A logical starting point for a policy language was therefore to investigate the use of the OASIS system as the SECURE AC Manager. This would also facilitate the incorporation of trust-based reasoning into existing applications that use rôle-based access control.[1]

In OASIS, authorisation rules are evaluated to determine whether a principal has the right to use a particular service. SECURE's semantics are much richer, allowing a principal to make a *request* for which there can be a number of *responses*. This could be encapsulated into OASIS by making each possible response the subject of an authorisation rule:

$$\begin{aligned} r, e_1, ..., e_{n_e} &\vdash response_1(params, ...) \\ r', e'_1, ..., e'_{n_e} &\vdash response_2(params, ...) \\ r'', e''_1, ..., e''_{n_e} &\vdash response_3(params, ...) \end{aligned}$$

However, this requires additional logic in the request analyser to split the request into separate invocations of the AC manager, one for each possible response. By using the OASIS type system to encapsulate requests as an action type, submitted as a parameter to a generic *request* privilege, the logic can be

[1] The OASIS-based TBAC model described in this section was developed in collaboration with Jean Bacon, András Belokosztolszki, David Eyers and Ken Moody, and published in a paper at the Ninth Symposium on Access Control Models and Technologies (SACMAT04) [DBE+04].

kept entirely within the OASIS model. The template for an authorisation rule would then be:

$$r, e_1, ..., e_{n_e} \vdash request(principal : String, act : Action, resp? : Response)$$

Sub-types are used to distinguish between different sorts of action, as shown in figure 5.1. Actions are in turn parameterised with request-specific information, for example, filename for a "get_file" action. Depending on the action type, OASIS will use different functions to access these parameters. In this section capitalised subwords are used to denote types (for example, "FileAction") and lower case words separated with underscores for parameter names (e.g. "file_action").

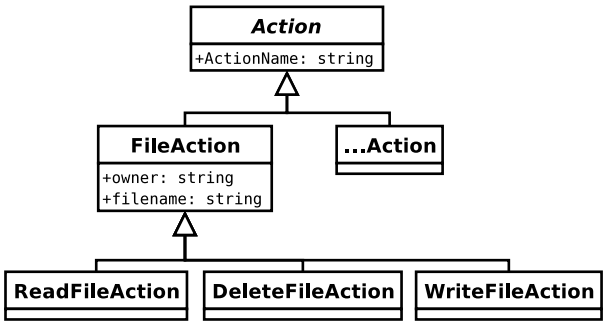


Figure 5.1: The Action type and its sub-types.

The following predicates can be used to access parameters of the subtypes shown in figure 5.1:

$$getActionName(act : Action, action\_name? : String)$$
$$getFileActionOwner(act : FileAction, owner? : String)$$
$$getFileActionFilename(act : FileAction, file\_name? : String)$$

5.2.1 OASIS Environmental Predicate Interfaces to SECURE

Due to the openness of the OASIS environmental predicate call-out mechanism, it is straightforward to define the interfaces required to allow trust and cost/benefit computations to be included within OASIS policy rules. How an OASIS-based Access Control manager would integrate with the SECURE framework is shown in figure 5.2. Initial investigations suggested that three main predicates were required:

Trust retrieval: The first predicate uses the contexts[2] associated with the action to retrieve relevant trust values into rule parameters for a named principal:

$$trust(principal : String, context : String, value? : TrustValue)$$

[2]NB: The model in this section predates, and to some extent motivated, the unification of outcomes and contexts as event structures.

Cost retrieval: The second predicate is one which retrieves the set of outcome costs into rule parameters for a named action:

$$cost(action : String, outcome : String, cost? : Number)$$

Risk thresholding: This third type of predicate fails if the trust is too low for the outcome's cost. This failure is in the same sense as any other OASIS rule predicate returning false — if using OASIS semantics in which all routes are explored to reach the potential target rôle or privilege, such failure will cause backtracking to attempt other potential activation or authorisation rules. Further details of these "risk" predicates are given below.

5.2.2 Risk Thresholding Predicates

Although the OASIS policy language is not sufficiently expressive to encode all the policies required in SECURE, the ability to call external services via environmental predicates can be used to extend the OASIS model to provide the required functionality. Initially it was thought to try and support both the risk-based thresholding and the risk metric style policies described in section 5.1. Therefore the following grammar for writing risk predicates was proposed:

```
PREDICATE ::= CONDITION
            | if CONDITION then PREDICATE else PREDICATE endif
CONDITION ::= CONDITION CONDOP COND2
COND2     ::= EXPR | COND2 CONDOP2 EXPR
EXPR      ::= true | false | (CONDITION)
            | ARITHEX EXPROP ARITHEX
ARITHEX   ::= ARITHEX2 | ARITHEX ARITHOP ARITHEX2
ARITHEX2  ::= ARITHEX3 | ARITHEX2 ARITHOP2 ARITHEX3
ARITHEX3  ::= VALUE | exp(ARITHEX) | (ARITHEX)
VALUE     ::= tval | cost | id | NUM | -VALUE

CONDOP    ::= ||
CONDOP2   ::= &&
EXPROP    ::= == | != | > | < | =< | >=
ARITHOP   ::= + | -
ARITHOP2  ::= * | /
NUM       ::= int | float
```

The standard definitions of the numeric datatypes int and float are assumed. `id` is an identifier (variable name) and `exp(ARITHEX)` is the exponential function e^x where x may be another arithmetic expression. Operator binding and precedence is based on ANSI C [ISO99].

5.2.3 Policies for Spam Detection

This model of access control will now be illustrated using the application of spam detection. Based on the risk analysis described in the previous chap-

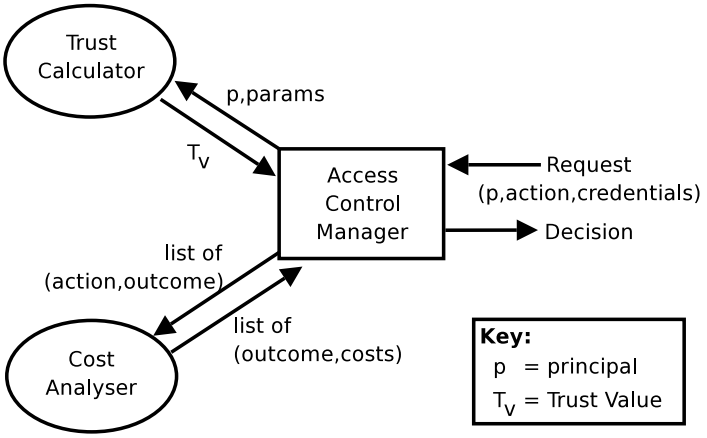


Figure 5.2: Using OASIS as the SECURE Access Control manager.

ter (section 4.3.2), the expected cost of marking the message as spam is:

$$P \times E + (-1)(1 - P) = P \times (E + 1) - 1$$

where P is the probability that the message is spam. In this application, P might be determined as follows.

- *trust* is the probability the sender is a spammer, based on past interactions, or reported by discounted recommendations.
- *history* is the number of pieces of evidence (observations and recommendations) the trust value is based on.
- *conf* = ER confidence; a value in the interval $[0, 1)$ based on <from, route>.
- *bayes* = content-based real email probability; based on <subject, body>.

A straightforward way to combine two probability values is with a weighting, as follows:

- estimated probability(real email) = $q.trust + (1 - q).bayes$;
- q depends on *history*, *conf*, and possibly a user-specified weighting.

The required characteristics are as follows:

$$\begin{aligned}
&\text{As } history \to 0, q \to 0 \\
&\text{As } conf \to 0, q \to 0 \\
&\text{As } (history \to \infty \textbf{ and } conf \to 1), q \to 1
\end{aligned}$$

The most natural way to achieve this is to make q proportional to *conf*, and also to some negative exponential function of *history*, for example:

$$q = conf.(1 - e^{-history/H})$$

The table below shows the relationship between *history* and $q/conf$, and the effect of the scaling constant H:

history	*q*/*conf*
0	0
H	$1 - 1/e \approx 0.63$
$2H$	$1 - 1/e^2 \approx 0.86$
∞	1

Therefore, H is the number of pieces of evidence for the trust engine to represent 63% of the weighting relative to content-based analysis and assuming perfect entity recognition (proportionally less otherwise). This can be tuned by a system administrator and informal analysis suggests that a value of two or three would be appropriate in this scenario since few emails need to be received to determine whether the sender is a spammer or not.

The Bayesian analysis has the greatest effect when we have absolutely no trust information or if the entity recognition confidence is not very high (quite likely since email headers can be forged easily). Hence:

$$\begin{aligned} P &= \text{estimated probability(real mail)} \\ &= trust.conf.(1 - e^{-history/H}) + bayes.(1 - conf.(1 - e^{-history/H})) \end{aligned}$$

Access control policy

Suppose the policy is to mark the message if the expected cost is negative, that is, if $P \times (E + 1) < 1$. The email is presented to the Access Control manager as:

$$request(sender, email\_action, response)$$

The following rule is then used to determine whether to mark it as spam:

$$\begin{aligned} trust(sender, \text{'prob'}, prob), \\ trust(sender, \text{'history'}, history), \\ trust(sender, \text{'identification'}, conf), \\ bayes\_analysis(email, bayes), \\ cost(email, \text{'mark\_real\_msg'}, E), \\ risk\_mark(prob, history, conf, bayes, E) \quad \vdash request(sender, email\_action, \text{'mark'}) \end{aligned}$$

If request is granted then the email is marked as spam. The *bayes_analysis* predicate is just an environmental predicate call to an external Bayesian analysis service; the *risk_mark* predicate is defined as:

```
( prob*conf*(1-exp(-history/H))
+ bayes*(1-conf*(1-exp(-history/H))) ) * (E+1) < 1
```

5.2.4 More Complex Policies: The Grid

While recent years have seen tremendous performance advances in micro-computers, there are still some science, engineering and business applications that require specialised resources. Such specialised resources are usually beyond the means of infrequent users of these applications. One solution has been to

attempt to create something akin to a "power grid" of computational resources — the user "plugging-in" to this grid may access a whole range of services not available at their current location, possibly in return for a fee [App02].

As the Internet has shown, exposing computers to the entire world makes for some tricky problems — how does a user know which server to trust with their computation, or which server has the correct version of the document they want to retrieve. Likewise, the server must decide whether this user is a genuine customer, if they are likely to pay for the work carried out, or if their program will use more resources than the server is willing to allocate. Linking the user to a credit card account may mitigate some of these problems, but this is inappropriate for small transactions or pro-bono agreements (for example, where there exist reciprocal agreements between organisations to allow them to use each others' computing resources).

A file-storage and publication service for the Grid

To further demonstrate the utility of this model, it has been applied to a Grid file-storage and publication service. There are a number of possible use-cases.

- Alice is going on a long research trip, and wishes to ensure some data she needs is widely available, not only for herself, but to potential collaborators she may meet along the way.
- Bob has a very large data set he needs processed by a grid computation server. To save network cost, he wants to host the data as close to the compute server as he can.
- Charlie has written a very controversial paper he is expecting many people will wish to download. His local sysadmin has told him that their webserver is likely to be overwhelmed, and Charlie is also worried that people who object to his paper may try to block its distribution.

Each server offering this service has different properties. Apart from their fee, they differ in bandwidth available, uptime guarantee, storage integrity and the strictness with which they check access control credentials. These are all represented in the trust framework via contexts. Correspondingly, the user must specify the importance of the file in each of these categories. Alice will wish to use several servers that are geographically distributed with average availability and integrity, but are known to be very trustworthy when it comes to not allowing unauthorised people to access stored files. In contrast, Bob needs only one server (so can pay a little more for it), but he needs very good availability and storage integrity. Finally, Charlie wants to publish his paper widely on a large number of cheap servers which are trusted for their high level of integrity.

The Grid hosts also use SECURE to make decisions about which clients they can trust. Since there is a certain amount of anonymity on the Internet, there is always the danger that a service vendor will not receive payment, or that they will publish a file that results in the server being overwhelmed or leading to legal difficulties. Of course, there are also services that may want

very popular files (if they have a good connections and an appropriate charging scheme in place) and even ones that actively look to host information that has been censored in other legal jurisdictions (for example, `http://cryptome.org`).

The advantage of SECURE here is that while a Grid host may have never personally dealt with a potential client before, the trust framework allows the decision-maker to consider the reported experiences of its peers. This nicely models human communities in which persons who repeatedly act in an anti-social manner are eventually rejected by the community, thus giving principals an incentive to behave well.

File retrieval and deletion

Trust-based reasoning may also be used for determining who should be able to retrieve or delete the shared files. During her travels, Alice meets David at a conference and wants to give him access to a copy of her slides. Rather than having to log into all the servers where the files are stored to update the access control policy, she gives him a signed recommendation that any Grid host containing her files should permit him to download the file, `slides.pdf`.

David returns home from the conference and contacts a server that Alice told him contains the file. Retrieving files does not require payment, so he submits a request to read Alice's file, along with the recommendation she gave him. The server weighs up the strength of Alice's recommendation and its confidence in David's authenticity against the importance Alice has indicated she places in the confidentiality of the file. For a public presentation, only a low confidence in David's authentication as being the David referred to in the recommendation is needed. In contrast, for a presentation intended only for project partners, the server may decide the file is sufficiently important to expend resources on further authentication, such as a challenge-response exchange.

A similar process is used when a server is asked to delete a file, but in addition to checking the authorisation and identity of the requester, it also examines their *carefulness* trust-context to check how much the user themselves can be trusted when issuing commands of serious consequence, such as delete! This models the concept of trust in oneself.

Full details of the possible actions, their outcomes, relevant trust-contexts and format of trust-values are shown in table 5.1. Most of the trust-values used are self-explanatory, but for some contexts (*belief*, *disbelief*) pairs are used which summarise the weight of evidence for and against a particular trust-assignment, with $belief + disbelief \leq 1$. This is similar to the format of trust-values used in the application in chapter 3. A further complication of the *honesty* trust-context is that it was observed that a person's ability and willingness to pay might be dependent on the amount of money involved. For example, a user might be known to be willing to pay small amounts but tends to renege on the deal if the amount is large, or a credit-card company might recommend that someone is able to pay up to £100, but no more. To model this, the space of possible payments is divided into intervals and a (*belief*, *disbelief*) pair computed for the relevant one. For example, in the prototype implementation (see section 5.2.5) the intervals $[0, 10), [10, 30), [30, 60), [60, \infty)$ are used.

Action	Outcomes	Trust Context	Trust-value
FileAction (client publishes)	(un)available ignore AC policy too strict on authorisation file is corrupted or deleted	availability confidentiality confidentiality integrity identification	% uptime, no. of days measured no. of incorrect grants, no. of incorrect denials, total requests (as above) no. of corrupted/deleted files, no. of files checked % confidence
WriteFileAction (server)	client does not pay popular file (overloaded) controversial file (legal difficulties)	honesty popularity controversialness identification	(belief,disbelief) avg. bandwidth consumption, no. of files evaluated no. of controversial files, no. of files hosted % confidence
ReadFileAction	(un)authorised	authorisation identification	(belief,disbelief) % confidence
WriteFileAction	(un)authorised (un)intended	authorisation carefulness identification	(belief,disbelief) no. of mistakes, no. of operations % confidence

Table 5.1: Outcomes for a Grid file storage and publication service.

A publish-file policy in OASIS

For each server which offers a publication service within her budget, Alice submits the request:

$$request(server\_id, file\_action, response)$$

The *file_action* object (of type "FileAction") is parameterised with the name of the file to be published, and Alice's name. Alice has specified in the file's meta-data that high file availability is important to her, but she intends to achieve that via replication, so she only attributes medium importance to the individual servers' uptime and integrity. The file does not contain highly confidential data so Alice is able to assign a low importance to the server giving the file away indiscriminately. Further, she needs to be able to retrieve the file easily and so she is anxious not to host it with servers that are over-zealous in enforcing access control policy therefore assigning low importance to the enforcement aspect. All of this information is made available to the AC manager via the SECURE cost analyser.

The OASIS policy is then used to determine whether this server is a suitable host for her files as shown in figure 5.3, where the costs are computed from the information supplied by Alice as to the importance of her file.

$$\begin{aligned} trust(server\_id, \text{'availability'}, t_1), \\ trust(server\_id, \text{'confidentiality'}, t_2), \\ trust(server\_id, \text{'integrity'}, t_3), \\ cost(file\_action, \text{'unavailable'}, cost_1), \\ cost(file\_action, \text{'ignore\_AC'}, cost_2), \\ cost(file\_action, \text{'too\_strict'}, cost_3), \\ cost(file\_action, \text{'lost'}, cost_4), \\ risk\_unavailable(t_1, cost_1), \\ risk\_ignoreAC(t_2, cost_2), \\ risk\_tooStrict(t_2, cost_3), \\ risk\_lost(t_3, cost_4) \quad \vdash request(server\_id, file\_action, \text{'host'}) \end{aligned}$$

Figure 5.3: The OASIS policy used to determine whether *server_id* is a suitable server for hosting Alice's files.

A retrieve-file policy in OASIS

To retrieve the file Alice has given him permission to download, David submits the following request where the *read_file_action* is parameterised with (Alice, slides.pdf). The OASIS policy the server uses to determine whether he may read the file is shown in figures 5.4 and 5.5.

$$request(David, read\_file\_action)$$

The costs Alice assigned to her file are given in section 5.2.4, from which we can deduce that provided the server is over 60% confident that is it "David" it is interacting with, since $cost_1$ is "low", the server must just have more belief than disbelief that David is authorised to read the file in order to grant him access to it. Since David has supplied a recommendation from Alice strongly recommending that he be able to read the file, unless the server has further information to the contrary (perhaps another recommendation from Alice with a later time-stamp revoking that authorisation) it will therefore allow him to read the file.

$$
\begin{array}{l}
trust(principal, \text{'authorised'}, t_1), \\
trust(principal, \text{'identification'}, t_{id}), \\
getFileActionName(read\_file\_action, file\_name), \\
cost(file\_name, \text{'authorised'}, cost_1), \\
cost(file\_name, \text{'unauthorised'}, cost_2), \\
read\_file\_risk(t_1, t_{id}, cost_1, cost_2) \ \vdash \ request(principal, read\_file\_action)
\end{array}
$$

Figure 5.4: The OASIS policy used by the server to determine whether a principal may access a file.

```
if trust_id < 0.6 then
  false
else
  if cost1 == low then
    t1.belief - t1.disbelief > 0
  else
    if cost1 == high then
      if cost2 == high then
         t1.belief - t1.disbelief > 0.6
      else
         t1.belief - t1.disbelief > 0.7
      endif
    else
       if cost2 == high then
          t1.belief - t1.disbelief > 0.5
       else
         t1.belief - t1.disbelief > 0.6
       endif
    endif
  endif
endif
```

Figure 5.5: Definition of the *read_file_risk* predicate.

Trust reasoning in rôle-activation

The ability for Alice to make recommendations about rôles, as well as individuals demonstrates the expressiveness gained by using the OASIS infrastructure. For instance, she might recommend that all members of the University of Cambridge Computer Laboratory can retrieve her presentation, but members of the University of Oxford may not read any of her files. Example policies using rôles more extensively are given in section 5.3.5.

5.2.5 Implementation

To check the viability of these ideas, the SECURE trust predicate interface was integrated into the Prolog OASIS implementation. For example, translating OASIS XML policy to check whether a user is willing to store a given file on a particular server, the following Prolog OASIS is obtained:

```
privilegeRequest(fileAction_store,[ServerID,FileName]):-
        % check policy parameter modes
        nonvar(ServerID),nonvar(FileName),
        % check prerequisites
        trust(ServerID,availability,T_1),
        trust(ServerID,confidentiality,T_2),
        trust(ServerID,integrity,T_3),
        trust(ServerID,reliability,T_4),
        cost(FileName,unavailable,Cost_1),
        cost(FileName,ignore_AC,Cost_2),
        cost(FileName,too_strict,Cost_3),
        cost(FileName,lost,Cost_4),
        risk(unavailable,T_1,Cost_1),
        risk(ignoreAC,T_2,Cost_2),
        risk(tooStrict,T_3,Cost_3),
        risk(lost,T_4,Cost_4).
```

Note that the above is a particular instance of a FileAction, with a set parameter structure (server ID followed by the filename) — this is a consequence of the Prolog translation; in fact the XML policy format allows more flexibility than this when specifying the binding of parameters within rules.

The `risk` predicates are encoded into Prolog from the risk expression syntax given on page 74. This translation is done using the Definite Clause Grammar (DCG) extensions of SWI-Prolog (DCGs are not part of ISO Prolog, but commonly appear nonetheless).

A number of manual privilege requests were issued to check that the framework was producing the expected results. Advancing beyond this proof of concept level requires the completion of the SECURE framework, which was revised before this took place. The development of an AC manager for use with the revised SECURE trust model is discussed below.

5.3 Trust-reasoning using the Revised SECURE Model

As a result of attempting to permit both forms of risk-policy — dynamic-thresholding and metric-based — the interface between OASIS and SECURE components described in the previous section is quite convoluted. The two approaches are also not particularly complementary, with most policies exclusively using the one form or the other. As demonstrated in section 5.2.4, the policies produced are also complex and therefore rather unsuitable for pervasive computing applications where it is likely that the end-user will be involved in the setting of policy.

The revision of the SECURE trust model to use event structures for trust-contexts [CDKN04] meant that the interface needed to be redefined to account for this. Since an extension to the OASIS language was in any case required to support the full amount of reasoning required by SECURE, it was also decided to initially define a generic policy language that could be integrated into any suitable policy engine instead of an OASIS-specific extension.

5.3.1 A Generic Policy Language

With the goal of simpler, more user-friendly language in mind, the second version of the policy language will implement only the risk-metric form of policies. As section 4.1 demonstrates, although risk is understood in many different ways by various groups, all definitions centre around two simple concepts (risk and detriment) that are easily understood. Therefore by developing policies solely around the concept of risk, without muddying the waters with abstractions such as trust, it should be much easier for end users to comprehend and write policies. The revision of the SECURE trust model to standardise on (s, i, c) triples as trust values also makes dynamic thresholding the less interesting of the two paradigms.

With this in mind, for each outcome, o_i, the following policy **atoms** are defined that, when combined with standard arithmetic operations, provide to the reasoning engine all the major risk metrics discussed in section 4.1.

- *area*(o_i, c_1, c_2) – the probability that the cost of o_i lies between c_1 and c_2.
- $\mu(o_i)$ – the expected (mean) cost of o_i.
- $\sigma(o_i)$ – the standard deviation of the cost distribution of o_i.

To reason about risk, it is then necessary to perform standard comparison operations (such as $<$ and $\geq$) on these atoms, and combine expressions using boolean operators AND, OR and NOT. Rather than developing a new grammar from scratch it is more efficient to extend an existing one to incorporate the atoms described above. The grammar of the Python language [vRD04] was chosen as it has fairly standard expression semantics and, being an interpreted language, extensions are easy to incorporate because all expressions are evaluated at run-time.

Since a request may have multiple parameterised responses, request-specific rules of the following form are evaluated in order, until one succeeds or the list is exhausted. EXPR is a standard Python expression (as defined in [vRD04]) with the addition of the *area()*, $\mu()$ and $\sigma()$ atoms defined above[3] and $\backslash n$ is a newline character.

$$\begin{aligned} \text{POLICY} &::= \text{RULE} \mid \text{RULE POLICY} \\ \text{RULE} &::= \text{EXPR} \vdash response(param_1, param_2, ...)\backslash n \end{aligned}$$

5.3.2 Example Policies and Validation

In the e-purse scenario described in section 4.3.1 there are three possible outcomes:

1. The user pays the correct amount.
2. The user has insufficient funds to cover the cost of the service received (the bus journey).
3. The user pays with forged e-cash that is rejected by the bank when the payee (the bus company) tries to redeem it.

The payee has three possible responses when a user requests a service: they can accept the contract and serve the customer; accept the contract in return for an up-front payment in the form of a deposit; or they can decline to serve the customer. For example, the payee might have the policy of refusing to serve any potential customer if the expected cost is greater than 0 (rule 5.1), will accept the contract if there is a 99% probability of making a profit (rule 5.2) and will otherwise request a suitable deposit (rule 5.3). This can be represented in the generic SECURE policy language as:

$$\mu(3) > 0 \vdash refuse \tag{5.1}$$

$$\mu(1) + 3 * \sigma(1) < 0 \vdash accept \tag{5.2}$$

$$\text{True} \vdash deposit(\mu(2) - \mu(1)) \tag{5.3}$$

Python has a powerful and expressive grammar. For example, the following policy states that if the expected costs of either outcome one or two are greater than five then the response should be to refuse.

```
filter(lambda x:  x > 5, [mu(1), mu(2)]) |- refuse()
```

To validate the suitability of the Python language for use on mobile platforms, the above example policies were implemented on Pippy,[4] the free Python for Palm implementation. Since no suitable implementation of the e-purse trust model was available, and to allow deterministic testing, constant likelihoods were assigned to all outcomes, although the lack of floating point arithmetic

[3] In the prototype these are implemented as normal Python functions.

[4] `http://pippy.sourceforge.net`

support on the Palm meant this could not be done in a straightforward manner. The solution that was adopted was to use values in the range 0–100 for probabilities instead of 0–1.0. Square roots were then estimated using Pell's equation [Wik04].

Both policies executed without difficulty in under one second[5] on the Palm Tungsten-T device used for testing. The Pippy environment consumes just 430K of memory, the policies and associated test code another 2.9K.

To gauge the accuracy of the integer estimates of the floating point calculations, the same policies were also implemented on a normal implementation of Python using floating point calculations. The results, shown in table 5.2, indicate that, while the estimation of the square root introduces only a very small error (Pell's equation has a maximum error of 1), estimating floating point equations in integer arithmetic introduces a larger error which, when combined with the error due to estimating the square root, is potentially significant. This problem can be mitigated with more complex fixed point structures at the cost of increased programmer time and effort.

	$\mu(1)$	$\sigma(1)$	$\mu(2)$	$\sigma(2)$	$\mu(3)$	$\sigma(3)$
Floating point	-13.93	11.404	-2.985	11.401	0.0	0.0
Integer estimation, fp sqrt	-13	12.450	-2	11.576	0	0.0
Integer estimation, approx sqrt	-13	13	-2	12	0	0

Table 5.2: Comparison of numerical computation methods.

This set of generic extensions for making trust- and risk-based decisions will now be integrated into the OASIS RBAC policy language.

5.3.3 Integration with the OASIS RBAC System

The previous OASIS-based AC manager performed the risk evaluation internally (albeit, via calls to external risk predicates), based on information provided by the SECURE trust framework and cost analysis components. In this new model for access control, the generic risk language of the previous section is implemented as an external software component to which the OASIS system communicates using its environmental predicate call-out system, as before. This is illustrated in figure 5.6. The predicates used are:

Trust retrieval: As before, this first predicate retrieves relevant trust values into a rule parameter for a named principal. However, instead of retrieving named contexts, the name of the action is passed, and all the trust information associated with that event structure is retrieved into T_v:

$$trust(principal : String, action : EventStructure, T_v? : TrustValue)$$

Risk evaluation: This predicate evaluates the risk of the action using the rule evaluation described in section 5.3, returning a response value if one is

[5] Unfortunately this was the smallest unit of measurement available.

needed. In some situations, such as rôle activation, the response parameter is optional since risk evaluation does not need multiple responses, but instead succeed/fail semantics are sufficient. T_v is the trust values for all outcomes in the event structure *action*.

$$risk(action : EventStructure, T_v : TrustValue[, response? : String])$$

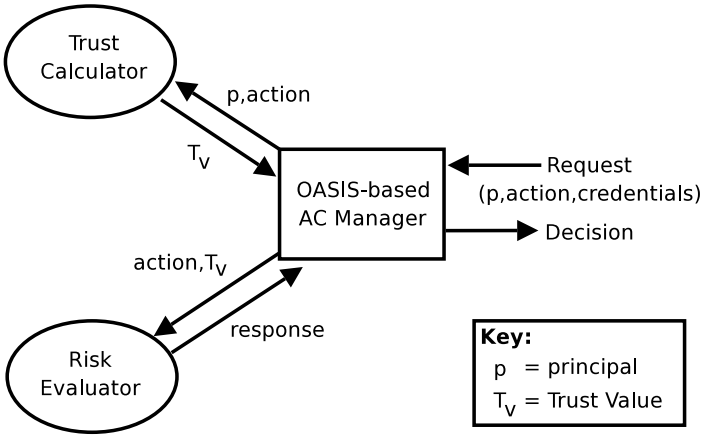


Figure 5.6: SECURE framework with OASIS-based AC Manager.

5.3.4 Spam Detection Revisited

Using this new model, the spam detection policy of section 5.2.3 becomes:

$$trust(sender, email\_action, t),$$
$$risk(email\_action, t, response) \quad \vdash request(sender, email\_action, response)$$

The use of the expectation abstraction also allows the risk predicate to be greatly simplified, in keeping with the aims of the new model.

$$\mu(\text{'mark\_real'}) + \mu(\text{'mark\_spam'}) < 0 \quad \vdash \quad mark$$
$$True \quad \vdash \quad pass$$

The implementation of this policy in the Python-based policy language described in section 5.3.1 is shown in figure 5.7. Although the complexity of the policy still exists, it has been moved to the $\mu()$ function, thus would be defined by the application programmer rather than the policy writer.

5.3.5 Rôles and Trust in Grid Policies

Rewriting the grid policies described in section 5.2.4 to use the new model is more difficult as the change in the trust model requires the application to be reformulated to use event structures and (s, i, c) triples. Instead the improved properties of the new model will be demonstrated using the second Grid use-case outlined in section 5.2.4, that of user Bob who wishes to store a data-set

on a storage server near to his compute server. The server bills for space used after-the-fact so it must trust that Bob is a genuine member of the University, he is authorised to use the service, and the University will pay the bill.

Bob's university issues rôle certificates of the form:

$$\begin{aligned} \mathit{Professor} &\vdash \mathit{Researcher}(\mathit{Cambridge}, 5) \\ &\ldots \\ \mathit{Research\_Asst} &\vdash \mathit{Researcher}(\mathit{Cambridge}, 2) \\ \mathit{PhD\_Student} &\vdash \mathit{Researcher}(\mathit{Cambridge}, 1) \end{aligned}$$

In order to use any grid services, Bob must enter the rôle of *GridUser*. The Grid OASIS server uses the following policy which states that a user can only enter the *GridUser* rôle whilst they hold the rôles of a logged in user and researcher. It also computes its trust in the researcher's institution to make good the payments of its grid users, and the risk associated with issuing such a rôle certificate. If the risk evaluation succeeds then the rôle of GridUser is granted, parameterised with the user's institution and research grade.

$$\begin{aligned} &\mathit{LoggedInUser}^*, \mathit{Researcher}(\mathit{inst}, \mathit{lvl}), \\ &\mathit{trust}(\mathit{inst}, \text{`GridUser'}, t), \mathit{risk}(\text{`GridUser'}, t) \vdash \mathit{GridUser(inst,lvl)} \end{aligned}$$

Bob then presents his GridUser certificate to the file server to use the storage service. The file server evaluates the following policy, which checks that Bob holds the rôle of *GridUser*, computes its trust in the institution and checks the risk of permitting the requested action:

$$\begin{aligned} &\mathit{GridUser}(\mathit{inst}, \mathit{lvl}), \\ &\mathit{trust}(\mathit{inst}, \mathit{action}, t), \\ &\mathit{risk}(\mathit{action}, t, \mathit{response}) \vdash \mathit{request}(\mathit{principal}, \mathit{action}, \mathit{response}) \end{aligned}$$

This policy demonstrates the disadvantage of the simpler, cleaner interface between OASIS and SECURE — the *lvl* parameter of the *GridUser* rôle can no longer be passed to a trust or risk predicate for inclusion in further computations.

Bob gives the compute server permission to access the files by issuing it with a *FileReader(Bob,filename)* rôle certificate. The storage server must trust that the compute server will behave responsibly, and not consume excess resources by, for example, using a pathological access pattern.

Therefore the policy the storage server uses to determine the compute server's access to Bob's files is as follows:

$$\begin{aligned} &\mathit{ComputeServer}(p), \mathit{FileReader}(p', \mathit{filename}), \\ &\mathit{trust}(p, \mathit{read\_file\_action}, t), \\ &\mathit{risk}(\mathit{read\_file\_action}, t, \mathit{response}) \vdash \mathit{request}(p, \mathit{read\_file\_action}, \mathit{response}) \end{aligned}$$

```
def entity_recognition():
    history = trust[‘spam’][0]+trust[‘spam’][2]
    H=1
    return (trust[‘conf’][0]/trust[‘conf’][2])*(1-math.exp(-history/H))

def mu(outcome):
    P_real = (trust[‘spam’][0]/(trust[‘spam’][0]+trust[‘spam’][2])) \
             * entity_recognition() + bayes()*(1-entity_recognition())
    if outcome == ‘mark_real’:
        return P_real * cost[outcome]
    elif outcome == ‘mark_spam’:
        return (1 - P_real) * cost[outcome]
    else:
        return 0

def email_mark():
    print "Message marked as spam"

def email_pass():
    print "Pass message"

policy = ["mu(‘mark_real’) + mu(‘mark_spam’) < 0 |- email_mark()",
          "True |- email_pass()"]
```

Figure 5.7: The implementation of the spam detection policy in Python.

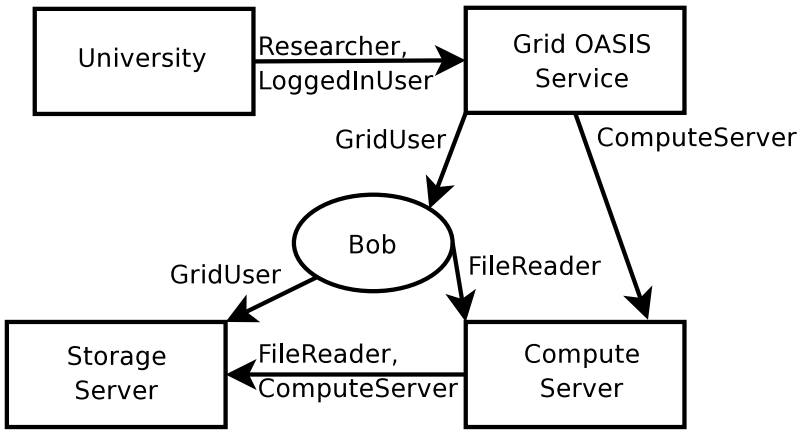
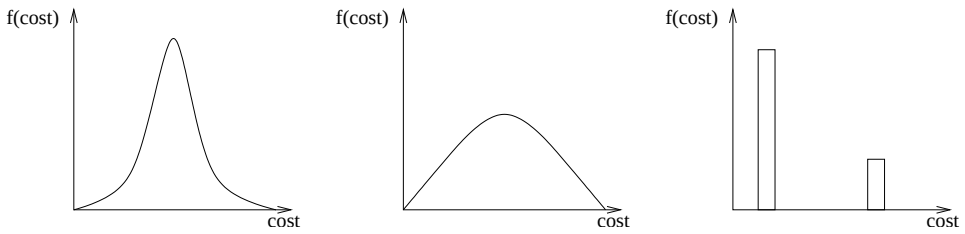


Figure 5.8: Overview of the rôle certificates used by Bob in his interactions with the Grid.

The compute server is required to present rôle certificates claiming that it is a compute server called p, and an authorised reader of the file *filename*, owned by p'. The storage server computes the trustworthiness of p and the risk of allowing access to that file. An overview of these interactions is shown in figure 5.8.

5.4 Evaluation of the SECURE Risk Model

This chapter has demonstrated that the SECURE risk model provides a powerful and flexible method for modelling risk in a variety of applications. Uncertainty in the cost of outcomes is easy to model, and the ability to parameterise cost-PDFs with trust-values allows the width of the distribution function to be adjusted according to the information content of the trust-value (see figures 5.9(a) and 5.9(b)). However, while this models some situations well, it is not a suitable approach for situations where the costs are not continuous and so it does not make any sense to "distribute" the probability across costs that do not represent any real world outcome, such as in figure 5.9(c).



(a) Low uncertainty in the cost.

(b) Higher uncertainty in the cost.

(c) Binary outcomes: difficult to represent uncertainty.

Figure 5.9: The width of the cost distribution functions can be adjusted to model uncertainty, but this is not applicable in all cases.

One of the design aims of this risk model was to allow uncertainty in the trust assessment to propagate through the decision-making process and be explicitly reasoned about. However, as the SECURE trust model developed in parallel with the risk model, it became clear that the information-ordering on trust values did not measure the accuracy of the trust assessment as first thought, but instead conveyed a measure as to the quantity of data that the trust assessment was based upon. Since standard deviation is a measure of uncertainty, a generic policy language based around the μ and σ operators cannot support reasoning about information content directly.

A second problem is that, while cost-PDFs are a good general formalisation, and applicable in scenarios such as the e-purse described in section 4.3.1, in general, constructing suitable cost-PDFs to model an application is difficult. Handling continuous functions computationally can be expensive, particularly when integration is required (for example, for finding the means and standard deviations needed for the generic policy language of section 5.3.1). While this

cost may be acceptable on large servers with plenty of processing power, it is inherently unsuitable for less powerful embedded and handheld computers.

The use of purely monetary costs has also been criticised as being unrealistic since there are examples, such as human life, to which assigning monetary value is impossible. The author disagrees, since insurance companies and courts of law frequently assign arbitrary value to human life, and using a tangible substance, such as cash, allows the use of a market mechanism to quantify preference and priorities that may then be distilled into policy. Computers make decisions according to logical criteria, and some form of quantification is inevitable. However, it is acknowledged that, from the wide range of application examples explored subsequently, in systems where money is not a factor such as the spam example of section 4.3.2, its use can be somewhat artificial and confusing.

A third drawback of this risk model is that outcomes are considered to be independent of the server's response, which leads the model to be slanted towards yes/no decisions, as found in traditional access control. This also makes for a poor abstraction in situations where the same outcomes were possible under different responses by the decision-maker such as the spam example described in section 4.3.2.

A revised model which aims to address these issues will be described in the next chapter.

CHAPTER 6

Economic Models of Risk and Decision Making

The previous chapter concluded with an evaluation that highlighted a number of weaknesses in the SECURE risk and access control models. This chapter develops a new model of risk and access control, based on the economic theory of decision-making.

6.1 Decision-Making under Uncertainty in Economics

In chapter 4 it was noted that economists such as Knight [Kni21] characterise risk as being a quantifiable type of uncertainty. After considering the application scenarios described so far in this thesis, it was noted that for the agents operating in this global computing environment, the uncertainty in the outcome is usually due to uncertainty as to the future behaviour of the other agents — the potential for a passenger to pay, not pay or defraud a bus company for instance. The trust models of evidence and reputation described in chapter 2 ultimately allow this uncertainty about potential actions to be transformed into risk, and consequently the uncertain situation can be reasoned about as a risky one, using trust-based decision engines as described in the previous chapter, and later in this one.

6.1.1 von Neumann-Morgenstern Expected Utility Theory

Economists have been searching for analytical models for making risk-based decisions for some time. von Neumann and Morgenstern [vNM47] were among the the first to formally axiomatise an expected utility hypothesis of agents'

preferences over different ventures with random prospects, usually referred to as *lotteries*.

In terms of trust-based access control (TBAC), one could consider the actions of another principal to be a lottery, with the parameters of the lottery determined by the response of the TBAC system. However, the von Neumann-Morgenstern expected utility theory relies upon *objective* probabilities, that is, probabilities calculated using only measured past behaviour, with no ability to encode belief, or the inherent subjectivity of risk noted in section 4.1. In the objectivist world, for example, two rational agents with equal preferences with respect to the consequences must make the same decision when presented with identical information.

For risk and trust-based systems the theory of *subjective* probability [Ram31] seems more applicable. This theory can be illustrated with the example of a horse race. Spectators have more or less the same (lack of) knowledge about factors such as the horses, track and jockeys, yet while sharing the same information, different people place different bets on which horse will win. Thus, the subjective probabilities held by each spectator can be inferred from observation of their actions, that is, on which horse(s) they place bets. This was formalised, and developed into a preference function analogous to the von Neumann-Morgenstern expected utility theory for objective probabilities, by Savage [Sav72].

Unfortunately the lack of objective uncertainty in Savage's theory means that obtaining the required representation is very difficult, and the more tractable Anscombe-Aumann formulation [FU01] relies upon part of the problem being analysed using the von Neumann-Morgenstern idea of objective lotteries. This can be applied to the problem of TBAC by representing principal behaviour in the subjective part of the formulation and environmental events beyond the control of the actors in the objective analysis. However, the later work of Hirshleifer [HR92], seems a much more natural fit to the problem.

6.1.2 The State-Preference Approach

Hirshleifer's state-preference approach [HR92] aims to reduce choice under uncertainty to a conventional choice problem by changing the commodity structure appropriately. Thus, instead of forming preferences over lotteries, preferences are formed over state-contingent commodity-bundles or, if it is assumed the commodity is currency, then *state-payoff* bundles. To use Hirshleifer's classic illustration, "ice cream when it is raining" is a different commodity from "ice cream when it is sunny".

Given a suitable formulation of the situation, the von Neumann-Morgenstern (vNM) expected utility rule can then be applied to choose the best course of action given the decision-maker's beliefs about which state is likely to obtain. Since state-payoff bundles are easier to work with than the abstract lottery formulation, Hirshleifer's model was chosen as the basis for a new TBAC model for SECURE which will be described in the next section. Before that, a brief look is taken at the objections to the expected utility theory.

6.1.3 Objections to Expected Utility Theory

Extensive experimental research has resulted in many researchers concluding that the expected utility theory is not an accurate model of how humans make decisions, and much debate on this subject continues [Mac87]. It is felt that the psychological side of this debate — such as the supposition that humans do not meet the criteria for being rational[1] agents, and issues of the way in which the problem is framed [Sim82] — is irrelevant to the SECURE model. *The aim of this thesis, and the SECURE project, is to build autonomous agents, not artificially intelligent ones.* Machina [Mac87] details a number of phenomena that are not well-modelled by the vNM utility rule, but since no unified model for all the phenomena described by [Mac87] has yet been found, for simplicity, initial experiments will be restricted to using the vNM rule.

6.2 A Revised Trust-Based Access Control Model

The state-preference model has the following five components:

- a set of acts available to the decision-maker (X);
- a set of (mutually exclusive) states available to Nature (S);
- a consequence function, $c(x, \varsigma)$, showing outcomes under all possible combinations of acts and states[2];
- a probability function, $\pi(\varsigma)$, expressing the decision-maker's beliefs;
- an elementary-utility function (or preference scaling function), $v(c)$ measuring the desirability of the different possible consequences.

The states available to Nature represent any uncertainty relevant to making the decision. In economics, Nature is said to "choose" the state of the world [HR92]; in global computing Nature shall be taken to be the principal, or aggregation of the principals, about whom the uncertainty in the decision originates, where the possible states represent their potential actions.

Provided $v(c)$ is a suitable utility function, as defined below in section 6.2.1, the von Neumann-Morgenstern theory then gives the utility of each act, $x \in X$ as:

$$U(x) \equiv \sum_{\varsigma \in S} \pi_\varsigma v(c_{x,\varsigma})$$

6.2.1 Defining Preference using Utility

In the original SECURE risk model, preference over outcomes was expressed in the form of monetary costs and benefits, the advantages and disadvantages of

[1] that is, will act to maximise their personal utility

[2] NB: ς is used to denote elements of S to avoid clashing with the existing SECURE terminology of using s as part of an (s, i, c) trust-triple.

which were discussed in section 5.4. In this new model the economists' concept of utility, an abstract metric that represents money adjusted for time-effects (such as inflation and interest) and relative wealth, is used instead, thereby overcoming the drawbacks associated with using an actual currency.

Whilst Savage observes that economists have proved that there is no meaningful universal measure of utility [Sav72], the following definition from page 73 of [Sav72], seems well-suited to this framework:

> **Definition** A real-valued function of consequences, v, is a utility; if and only if $f \leq g$ (i.e. g is preferred to f) is equivalent to $v(f) \leq v(g)$, provided f and g are both with probability one confined to a finite set of consequences.

It is noted that the concept of monetary costs and benefits satisfies this definition, which could be useful in applications where financial considerations are already present, for example the e-purse scenario outlined in section 4.3.1.

In order to apply the von Neumann-Morgenstern expected utility rule, $v(c)$ must also be determined using a method termed "the assignment of cardinal utilities to consequences". von Neumann and Morgenstern showed how this could be done using the *reference-lottery technique*, which maps the income received in each state to a probability [HR92]. Thus, $v(c)$ determines the decision-maker's risk policy (whether they are risk-averse, risk-neutral or risk-taking) with respect to the utility of each consequence $c_{x,\varsigma}$. Experiments in this thesis are restricted to a risk-neutral (linear) functions of $v(c)$.

6.2.2 Access Control Policy

A simple access control policy would be to choose any act, $a \in A$ where:

$$A = \left\{ a \mid a \in X \wedge U(a) = \max_{X}[U(x)] \right\}$$

If $|A| > 1$ then choose the action $a' \in A$ with the smallest variance of utility.

Unfortunately this model does not yet address the problem of reasoning about the uncertainty and information content of trust-values. Intuitively a person will not feel confident in taking a course of action based upon a decision for which they have too little information. The quantity of the information required to satisfy them that this is the correct course of action will depend on their risk-aversity, and therefore must be encoded as part of the person's policy. Therefore the policy-writer is permitted to set a lower-bound l for the number of bits of information known about the chosen action a, in order for it to be executed. If this check fails, then the system may test other actions in the set A or fall back to a pre-defined, "default" action.

6.2.3 Integration with the SECURE Trust Model

While this model is designed to be used with any suitable trust metric, it will now be shown how it can be instantiated with the *Revised SECURE Trust Model* (section 2.3.2 and [CDKN04]).

In this trust model observations on the past behaviour of principals are recorded in an event structure which, when combined with observations from other principals, can be used to deduce a trust value for that principal. Figure 2.1 shows an example of an event structure where the following observations are possible:

- Nothing may have been observed ($\emptyset$);
- *reject* may have been observed;
- *accept* may have been observed, but it is not yet known whether this is of subtype *forged* or *verified*;
- *forged* may have been observed (implying observation of *accept* as well);
- *verified* may have been observed (implying observation of *accept*).

The SECURE trust engine [CDKN04] can therefore compute a trust value of the form, (s, i, c) for each of the possible events, *reject*, *accept*, *forged* and *verified*, where s is the number of supporting observations, i is the number of inconclusive observations and c is the number of contradictory observations. Given a suitable mapping from the appropriate event structure E to mutually exclusive states S, conversion of this form of trust value to a probability measure π, is trivial. A typical mapping from E to S would be to use the leaves of the structure as these represent the set of possible final outcomes.

Entity Recognition (ER) and Information

As with information, the required level of ER confidence should be determined by the risk associated with the decision.

Intuitively, if an entity is incorrectly recognised, then the estimates for $\pi(\varsigma)$ are likely to be completely invalid. One way of representing this in the model would be to use the confidence in entity recognition to scale the measure of information the decision is based upon. The decision to perform act a is known to be based upon trust-values of the form (s, i, c) and since states are mutually exclusive, $\sum_S \pi(\varsigma) = 1$. Therefore it is possible to compute an information metric, $I = s + i + c$, which is a constant for all $\varsigma \in S$. This is then reduced in magnitude by the probability of incorrect entity recognition, $(1 - p_{er})$ before being compared to the information threshold l, described section 6.2.2.

An alternative method would be to perform a second risk analysis using the state-preference approach, once the set A has been determined. In this second analysis there would be just two states, {*entity recognised correctly*, *entity recognised incorrectly*}, and the acts would be A plus a fall back act for the eventuality that the risk of mis-recognition is too high to take any of the acts in A.

6.3 Spam Detection Revisited

Initial evaluations of this new model were carried out using the example of spam filtering. This is a somewhat simpler initial example than the e-purse described

in section 4.3.1 and large quantities of test data are readily available.

When determining how to handle an email message, the decision-maker must decide whether to *mark* a message as spam or allow it to *pass* into the inbox. The situation can therefore be modelled as follows:

- The set of acts available to the decision-maker is $X = \{mark, pass\}$.
- The set of (mutually exclusive) states available to Nature is $S = \{spam, notspam\}$.
- The consequence function, $c(x, \varsigma)$ with example utilities for each consequence shown (where E is a parameter that allows the sensitivity of the filter to be adjusted):

X/S	spam	notspam
mark	1	-E
pass	0	0

- The probability function, $\pi(\varsigma)$, expressing the decision-maker's beliefs in whether the message is spam or not. This uses the same event structure as previous instantiations of this application in this thesis, and methods of determining the belief are discussed in chapter 7.

6.4 Selection under Uncertainty as a Decision Problem

One of the weaknesses of the model described in chapter 4 and 5 is that it is optimised for yes/no style decisions and making a selection, for example choosing the best principal (or principals) from which to request a service, is very inefficient. This new model may be used to make a selection in the following way:

- The set of acts (X) available to the *selector* is equal to the set of potential service providers, P, or the power-set of P if combinations of providers are to be considered.
- The set of (mutually exclusive) states available to Nature (S). Each state is whether a particular principal was "good" or "bad". For example, $\varsigma_1 = \{p_1\}, \varsigma_2 = \{p_1, p_2\}$ means that only p_1 satisfied the selector in state ς_1, while in state ς_2, p_1 and p_2 satisfy the selector, and so on.
- The consequence function, $c(x, \varsigma)$, showing outcomes under all possible combinations of acts and states — so $P \times S$ in the simple case where $X = P$. Good consequences, (x_i, ς_j) are those where $p_i \in \varsigma_j$.
- The probability function, $\pi(\varsigma)$, expressing the user's beliefs.
- The elementary-utility function (or preference scaling function) $v(c)$, measuring the desirability of the different possible consequences — so those

that are "good" consequences, as mentioned above, have higher utility than those that are "bad". This function could also encode constraints such as a budget.

6.4.1 Choosing a Grid Server: An example of selection under uncertainty

Recall that in chapter 5 Alice had to choose a suitable set of Grid servers on which to store her files. For the sake of keeping this example simple, assume that Alice wants to choose two servers from three possible providers: B, C and D.

- The set of acts available to Alice is $X = \{BC, BD, CD\}$, the set of possible combinations of choosing two providers from three.
- The set of (mutually exclusive) states available to Nature (S) is the power-set of the set of possible principals P, where $p \in \varsigma$ has the meaning that the principal p satisfies Alice's requirements when state ς obtained.

$$S = \{\{B\}, \{C\}, \{D\}, \{B, C\}, \{B, D\}, \{C, D\}, \{B, C, D\}, \emptyset\}$$

- The consequence function, $c(x, \varsigma)$, showing outcomes under all possible combinations of acts and states. For example, assigning a unit of utility for each of Alice's chosen servers that perform to requirements gives:

X/S	B	C	D	BC	BD	CD	BCD	$\emptyset$
BC	1	1	0	2	1	1	2	0
BD	1	0	1	1	2	1	2	0
CD	0	1	1	1	1	2	2	0

- The probability function, $\pi(\varsigma)$, expressing Alice's beliefs about the likelihood of each of the states in S, that is, Alice's computed beliefs as to how well each server will perform.

6.5 Using Bayesian Trust Models

Although designed as an access control model for the SECURE framework, the model described in this chapter should be general enough to interact with any suitable trust model, not just the one outlined in section 2.3.2. Another popular form of trust, reputation and evidence model is based on Bayesian statistics and networks. There are a number of similar such models, although the model presented by Buchegger and Le Boudec [BB04] appears to be one of the most formally grounded and clearly explained.

Buchegger and Le Boudec's model assumes that all potential collaborators will misbehave with probability θ, and that the outcome is drawn independently for each interaction. The parameters for θ are themselves unknown, and this is modelled by assuming θ is itself drawn from a distribution (the prior) that is

modified by each piece of evidence received. By using a Beta(α, β) distribution as the prior, the initial case for previously unknown principals is the uniform distribution. In the limit, as the quantity of evidence tends to infinity, the distribution tends to the Dirac function at θ.

The benefit of this model over the SECURE (s, i, c) triples is that it it also allows for more expressive modelling, particularly with respect to time. The SECURE model stores only raw "event counts" which makes it difficult to model potentially desirable behaviours such as more recent events being more important than old history ("reputation-fading" in [BB04]). In contrast the update functions used in a Bayesian model can easily be modelled to change the distribution in whatever manner is required.

The disadvantage of Bayesian models is that they revolve around the binary concepts of "good" and "bad" behaviour, compared to the semantically rich event-structure of outcomes approach of SECURE. However, Halpern and Pucella's logic for reasoning about evidence [HP03], allows the extension of this basic Bayesian model to a world where a sequence of observations may be used to determine the likelihood of truth about a set of mutually exclusive and exhaustive hypotheses, $\mathcal{H}$. If $S = \mathcal{H}$ then Halpern and Pucella's logic can be used to determine a belief function $\pi(\varsigma)$, as required by the access control model.

In general, the SECURE trust model can use any set of trust values, T, provided it is possible to define both trust and information (partial) orderings over it. (The (s, i, c) trust value format is only an example of one such suitable format.) Suppose, instead of using the set of outcomes as the basis for trust values, T represents a set of principal models. A simplistic example of this might be $T = \{$*Honest,Dishonest,Incompetent*$\}$. If $\mathcal{H} = T$ then Halpern and Pucella's logic of evidence can be used to determine a probabilistic trust assignment for p; that is, a probabilistic assessment that principal p is of underlying trust-type t.

The information ordering is determined by the probabilistic assignment of trust values to the principal. The quantity of information a decision is based upon can therefore be measured using the variance of the distribution[3] $v(c_{x,\varsigma})$. In [BB04] unknown principals are modelled using the uniform distribution which is equivalent to making all trust values equally likely in the model outlined here. However, this is unsatisfactory as it is then impossible to distinguish between an unknown principal and a very well known principal who alternates between different underlying states of behaviour on an equal basis. The significance of this can be illustrated by considering that if 60% of previously unseen principals behave maliciously, then a principal who behaves badly just 33% of the time would be a preferable customer to an unknown one. Therefore the initial probability distribution used for an unknown principal should be determined by the behaviour of all previously encountered unknown principals on their first interaction, effectively the population mean.

This approach also allows for a better abstraction in the SECURE architecture as the trust life-cycle manager component may then also have a policy component that specifies how to handle new principals without adding more complexity to the access control policy with special rules for new principals.

[3] c.f. the utility of act x is the expectation of this distribution.

For example, the population mean trust value assigned to a new principal may depend on the context in which they are encountered, or what static credentials they can present: an unknown principal who can present a "Member of University of Cambridge" credential may be assigned a different trust value to an unknown principal who cannot. This idea of a *trust formation* policy allows the policy-writer to determine for themselves what trust value should be assigned to an unknown principal before it is possible to compute a population mean (the "bootstrapping" problem of reputation systems) and also provides a solution to the situation where malicious users make the population mean of new users so low that new users are never accepted into the system.

6.5.1 Example: An e-purse

Modelling trust in ability to pay is difficult as there are many possible behaviours that the model should capture. For instance, an attack on a naïve system would be for Eve to build up trust by successfully executing a very large number of low-value transactions, and then defaulting when she was sufficiently trusted to execute a high value transaction, such as buying a car.

In the Grid application described in section 5.2.4 this was handled by dividing the space of possible payment values into intervals and making different trust assessments for each one. The same method could be used with a Bayesian model, the two possible states being payment or default, but depending on the size of the intervals, this provides very coarse granularity on the value of transactions and is only suitable when the amount to be paid is known to the decision-maker in advance.

Instead let the set of trust values T be a set of intervals $[l, u]$ where $0 \leq l \leq u \leq$ `maxprice`, and `maxprice` is the price of the most expensive item sold. A principal with trust value $[l, u]$ is considered to be "good" for amounts up to £l (perhaps, for example guaranteed by a credit card company), and no good for amounts over £u. For amounts in the region $l < £ \leq u$ there is uncertainty as to whether the principal will pay or not.

For the sake of example, suppose bus tickets are available at a cost of 1,2,3,4 and 5 units (formally, $\lambda \in \Lambda = \{1, 2, 3, 4, 5\}$). A trust value of [0,5] then represents an "unknown" principal, whilst [0,0] and [5,5] represent fully distrusted and fully trusted users respectively. Defining the following trust-ordering on this set of trust values gives the partial ordering required by the SECURE trust model:

$$[a, b] \preceq [c, d] \Leftrightarrow (a < c \wedge b \leq d) \vee (a = c \wedge b < d)$$

Consequences, Preferences and Access Control

When a person seeks to board a bus the three possible responses available to the bus company are: Board, Ask for a deposit, and Refuse. Intuitively if the potential passenger has trust [0,0] then there is maximum utility in refusing them permission to board, no matter how far they might travel. Similarly if they have trust value of [5,5] then there is maximum utility in permitting them to use the bus. Since nothing is known about a new principal, there is

some utility in allowing them to board (they will hopefully become a valued customer!) but probably more utility in asking them for a deposit. These intuitions are illustrated in table 6.1.

In the general case, the utility for each action will depend on how far the principal is likely to travel. Since the evidence manager stores historical information about the journeys taken by each principal for the purposes of computing a trust value, further data mining is possible to give a probabilistic estimate of the price of the bus ticket that will be purchased, λ. The set of potential states of Nature, S, is therefore the cross-product of the set of possible trust values and the set of possible ticket prices, $([l, u], \lambda) \in T \times \Lambda$.

The general policy is then that if λ is less than l then the user should be allowed to board. Conversely, if λ is greater than u then boarding should be refused: the utility in each case being equal to the expected value of the journey. If λ lies between l and u then a deposit should be requested. The utility (to the bus company) is the ticket price, λ, as shown in the functions in rightmost column of table 6.1. Note that there is also utility in asking for a deposit if $\lambda < l$ for the purposes of audit — game theory [Ras94] demonstrates the effectiveness of such apparently "random" checks, a topic that will be returned to in section 6.8.

X/S	$[0,0],\lambda$	$[0,5],\lambda$	$[5,5],\lambda$	$[l,u],\lambda$
Board	0	1	5	$\lambda > l : 0, \quad \lambda \leq l : \lambda$
Deposit	0	2	1	$\lambda > u : 0, \quad \lambda \leq l : \lambda, \quad l < \lambda \leq u : \lambda$
Refuse	5	0	0	$\lambda > u : \lambda, \quad \lambda \leq u : 0$

Table 6.1: The consequence functions for the e-purse scenario.

6.6 The Risk of Seeking More Information

If the outcome of a decision is that there is insufficient information to make a correct decision then in many applications there will be the opportunity to seek some additional trust information about the principal being considered. For example, in the PDA scenario in chapter 3 the PDA could ask the owner of the PDA for guidance. Seeking more information is rarely without cost; this may be monetary in nature, such as paying for a report from a credit bureau, or in terms of the use of a scarce resource such as battery power to perform the network query. Whether this expenditure is worthwhile is the *risk* of seeking this additional information.

In a data-orientated application, such as the PDA application mentioned above, the question is easy to answer: if the value of the data involved in the operation is less than the cost of the query then the expenditure is not worth it. But what about in applications where the value is less clear cut?

Suppose a query for additional information results in an event $b \in B$. Once the decision-maker learns that event b has obtained, then there is a new decision problem to be considered. Savage [Sav72] notes that this new problem is related to the original one as the acts available to the decision-maker remain the same,

and that the utility of each one is now $U(x|b)$, the utility of act x given event b. The original decision problem is called the "basic decision problem" and the new, related one, the "derived decision problem".

The original decision problem, $\mathbf{f}$, has (expected) value:

$$E(\mathbf{f}) = \max_X [U(x)]$$

The derived problem, $\mathbf{g}$, has (expected) value:

$$E(\mathbf{g}) = \sum_B E(\mathbf{g}|b)P(b)$$

where $P(b)$ is the probability of event b. Rationally the maximum cost that should be incurred in seeking more information is $E(\mathbf{g}) - E(\mathbf{f})$, that is the increase in utility expected as a result of observing which event in B obtains [Sav72].

6.6.1 Formal Definition

[Sav72] gives the following formal definition, adapted here for the model described in section 6.2.

1. There is a set of *basic* acts, X.

2. The event is a random variable b associating with each state ς an observed value $b(\varsigma)$ in some set B of possible events.

3. The set of *strategy functions* is the set of all functions associating an element of X with each element $b \in B$.

   $$SF = \{f(b) : b \rightarrow x\}$$

   A strategy is a mapping from observations to basic acts, so the number of strategy functions is $|X|^{|B|}$.

4. Each strategy function $f(b)$ corresponds to a derived act, $x' \in X(b)$ where $X(b)$ is the set of derived acts. The consequences of the derived acts are defined by:

   $$c(x', \varsigma) = c(f(b), \varsigma) \quad \forall \varsigma \in S \text{ where } x' \equiv f(b)$$

   that is, the consequence of derived act x' in state ς is equal to the consequence of basic act x in the same state, where the basic act x is given by the strategy function f corresponding to derived act x'.

5. The value of X given b is:

   $$v(X|b) \equiv \sup_{x \in X} E(x|b)$$

6.6.2 Example – Spam Detection

This table gives the consequences and utilities for the spam application discussed in section 6.3.

X/S	spam	notspam
mark	1	-2
pass	0	0

Suppose the trust model determines there is equal probability a mail is spam and notspam, the resulting expected utility for mark is $-\frac{1}{2}$ and 0 for pass. However, the decision-maker has access to a service that scores each message out of three. The probabilistic relationship between this score (b) and S is shown in table 6.2. It is now possible to determine the set of strategy functions. Since there are two possible acts, and three possible messages/observations from the mail analysing service, there are eight possible strategies of the form $x_1x_2x_3$ where x_1 is chosen if $b = 1$, x_2 is chosen if $b = 2$ and x_3 is chosen if $b = 3$. The eight strategies (derived acts) are therefore: MMM, MMP, MPM and MPP, plus the same again substituting M's and for P's and P's for M's.

b	spam	notspam	$P(b)$
1	0	0.25	0.25
2	0.125	0.125	0.25
3	0.375	0.125	0.5
	0.5	0.5	1

Table 6.2: $P(b \cap S)$

X/B	1	2	3
Mark	-2	-0.5	0.25
Pass	0	0	0

Table 6.3: $E(X|b)$

Table 6.3 shows the expected utility of each basic act given the value of b, $E(X|b)$.[4] This shows that the decision-maker's best strategy is to choose "Pass" if $b = 1, 2$ and "Mark" if $b = 3$. The expected value of the derived decision problem is then:

$$\begin{aligned} E(X') &= \sum_{b \in B} E(X'|b)P(b) \\ &= 0 \times 0.25 + 0 \times 0.25 + 0.25 \times 0.5 \\ &= \frac{1}{8} \end{aligned}$$

Since the expected utility of the basic decision problem was 0, the decision-maker has increased its utility by $\frac{1}{8}$ in asking the external service for additional information. Correspondingly the decision-maker should be prepared to pay up to $\frac{1}{8}$ utiles to use the service.

6.6.3 Events, Trust and Recommendations

The *lingua franca* for transferring trust-information in the SECURE framework is by passing raw trust values as *recommendations*. The set of events, B,

[4] where $E(X = x|b) = \sum_{s \in S} v(c_{x,s})P(s|b)$

used in the spam example are clearly not trust values in this sense. In fact, if the decision-maker is considering whether or not to ask another principal for a recommendation, B will be the set of possible trust values T it could receive. Depending on the application, the decision-maker may have to ask for recommendations from each principal individually, or there may be a broadcast mechanism that would permit a group request.

For simplicity, this thesis will consider only the former situation: the decision-maker will evaluate the utility of contacting each principal within range, where the utility is defined as being the difference between the derived decision problem and the basic one, minus the cost of making the query. Principals should then be queried, in order of their expected utility, until one responds. Once one has responded, there is a decision as to whether to continue seeking more information, or stop. This is taken in the same manner as the first such decision, and repeated until there is no longer any expected increase in utility to be made by contacting more principals.

The computation of $P(b \cap S)$ depends on the format of the trust values. How this is done using the two forms so far discussed in this chapter, (s, i, c) triples and Bayesian, is described below.

SECURE Trust Model

Using the revised SECURE trust model, B is the set of all possible (s, i, c) triples which, while being countable, is sufficiently large to make computation of $|X|^{|B|}$ strategy functions impractical, though not impossible. In practice, the quantity of information in a recommendation will be capped to defend against the attack where a malicious recommender claims to have performed a very large number of interactions with the subject of the recommendation in order for their recommendation to carry as much weight as possible [Ing03]. $P(b \cap S)$ is then determined using the normal rules for incorporating recommendations into the current trust policy — an example is given in section 7.3.4.

Bayesian Trust Model

In the Bayesian trust model described in section 6.5, a principal's trust value is computed as a function of a number of pieces of evidence, including personally made observations and previously received recommendations. Therefore the set of recommendations that could be received from another principal, B, can be treated in the normal manner and speculatively processed by the trust value update function to give $P(b \cap S)$ as required.

6.7 Access Control Policy for the Evidence Store

The previous section described how to determine the "risk" of seeking more information when faced with an uncertain situation. This section will discuss the issues and risks of replying to such requests.

Some of the risks associated with responding to information are similar to those of requesting it in the first place — the use of the network may incur a

traffic or power usage cost. However there is also a privacy risk associated with giving out information as it can reveal with whom a principal has interacted recently, information that would be valuable to market-place competitors and malicious users. Giving out recommendations received from other principals carries further risks as it may, in addition to revealing from whom the principal has received information, also compromise the privacy of the originating principal.

On the other hand, as noted in the PDA scenario described in chapter 3, there are potential benefits to interacting with peers, such as the opportunity for the mutual exchange of information or services. In return for Alice recommending a good supplier of X, Bob will hopefully then look favourably on Alice's request for a recommendation about Y. Alternatively, Bob may have a more recent version of a web page that Alice would like to view, thereby saving her a costly network connection to the Internet. Depending on the information exchange protocol, a more tangible benefit may be that if a principal does not participate then they may be considered to be "freeloading" by other peers and ejected from the network. Freeloading, the practice of consuming from, but not contributing to, a community, is a common problem in existing peer-to-peer networks [AH00]. The anarchic nature of such networks makes it impractical, if not impossible, to centrally enforce rules concerning the ratio of consumption to contribution. In response researchers are developing community-based enforcement mechanisms [NWD03], and it is envisioned that trust assessments could play an important rôle in such systems.

6.7.1 The SECURE Approach

Given the difficulty of enforcing participation in distributed protocols, SECURE takes a *laissez-faire* approach where each principal may decide arbitrarily whether to respond to requests or not. This allows each principal to implement its own *interaction policy*, and thus individually punish principals it views as not participating, but precludes community enforcement. A principal may however inform other principals of its opinions of others using the usual SECURE recommendation framework.

The interface for exchanging recommendations is defined as follows: a principal, the Judge in SECURE terminology, may make a request of another principal for either a recommendation about a specific principal S in a particular context c, or for a recommendation for the most trustworthy principal known to the queried principal for a context c. The queried principal may make no response, or return at least one recommendation, either with itself as the Witness, or pass on recommendations from other principals which it has received. The SECURE architecture defines a recommendation as being a (Witness, Subject, t) tuple which is cryptographically signed.

Upon receiving a request for a recommendation, a principal makes a two-stage decision. In the first stage it must decide whether to respond at all to the request. The set of actions here are "respond" and "ignore". The set of states, S, is as usual defined by the potential actions of the other principal.

If the principal decides to respond at all, then at minimum it must send one

recommendation. Given that there are fewer risks associated with sending a recommendation based on one's own observations than other people's, the first recommendation to be sent must therefore be the sender's own. The possibility of obfuscating, or otherwise perturbing the information to mitigate the risks in situations where a principal is only partially trusted has been considered, but there seems little to be gained from it. [JGK03] shows that identities must be clear for proper analysis of trust chains and adjusting the trustworthiness component of t will reflect badly on the honesty of the sender. Changing the information component of t would help to hide the quantity of transactions the Witness has had with the Subject, but again this reduces the usefulness of the recommendation and since it could only be changed downwards, the recipient will always know that the Witness has had at least that many interactions with the Subject. In light of this it seems the decision to trust or not trust the requester must be binary: the states of Nature are that the principal is *trusted* or *distrusted*.

$\pi(\varsigma)$ depends on the principal's trustworthiness in other areas, static policies,[5] and most importantly, recent behaviour. A typical policy in this regard might be that a request from a principal who has recently responded to a request from the target principal should be looked upon favourably, while a principal who is making unreasonably frequent requests should be ignored for a while. Clearly a key component of this policy is time, which is something the evidence-based SECURE trust model is currently insufficiently expressive to capture. Note that unlike the usual SECURE trust computation, to avoid possible loops in the protocol, no recommendations will be sought from other principals while computing $\pi(\varsigma)$, although principals may asynchronously "push" recommendations about principals they consider to be freeloading.

6.7.2 Forwarding Third-Party Recommendations

If a principal decides to send any response at all, then it must also consider whether to forward any additional recommendations from other (third-party) principals. If the risk to one's own privacy is hard to quantify, the risk to someone else's is impossible to determine. The problem was made tractable in chapter 3 through the re-use of an existing information framework to infer intention and preference; in a general trust-based application, it cannot be assumed that the information can be re-used out of context in this way. The sensitivity of recommendations is likely to be application dependent, although it is also observed that even nonsensitive data can yield intrusive conclusions under sophisticated processing [Swe02].

On the other hand, restricting data flow is also self-defeating — the full power of trust-based recommendation networks such SECURE can only be fully utilised when information is free and open. Therefore, in the SECURE framework, recommendations are considered to be redistributable at the discretion of the recipient unless marked otherwise, using a binary distribute/do-not-distribute field. Privacy can further be protected using pseudonymity: the use

[5] The ability to pre-configure a number of trusted "friends" is useful when bootstrapping recommendation systems.

of separate pseudonyms for different activities allows a user to prevent overly intrusive profiling. Of course a balance is needed here since the greater the availability of identities, the greater the threat of the Sybil Attack [Dou02]. This attack is discussed further in section 7.1.2.

Privacy risks aside, in practice, there is still variable benefit to passing on all the relevant recommendations a principal has — each one still incurs a network cost, for example. A second decision is therefore taken where the set of possible acts is to send any number of redistributable third-party recommendations, from zero through to the total number available to send. The states of Nature remain as in the first decision, that is *trusted* and *distrusted*.

6.8 Game Theory for Strategic Policy Development

In the previous section the interaction between principals became very strategic: decisions became dependent on how the other principal had acted recently, or how the deciding principal hoped they would act in the future. This suggests that further work in this area should be based around game theory [vNM47, Ras94].

Section 4.2 described how the use of monetary costs and benefits to encapsulate policy was intended to help solve the problem of refining policy from high level, often business-orientated, goals to technical implementation. The use of preference-scaling functions in the models developed in this section provide a similar benefit, whilst being more tractable in situations where monetary costs are alien and inappropriate. This process can be taken one step further by noting that the relationship between the fields of game theory and (economic) decision theory can be exploited to further develop this process of turning high level *strategy* into the technical *policy* required to use the models developed here.

The next chapter applies the ideas introduced here to the spam detection example application.

CHAPTER 7

Implementation and Evaluation

This chapter describes the implementation and evaluation of the risk and access control aspects of the SECURE framework. A set of evaluation criteria will be developed which will then be used to review the SECURE framework through the implementation of a number of prototype applications.

7.1 Evaluation Methodology

As an emerging access control paradigm, there is not yet any established evaluation criteria for computational trust systems such as SECURE.[1]

The high complexity and many inter-connected components of computational trust systems means that many papers address and evaluate only individual components, such as the trust metric [FKÖD04] or within the context of a particular application for which the trust system was specifically designed [KSGM03, BLB03]. However, as this thesis has shown, computing a trust value for a principal is only the first step, and the application-specific access control decision only the ultimate goal; in between confidence must be established in the identity or credentials of the requesting principal(s) and the risk of the various courses of action assessed. Some issues can only be addressed in the context of the framework as a whole — assuming the use of pseudonyms to protect privacy cannot be done without considering the Sybil attack, for example.

7.1.1 System-Level Evaluation Criteria

Clearly the most important criterion for any application is how well it improves on the current situation. In some applications, such as spam filtering,

[1] This evaluation methodology was developed in collaboration with Ciarán Bryce, Vinny Cahill and Jean-Marc Seigneur and published in [BDK+05].

this may be how much better the trust-based solution is compared to existing, non-trust-based ones. In other applications, such as mobile ad hoc networking, computational trust may offer unique features that could not otherwise be offered, or the benefit may be more intangible — convenience and ease of use, such as in the PDA application described in chapter 3.

Beyond that, what is the stated aim of a computational trust system? In section 2.5 it was noted that:

> For a given application domain, a computational trust system should be able to select an interaction partner (or partners) such that the risk of interacting with that principal is deemed to be acceptable to the decision-maker.

Ease of use is also important. Discussions with the Information Security group of a large company revealed that they experience significant tension between security and convenience for end-users: intrusive security measures that prevent effective working are often circumvented and ignored.

Finally, the computational trust system must perform efficiently on the platform on which the application is designed to run. For example, in mobile ad hoc networking where decisions are being made at the network level, efficiency is of much greater importance than in a server-based application-level access control system.

7.1.2 Component-Level Evaluation

Using the system-level evaluation criteria outlined above, an evaluation methodology for each major component in the SECURE framework will now be developed. The approach used is a threat-based analysis: for each component the possible attacks will be enumerated and the correct behaviour that should be demonstrated to prove correctness described. An attack is defined as follows:

> An attack by a principal or group of principals is any behaviour which aims to influence a decision by another principal in the system in such a way as to be detrimental to that principal.

Trust Model: Behavioural Attacks

In line with the definition of attack given above, behavioural attacks are patterns of behaviour by a principal with the aim of directly influencing another principal's trust in them. One such attack would be *oscillating*, where a principal repeats an infinite cycle of building trust through a number of interactions before turning malicious until their trust value drops again, then appearing to mend their ways, and repeating.

Many attacks will be application-specific. Therefore to evaluate this component it is necessary to define a set of principal types expected to be seen in the application, *regular user*, *infrequent user*, or *spammer* for example, and specify what trust value the model is expected to give to principals who have been

identified as being of one of these types.[2] Evaluation is by means of simulated interactions to measure the time to converge to the required trust value.

Trust Model: Reputation-based attacks

The reputation component of the SECURE trust model is an instantiation of the main trust model for the action of recommending other principals, hence a similar approach to evaluation is required. For reputation systems it is observed that there are the following types of principal:

Honest: Give truthful recommendations, to the best of their knowledge and ability.

False-Praisers: Give false positive recommendations with the aim of making the unsuspecting user interact with the wrong principal (perhaps pushing business towards one of their friends or other identities, for example).

Defamers: Give false negative recommendations with the aim of causing a user to not interact with a principal they would otherwise have done business with.

Liars: Give both false-praise and defamatory recommendations.

Randoms: Give out recommendations that have no basis in fact or purpose other than to mislead and confuse. This may be deliberate or accidental (for example, a misconfigured client).

The evaluation technique of determining the "correct" trust value for each principal, then measuring the time to converge in a simulated environment, is again applicable.

Sybil Attack and Identity Theft

Whilst evaluating entity recognition is beyond the scope of this thesis — all the example applications used herein rely on public-key cryptography for identity recognition — it is still necessary to consider other identity-based threats such as "Sybil Attacks" [Dou02] and identity theft. A Sybil attack is where a single entity forges multiple identities in order to attack the system. Douceur demonstrates the practical difficulty of determining whether any two entities are distinct in protocols, such as recommendation systems, that rely on the redundancy to enforce particular policies [Dou02].

Defending against identity-based attacks requires an analysis of the trust metric to determine how many nodes need to be under the control of an attacker in order to be able to perturb or control the trust value of a particular principal [TD03]. Once the attack analysis has bounded the number of nodes in the trust network needed to influence trust values, the vulnerability to the Sybil attack can be considered. [FR01] suggests that the Sybil attack can only

[2] It is observed that this is much easier in the Bayesian trust models of section 6.5 where this definition forms part of the trust model.

be mitigated by limiting each individual to a very small number of lifetime identities. However, this precludes the ability for users to act pseudonymously, one of the aims of the SECURE project. Alternatively, if the number of nodes an attacker must control is known then rate-limiting the attacker's ability to create new identities should be sufficient. This could be achieved by using certification agencies [Ing05] or proof-of-work puzzles [SL03].

Access Control Manager and Risk Evaluator

In the models described in this thesis, the risk components have been mostly static, effectively forming part of the user's policy. If SECURE were to use dynamic risk, that is, risk that evolves over time in response to observations as postulated in section 4.2.3, then this would need to be evaluated with respect to the speed of adaptation and how prone is was to manipulation by an external entity. For example, whether a deliberate pattern of events would allow a malicious principal to change the risk assessment of a system it was attempting to attack for its own ends.

To evaluate the static components of policy such as static risk assessments and access control policy, the principal types used in the evaluation of the trust model can be re-used. For each request of a given level of risk, the expected response for each principal type should be defined. The component can then be tested by inputting the appropriate trust value for that principal type and comparing the response to the expected one.

Since the end-user is likely to have the most interaction with the policy components these should also be evaluated for expressiveness (can the user express the policies he or she requires?), ease-of-use (how much knowledge and/or training do they require?) and computational efficiency (can the policy be executed on the desired platform).

These criteria will now be used to evaluate the SECURE framework in a number of different application scenarios.

7.2 Using SECURE to Fight Spam

The worldwide cost of unsolicited bulk email ("spam") has become intolerable, with reports that since the middle of 2003, spam accounts for over half of all email sent over the Internet [Gra03]. There are many potential solutions, from legal remedies and suggestions for changes to the SMTP protocol, to improved blocking and filtering at the recipient, but none have so far succeeded.

The root cause of spam is ultimately the same property of email that makes it so attractive and useful: the low cost of communicating with a large number of people all over the world. Moreover, the near-zero cost of creating and spoofing an email identity ensures that even when the sending of unsolicited bulk messages is prohibited by law or ISP policy, tracing and punishing the offender is not easy, because the underpinnings of the current systems were not designed with authorisation and secure authentication in mind. Proposed solutions that attempt to remedy this oversight are infeasible in the short term

as transitioning all of the world's email users to a new system is a monumental task.

Spam is an excellent application scenario with which to experimentally validate the ideas developed in this thesis for the following reasons:

- Email is an Internet-wide application with millions of users, all of whom are affected by the problem of spam, thereby satisfying the definition of *Global Computing*.

- The problem is simple to formulate, yet difficult to solve because the system was designed when all principals could be trusted. Similarly, inverting the default trust position would destroy the advantages of convenience and ease-of-use that have made email so popular — trust cannot be considered in black and white terms.

- There is a large supply of readily available test data in the form of the author's email archive and publicly available spam corpuses.

Two novel approaches using trust to solve the problem of spam will be described here. The first method attempts to solve the underlying problem of trust in the email system by allowing senders to use trust-based authentication in a manner that is compatible with existing email systems. The second method aims to improve the accuracy and power of spam detection and filtering using a collaborative peer-to-peer network.

7.2.1 Event Structure for Spam Applications

Both spam applications use the same event structure for the trust model. The decision is to mark a particular message as spam or allow it to pass into the user's inbox, and the outcomes are that the message is ultimately spam or not spam. This is shown in figure 7.1.

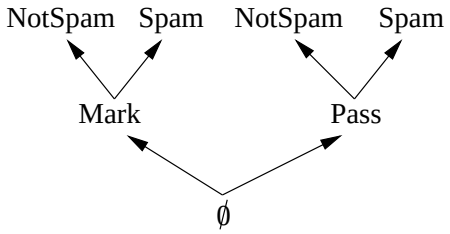


Figure 7.1: Event structure for spam filtering.

When used in conjunction with the model developed in the previous chapter, the "Spam" events contribute evidence to the probability that the state *spam* will obtain and the "NotSpam" events contribute evidence to the probability of the state *notspam* obtaining.

7.2.2 Combating Spam with Trustworthy Email Addresses (TEA)

One of the simplest anti-spam techniques is white-listing.[3] In this approach any legitimate email received has the contents of its From, To and CC fields added to a list of addresses from which mail is always accepted, therefore bypassing any other spam filter that may falsely label a real email as spam (a "false-positive"). In reality this method is one of the least effective for two reasons: firstly, the lack of authentication in the SMTP protocol [Kle01] means that addresses are so easy to spoof that many spams appear to come from a legitimate address, and secondly it makes it very difficult to establish a communications channel with a new person (or an old person using a new address).

The first problem can be overcome using some form of authentication method, as will be outlined below. The second is more difficult, but can be overcome using a recently proposed technique called *bankable postage* [ABB+03] which allows the sender of an email to attach a proof that guarantees that a certain cost has been incurred to obtain this proof.

Unfortunately whilst this is a technically feasible approach to solving the underlying problem of spam, namely the near zero-cost of sending it, how to set the minimal fee required to guarantee protection remains an issue. Additionally, using bankable postage imposes additional burdens on the sender which make it significantly less attractive to ordinary users than traditional email. This problem will be revisited later.

Making Email Addresses Trustworthy

The goal of TEA is to provide a sufficient protection against the large-scale spoofing of email addresses such that whitelists and legal measures may be employed to render spamming unprofitable. Previous systems to attempt this, such as PGP [ACGW99] and S/MIME [GMCF95], which are designed to run over top of the legacy system, have failed to gain large acceptance or solve the spam problem as they suffer from many usability issues in deployment, use and management. For example in "web of trust" style systems the users must validate keys out-of-band which is laborious, and while Certificate Authority (CA) schemes replace the need for individual users to check identities, the charges imposed by the CA act as a barrier to adoption.

Hence, since establishing the prerequisite binding between a key and real-world identity is too inconvenient for end-users, the TEA solution retains user-friendly plain-text email addresses (that are easily remembered and exchanged) and attempts to prevent spoofing by using a combination of two techniques: proof-of-knowledge of a shared message history and an automated proxy-based challenge-response (C-R) system. Both methods were implemented using the Java-based Claim Took Kit (CTK), developed by Trinity College Dublin as an implementation of their entity recognition framework [SJ05]. This allows additional information (a "claim") to be embedded into an email (like a conventional

[3]This section summarises work done in collaboration with Jean-Marc Seigneur, Ciarán Bryce and Christian Damsgaard Jensen, published in [SDBJ04].

PGP signature) that can then be used by the receiver's entity recognition process to determine the confidence in recognition. The proof-of-knowledge of a shared history was implemented using the CTK by including cryptographic hashes of recent (email) exchanges, although like most trust-based techniques it cannot be perfect as SMTP does not guarantee message delivery.

The second technique is to send a cryptographic challenge to the sender to confirm that he or she is the real source of the email. C-R systems that attempt to prove that the sender was human, or impose some "charge" on the sender (for example, proof-of-work puzzles) are already common tools in the fight against spam. These are widely (and rightly) criticised, but others assert that this is due to implementation and usability faults rather than being fundamental flaws with the principle [Tem]. The most common fault is so-called *collateral spam* where a spoofed sender address leads to an innocent third party being bombarded with challenges (and if the challenges include any part of the original email, allow the spammer to simultaneously spam the third party). However, TEA does not use C-R in either of these traditional forms, but is instead designed to establish a level of trust between the sender and receiver's mail clients — since the responses are handled automatically by the program, a human should never have to interact with a challenge. Challenges are only sent to users who have signalled their ability to participate in the protocol by including a claim in their email, thereby preventing collateral spam. Note that whilst the hashes technique can only be used with principals with whom the user has previously corresponded, C-R can be used with any TEA user.

TEA also supports asymmetric (public-key) cryptographic signatures as an additional method of identity assurance. However, unlike traditional methods such as PGP, there is no need to establish a binding between the key and a real-world identity, the key need only be bound to an email address for which the user has already established a trust-relationship. This relationship can be created in a variety of different ways: out of band, using the SECURE framework (as described below), or the TEA C-R method as described above.

The three techniques described above differ in strength, although there is no clear ordering as to which is better. For example, in the case of a valid signature with a short key length versus the ability to show that the sender is able to receive emails sent to a specific address, the key may be mathematically better but this makes many assumptions about the competence of the user with regards to key management. Keys are also inconvenient when one is away from one's home network, and allowing remote access to a key correspondingly reduces its security. Hence the TEA system allows the sender to choose zero (for legacy system compatibility), one, two or more combinations of techniques. The ER module then computes an anti-spoofing (or entity recognition) confidence level based on the number of techniques used and to what level.

Stopping Spam

Having established a trust level in the (pseudonymous) identity of the email sender, a trust assessment can then be made as to whether the sending principal is a spammer or not.

If the user has interacted with the sender before then simple black- and white-listing can be used. Otherwise the SECURE framework can be used to obtain recommendations about the trustworthiness of the principal. In the current prototype this is done by fully trusting the recommendations of a predefined number of *friends*, since that part of the framework was not complete when this work was done. Recommendations are also exchanged using the SMTP protocol (that is, are sent as special emails that the proxy will hide from the user) and authentication of their origin performed using TEA. Failing finding a suitable recommendation, the system falls back to using the *bankable postage* scheme mentioned above. The greater the postage paid, the greater the trust in the message being legitimate. If no bankable postage is attached then the final fallback is to use a traditional filtering system, such as SpamAssassin.[4]

Once a trust value for the sender has been established, the access control manager can make a decision as to whether to *mark* or *pass* the message, as described in earlier chapters.

Evaluation

Since this work concentrated on the entity recognition and risk aspects of the SECURE project the trust model is very simple, with users being either trusted or untrusted and only very limited recommendations permitted. The use of black- and white-lists for actual spam(mer) detection means that "convergence" takes just one (direct) interaction, although clearly the limitation on recommendations means that the rate of propagation is suboptimal. Sybil attack protection is provided by the use of bankable postage to cause newcomers who are not recommended by a user's friends to pay an entry fee, in line with the solutions to the problem proposed in [FR01].

The disadvantage of this highly simplified system is that it operates at a very coarse level of granularity: principals are either spammers, legitimate senders or unknown, and so the decision to mark or pass the message is an elementary one. Similarly this means that an attack need only successfully compromise one user to become trusted (although each successful compromise will only permit the sending of one spam). On the positive side, this does bring the advantage that the policy is very easy to use — the user simply has to specify the E value (see section 4.3.2), perhaps using a slider widget, which determines how aggressive the filtering should be.

A system level evaluation and further details are given in [SDBJ04].

7.3 Collaborative Spam Detection

Without some form of sender-authentication system like the one proposed above, it is difficult to make trust assessments about the sender of a message. As noted previously, forgery of existing identities and the creation of new ones is too easy with the current Internet email infrastructure, making trust assessments unreliable. The difficulty of migrating the millions of email users

[4] `http://spamassassin.org`

in the world to any new system also suggests that a piecemeal transition will be required, and so spam-filtering by the receiver will remain necessary for some time yet.

Collaborative spam detection is a new idea that takes the view that since identifying spam is an AI problem that can never be entirely solved using rule-based systems, even advanced ones based on Bayesian inference [Gra02], the best method of identifying spam is still a human. Therefore the first human to identify a spam publishes a hash of the message to a peer-to-peer (p2p) network and then each member of that network compares their incoming email with the published hashes, as shown in figure 7.2. A trust and risk analysis is used to determine whether to mark the message as spam or allow it to continue into the user's inbox, given the opinion of other trusted nodes on the p2p network. Since spam emails are increasingly obfuscated in an effort to defeat filters, straightforward hashing techniques will not be effective, a problem which will be addressed below.[5]

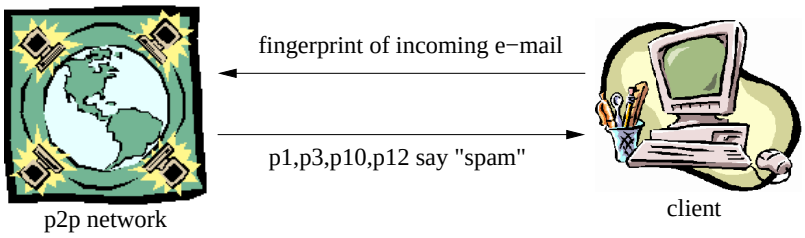


Figure 7.2: Overview of the operation of a Collaborative Spam Detector.

Similar systems are Vipul's Razor[6] and [ZZZ+03]. The former, which pioneered the idea of collaborative spam detection, uses a network of servers under central control and so is not truly distributed in the p2p sense. The latter uses the Tapestry p2p routing architecture and voting to identify malicious nodes, but relies on centralised identity management to prevent attacks.

7.3.1 Objectives and System Architecture

In order to evaluate the effectiveness of the application as a whole, the following objectives were decided upon:

Privacy and obfuscation: Collaborating with others requires the sharing of information, but the contents of a user's email must not be revealed. In addition, the system must overcome attempts by spammers to obfuscate the content of their message.

Efficient network usage: In order for people to contribute to the network it must generate a minimal amount of traffic and avoid consuming a large proportion of the user's available bandwidth.

[5]This collaborative spam detection application extends earlier work with Ian Maddison and David Ingram, published in the ACM Crossroads magazine [DM04] and at the iTrust 2005 conference [DBIM05].

[6]`http://razor.sourceforge.net`

Trust: The Internet is an untrusted environment, but information provided by others is the primary tool a collaborative system uses to filter messages. The reliability of this information must be considered.

Security: Spammers must not be able to subvert the p2p network as that would permit them to render the whole system ineffective.

Interface: The system must work with as many existing email clients as possible. A system requiring a custom mail client is not acceptable as that would represent a significant barrier to its adoption by the Internet community.

On a p2p network all nodes are equal and must act as both client and server. In this system, the client:

1. collects and parses email messages from the mail server;
2. searches the network, querying nodes for their conclusions on similar messages;
3. receives responses from these nodes and collates them;
4. invokes the SECURE framework to determine whether to mark the message as spam or not;
5. redelivers analysed email messages to the user's normal email client.

The server listens for queries from other nodes and answers them with conclusions it has reached on similar messages. When analysis is complete and a conclusion is reached, it must be published so the server can answer queries. If a user disagrees with this conclusion the published information will then be updated.

Both client and server were implemented in Java, using the Javamail and Apache XML-RPC libraries. An overview of the system architecture is given in figure 7.3

7.3.2 De-obfuscating Messages

Automated comparison of spam messages is made difficult due to the obfuscation techniques spammers use to evade existing filters. A detailed examination of spam messages received over a one week period showed the following common obfuscation techniques:

- appending a collection of nonsensical, but actual English words at the end of messages (e.g., "coco divisive stutter epilogue cloister dividend knapsack elfin ambidextrous lombard renegotiable");
- inserting random spaces, characters, punctuation, and new lines (e.g., Mor tg age, via-gra);
- substituting alphabet characters for similar visual representations such as accented characters, numbers or HTML entities (e.g., pi11);

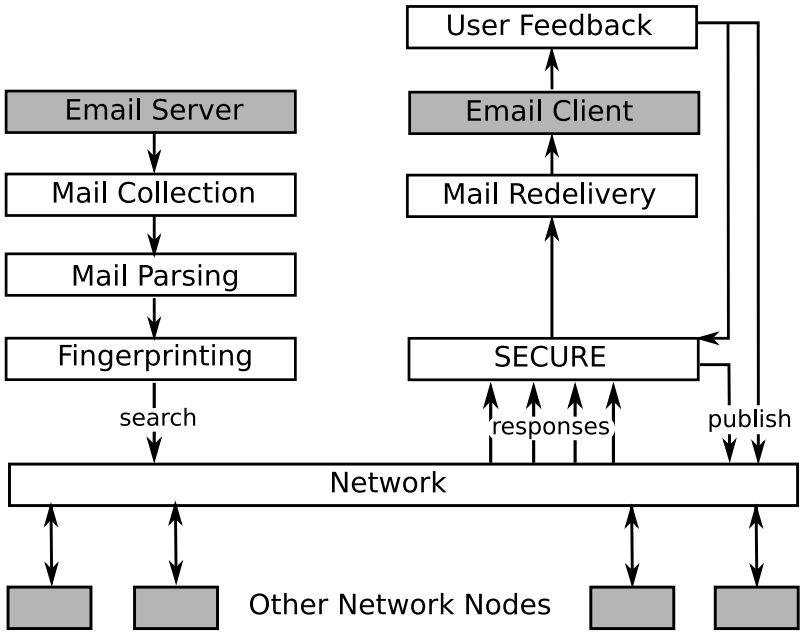


Figure 7.3: The System Architecture for the P2P Collaborative Spam Detector.

- inserting invalid HTML tags into words that will not be displayed by the majority of email clients;
- embedding an image which includes the actual content of the message.

Therefore, to find whether the same spam message has been received by other users, the system must search for messages that are the same but have been obfuscated differently. The most straightforward method for achieving this is to search for similar (not just identical) messages, thus making the problem one of approximate matching rather than de-obfuscation.
Two main techniques for approximate matching were identified:

1. Fuzzy hashing algorithms such as *nilsimsa* [cme04]. Unlike "secure" hash functions, these produce similar hashes for similar input. The similarity of two inputs is then given by the hamming distance between the two nilsimsa hashes.

2. Digest or feature vector approaches such as [ZZZ+03]. These find all possible consecutive substrings of fixed length L ("fingerprints" or "digests") from a document and use these as the feature vector of the document to be published or located in the p2p network.

Both techniques were evaluated for their viability. Using the nilsimsa approach, for 2259 real messages and 92300 spam messages (100 variants of each of 923 original messages) the average hamming distances were as follows:

Between two variants of the same spam:	51 bits
Between two different spams:	104 bits
Between two real emails:	102 bits
Between a real email and a spam:	112 bits

This seems promising, but in practice each message must be compared with *all* nilsimsa hashes found on the p2p network. This makes the problem much harder as a hamming distance that achieved a 75% detection rate also caused 3% of false positives, which is unacceptable.

[ZZZ+03] uses the feature vector approach, claiming a 97% success rate with no false positives. Accordingly this was the technique chosen for this application.

7.3.3 Networking

The bandwidth-usage of the system will primarily be determined by the implementation of the peer-to-peer network. The network must also be robust to ensure that a node unexpectedly leaving the network does not produce errors or lead to network fragmentation. There must also be no single point of failure that could be attacked by the spammers. Three possible network topologies were considered.

Flat: The simplest form of topology is an unstructured network. All nodes are considered equal, so organisation is easy, but load-balancing and tolerance to nodes leaving the network is harder to achieve. Searching is also highly inefficient and scales poorly with the size of the network since without any structure queries must be performed using brute-force flooding.

Hierarchical: A more complex topology involves having a number of levels, for example the two-tier FastTrack system used in popular p2p file-sharing clients, or a more traditional tree structure. Organising the structure is more difficult (for example, who decides which nodes go in which tier), although load-balancing and fault-tolerance are easier to manage. Searching also becomes more efficient and, depending on the structure chosen, can be made scalable.

Distributed Hash Tables: DHTs are a distributed form of the hash table data structure, where data can be looked-up directly by computing a *key* value from the data, which is then used to locate the node responsible for that data. This allows the creation of scalable, self-organising, load-balanced, and fault-tolerant structured overlay networks that permit efficient and scalable searching. Removing information from current systems, such as Chord [DBK+01] and Pastry [RD01], is more difficult, but that is not a problem in this application.

The Approximate Text Matching algorithm used in [ZZZ+03] is based on the Tapestry overlay network. Tapestry is not a DHT but a Decentralised Object Location and Routing system — essentially a decentralised directory service abstraction [DZD+04]. Chord provides both the DHT and DOLR abstraction and it was therefore decided to use that in this implementation.

7.3.4 Trust and Identity

As discussed in chapter 2, the simplest form of entity recognition is to represent each principal by a public/private key-pair. This was the method chosen for

this application in order to make evaluation of the other parts of the framework more straightforward. The key-pair is generated the first time a node joins the network. The private key is used to cryptographically sign published email fingerprints; other nodes can then confirm the authenticity of the publisher using the widely-known public key.[7]

The revised SECURE trust and access control models described in chapter 6 were used for this application. Each client locally stores a (s_j, i_j, c_j) triple for each principal, p_j, indicating how much they trust their judgement. s_j is the number of times they have given a correct opinion on whether an email is spam or not, c_j is the number of times they have incorrectly described an email and i_j is the number of opinions received that have yet to be confirmed by the user as correct or incorrect.

The probability that a mail is spam given that p_j says it is spam is then given by:

$$\rho_j = p(\text{spam}|p_j) = \frac{s_j}{s_j + c_j}$$

The probability that a mail is spam given that p_j says it is **not spam** is given by:

$$\rho_j = p(\text{spam}|\overline{p_j}) = \frac{c_j}{s_j + c_j}$$

The information from each principal is then weighted based upon the number of previous interactions the decision-maker has had with them, and a weighted-mean used to determine an overall probability of whether a message is spam or not:

$$p(\text{spam}) = \frac{w_1\rho_1 + w_2\rho_2 + ... + w_j\rho_j}{\sum w_j}$$

where $w_j = s_j + c_j$.

Information from principals with w_j below the information threshold, l, is ignored.

Recommendations

If there is insufficient information in the local trust database about principal Alice, who offers an opinion on a particular email, then the trust engine may also query the p2p network for recommendations from other principals about their experiences with Alice. "Insufficient information" is determined by the information threshold, l. To avoid problems with second-hand information and loops, only information from direct experiences is published, in the form of (s, i, c) triples.

Clearly, naïvely trusting recommendations from other p2p users is dangerous. In order to incorporate these recommendations into the probability calculation, they are first discounted [Jø01], using the belief in the principal's *recommendation integrity*; they are then averaged over all principals who supplied a recommendation, before being added to any local trust information.

[7] A note on terminology: To avoid any confusion, in this section all **principals** are nodes of the p2p network, where a principal is defined as the holder of a particular public key.

Recommendation integrity (also known as "meta-trust") is calculated using the concept of *semantic distance* [ARH00]. Received recommendations are stored in a cache and then, after a certain number of interactions with the subject of the recommendation have taken place (arbitrarily chosen as five in the initial prototype), the value of the (newly obtained) local *observed* trust value is compared with the *received* trust value. The main difficulty here is that the information content of the received value is likely to be far higher than the experience value. The two trust values are therefore normalised, giving (where $\text{info} = s + i + c$):

$$\left(\frac{s_o}{\text{info}_o}, \frac{i_o}{\text{info}_o}, \frac{c_o}{\text{info}_o}\right), \left(\frac{s_r}{\text{info}_r}, \frac{i_r}{\text{info}_r}, \frac{c_r}{\text{info}_r}\right)$$

The recommendation integrity, RI, is then given by the ratio:

$$\frac{s_o}{\text{info}_o} : \frac{s_r}{\text{info}_r}$$

It is observed that if $RI < 1$ then the recommendations tend to be better than the behaviour experienced, and so any recommendation should be scaled down accordingly. In contrast, if $RI > 1$ then the recommendations are more negative than observations indicate. Note, scaling-up does not occur in this instance as it would break monotonicity and allow an attacker to manipulate trust values by giving negative recommendations about those it wished to make appear more trustworthy. Instead it is asserted that anyone who is guilty of "bad-mouthing" another principal is unlikely to be very trustworthy at all and thus all recommendations from principals for whom $RI > 2$ are ignored.

The RI for a recommender is calculated at most once per recommendation received, to ensure that principals are not overly penalised if the principal they recommend subsequently changes their behaviour. Since principals are likely to make many recommendations, the RI value for a principal is an average over all the recommendations that it makes.

7.3.5 User Interface and Feedback

Most of the time, the p2p client program runs unobtrusively in the background, acting as a proxy between the user's mail client and their mail server, collecting the user's email and delivering it to their inbox or spam folder as appropriate. However, if the system makes a mistake then the user requires a mechanism through which they can provide feedback. A footer, like the one shown in figure 7.4, is appended to each message. Since the majority of email clients load clicked URLs in the user's web browser, the p2p client also acts as a small webserver running on the user's machine and listens for such page requests — requesting a page associated with a recently processed email is taken to mean that the user disagreed with the decision that was made. Since the software runs as a proxy that may be used in conjunction with any mail client it thus meets the *interface objective* set out in section 7.3.1.

For each message that is processed, a cache is maintained of:

$$(fingerprint, timestamp, key\_id, opinion)$$

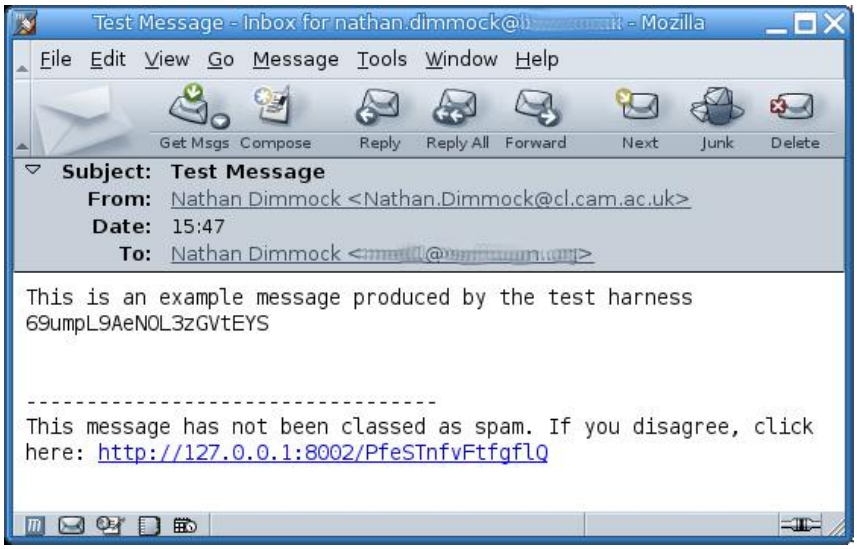


Figure 7.4: Example email which has not been marked as spam.

tuples for each key_id (principal) that published an opinion to the network. The timestamp records the date and time when the message was received and processed by the p2p client.

Since feedback is only received when a wrong decision is made, and there is no method of discovering when a user has seen a particular email to confirm its correct classification, classification is assumed correct after a fixed time period (configurable by user). When feedback is received, the appropriate tuple is removed from the cache, any trust updates that have already taken place reversed, new trust values computed and the email fingerprint published to the network, along with the user's opinion. In this way, only users who found the existing information on the network incorrect publish an opinion, thereby minimising the number of updates.

7.3.6 Evaluation

The SECURE framework elements were evaluated independently of the remainder of the application and implemented in the Python language, shown in figure 7.5. An abstract class was provided to allow the TBAC components to interface with a number of different DHT implementations, including a "null" implementation that was used for the tests described here.

Behavioural Attacks

Three main classes of principal have been identified.

Honest: Always give accurate reports. Required trust-value is any (s, i, c) triple where $s > c$.

Naïve Spammer: This principal classifies all mail as spam. Since reports indicate over 50% of email on the Internet is spam, this means they will

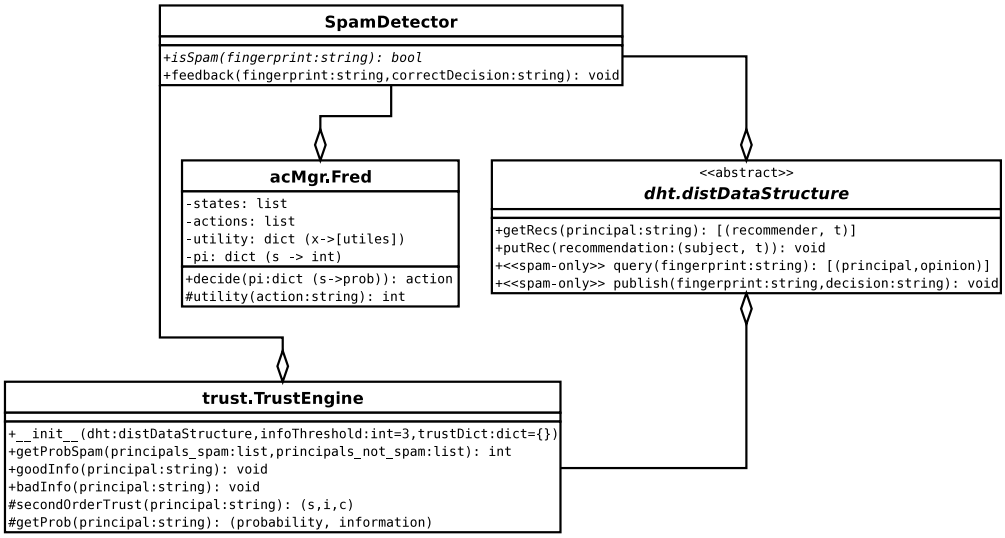


Figure 7.5: UML overview of the collaborative spam detector implementation.

be accurate 50% of the time, and therefore able to leverage that reputation to misclassify real mail as spam. Required trust-value is any (s, i, c) triple where $s < c$.

Clever Spammer: This principal accurately classifies mail from other spammers (to increase their reputation), but labels spam they send out as legitimate email. Required trust-value is any (s, i, c) triple where $s < c$.

Random: This principal indiscriminately marks mail as spam or not spam. This represents an attacker attempting to reduce the quality of the information in the network to the point where the system is useless. Required trust-value is any (s, i, c) triple where $s < c$.

For n emails, analysis shows that the *honest* principal will obtain the desired trust value of $(n, 0, 0)$. The *naïve spammer* will obtain $(\alpha n, 0, (1 - \alpha)n)$ where α is the proportion of emails that are spam. If $\alpha > \frac{1}{2}$, which as noted earlier is now thought be to the case for Internet email, then clearly this will not meet the $s < c$ requirement. The *clever spammer* will obtain a similar trust value where α is the proportion of emails that are not their own spam. Whilst correctly classifying arbitrary emails is difficult, the system can be used against itself by echoing the opinion of other principals. Therefore this principal type will also be assigned the wrong trust-value. Finally the *random* principal will, for large values of n, be assigned a trust-value of approximately $(\frac{1}{2}n, 0, \frac{1}{2}n)$ which is also not what is required.

Reputation-based Attacks

The potential principal types are as described in section 7.1.2, namely:

Honest: Give truthful recommendations. $RI \approx 1$.

False-Praisers: Give false positive recommendations. $RI \approx 0$.

Defamers: Give false negative recommendations. $RI > 2$.

Liars: Give both false-praise and defamatory recommendations. $RI < 1$ or $RI > 2$.

Randoms: Give out randomly generated recommendations. $RI \approx 0$ or $RI > 2$.

The required RI for the final two principal types is difficult to determine. *Randoms* are slightly easier since it may be assumed that randomly generated recommendations will be so far from the truth that RI will converge to either zero or greater than two, but there is no guarantee of that. It is supposed that *liars* are different from *randoms* in that they exhibit intelligence in their choice of recommendations, for instance by issuing recommendations that are the inverse of the true state. Analysis suggests that a resourceful attacker would be able to give out recommendations such that the averages of their incorrectness would cancel, and their RI would tend to one — clearly a highly undesirable outcome.

The fact that this model for recommendation integrity proves to be vulnerable to the resourceful attacker successfully demonstrates the inadequacy of trust metrics consisting of a simple scalar value.

Improving the Trust Model

The analysis described above demonstrates the weakness of the revised SECURE trust model in its default form. Defence against the behavioural attacks can be improved by making a number of simple changes.

1. Changing the trust update function so that c increases at a faster rate than s. For example, every wrong decision causes c to be increased by $+2$ instead of $+1$.

2. Only taking the opinion of those principals for whom $(s - c) > l$ where l is the information threshold.

3. Introducing an upper limit on the number of incorrect decisions a principal can make before being ejected from the network, that is introduce a threshold, c_{max} and ignoring all principals for whom $c > c_{max}$.

These measures aim to make it more difficult for an attacker to maintain its positive trust value without contributing useful information to the network. For instance, the first and second measures combined mean that in addition to having to perform l correct transactions in order to become established in the network, the attacker must continue to correctly identity two pieces of spam for every one it claims is legitimate. The third improvement ensures that an attacker using this strategy is eventually removed from consideration, and that the l correct identifications must occur sufficiently early that simply publishing

random responses carries a significant risk of being permanently ejected from the network.

Note that the second and third changes suggested here are actually changes to the policy rather than the trust metric, which demonstrates the value of evaluating the system in an end-to-end fashion rather than just by component.

Sybil Attack

The improvements to the trust model outlined above already provide a fairly robust defence against a Sybil attack, since over the lifetime of an identity, every principal is constrained to provide more useful information than bad. Further, if a principal exceeds c_{max} then the attacker has to generate a new identity and start again, publishing l correct entries.

The fact that information is published to a DHT gives some more protection from Sybil. If users contacted principals directly for information then an attacker could tailor their responses to manipulate their trust value, for instance by providing correct information the first l times and then changing their identity after c_{max} interactions. However, because information is published globally to the network, every requester will be served the same response and the attacker cannot control the order in which responses are seen.

This still leaves the related problem that attackers could attempt to subvert the network by publishing large numbers of opinions (under different identities) in the hope that some of them will prove sufficiently accurate to obtain a positive trust value. Since legitimate users should publish infrequently (only when their spam detector makes an incorrect decision as described in section 7.3.5), limiting the rate at which nodes can publish to the p2p network is required.

Since an attacker may easily create a large number of principals but can only control a finite amount of computing resource, this rate limiting can be achieved using a proof-of-work algorithm or puzzle. These come in two main forms: "Turing tests" to confirm that the operator is human, and computational puzzles that require the expenditure of a certain quantity of resource such as CPU cycles, memory or bandwidth [SL03]. Use of the former would limit the publication rate of an attacker to the rate at which its employees could solve the puzzles; the latter would limit the number of principals an attacker could run on each machine under its control.

Risk and Access Control Evaluation

The risk and access control components were evaluated using a 1000 message simulation (with 50% of messages being spam) and 100 principals. A principal is introduced into the current round with probability $\frac{100}{1000} = \frac{1}{10}$. Spam messages were sent to all principals currently in the network, non-spam messages were sent only to the user being simulated. No trust information (recommendations) is shared between principals in this initial test.

Three parameters were varied: the percentage of malicious principals, the number of collectives controlling the malicious principals and the information threshold l. The collectives model the *clever spammer* attacker-type described

above: a malicious principal will mark spam messages from their own collective as real whilst correctly identifying spam messages from other collectives. The averaged results from three simulation runs are shown in figure 7.6.

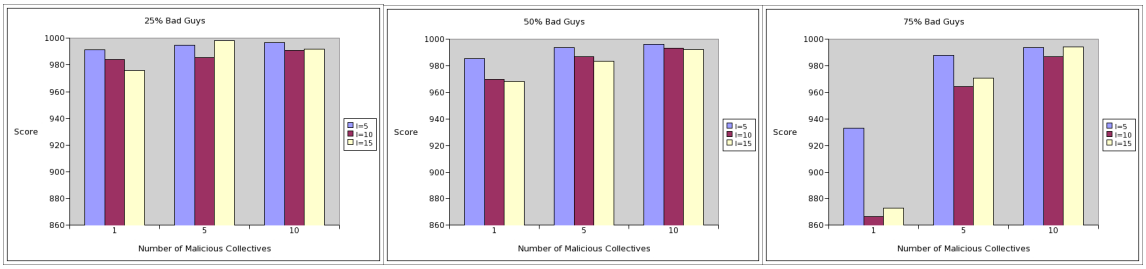


Figure 7.6: Graphs showing the effectiveness of the spam detector (score out of 1000) depending on the percentage of malicious principals, the number of collectives into which the bad principals are organised and the information threshold l.

As expected, the system performs better when the attackers form a minority of the p2p nodes. However, even with three-quarters of the principals being malicious, the system is still effective, especially when there are many competing collectives of principals. A more surprising result is that higher information thresholds actually decrease the performance of the filter. It is hypothesised that this is because the information threshold is a double-edged sword: it takes longer for malicious principals to gain sufficient trust to be listened-to, but it also takes good principals who are giving accurate information longer to become trusted. This suggests that the dynamics of trust must be asymmetric with a slow-increase and fast-decrease, to which the SECURE trust model conforms when using the extensions described above.

Another interesting observation is that an attacker's best strategy is to cooperate as a single collective rather than trying to bolster their own trust rating by identifying rival's spam. It would seem that the other collectives, using the same strategy, cancel out the effect of any positive trust rating the attacker might have.

Figure 7.7 shows when in the course of the simulation the errors are made — namely during the early rounds before a network of trusted peers has been identified. This graph only shows a sample of results, the simulations where fewer errors were made did not produce interesting traces, but the lines shown confirm the conclusions drawn from figure 7.6.

Access Control Evaluation Using Recommendations

The evaluation simulator was then modified to allow the good principals to share information between themselves using recommendations. In this application recommendations from others are only sought if the information about a particular principal is below the information threshold, l. There is also an update threshold (ut) which specifies how frequently (that is, after how many interactions with the subject of a recommendation) should witnesses' recom-

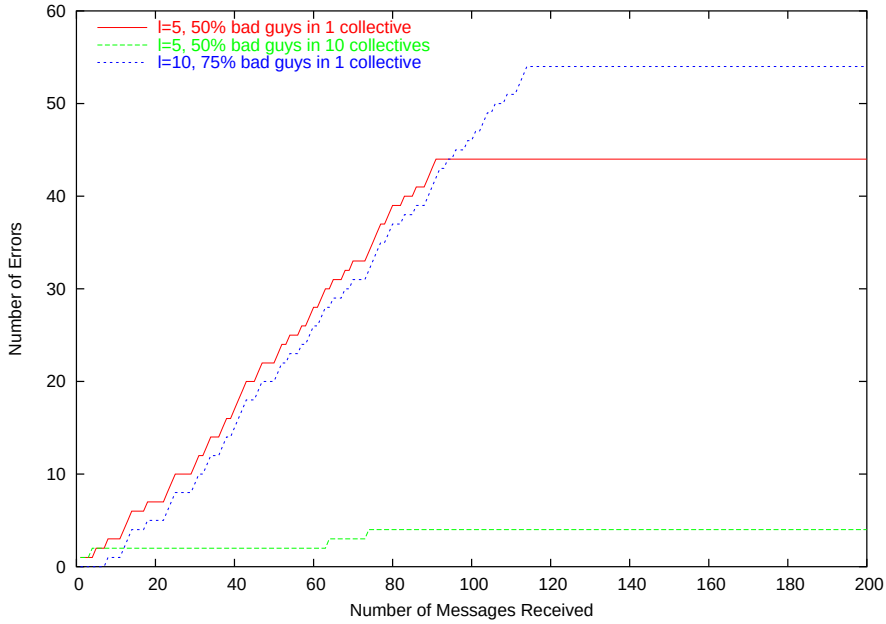


Figure 7.7: Cumulative number of errors made by the spam detector over time (since messages are received sequentially).

mendation integrity be updated.

To clearly illustrate the effect, the percentage of malicious principals and number of collectives were chosen from the previous results to give the most errors (that is, 75% bad guys and one collective). The update threshold was added as an additional parameter, with values of 1, 5 and 10. The DHT was populated with recommendations by running the first 500 messages of the simulation before initialising the test client. A number of defamatory recommendations were also inserted into the DHT.

Since the test client was joining an established network it was observed that the number of errors it made was limited to l — effectively the number of spams it had to see before learning which nodes to trust. Figure 7.8 shows how the use of recommendations can be used to reduce the number of errors to less than the information threshold l. An interesting feature of this graph is that smaller update thresholds do not necessarily lead to fewer errors. It is hypothesised that this is because RI is calculated as a moving average and therefore less frequent updates means larger changes that cause quicker convergence.

Security Threat Analysis, Ease of Use and Expressiveness

Having evaluated the trust components of the application, it is important to put this into the context of the traditional security problems computational trust systems are trying to solve.
Potential threats to this system are:

- impersonation of other principals;
- deliberate return of invalid or incorrect data as a response to a query;

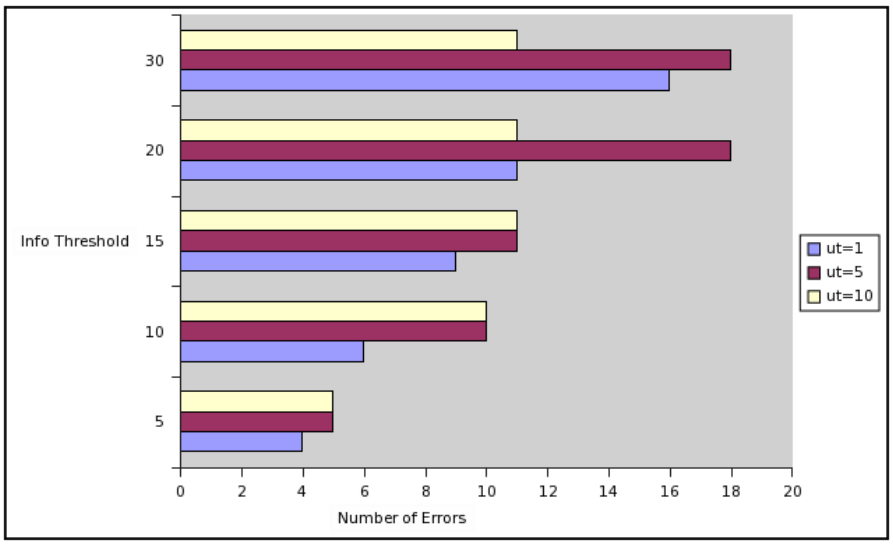


Figure 7.8: Effect of recommendations on number of errors.

- attempts to disrupt the network/denial of service attacks.

As described earlier, impersonation attacks were defended against using public/private keys and digital signatures. Likewise, deliberate propagation of misinformation is defeated through the use of SECURE for trust-based collaboration and decision-making.

By distributing the database of spam information using a peer-to-peer model such as Chord, denial of service attacks against the network as a whole are much more difficult than when the information is concentrated in a small number of well-known servers. However, denial of service attacks involving packet flooding directed against a particular node cannot be prevented at the application level as this is a security flaw to which all Internet applications are susceptible.

As noted earlier, ease of use and expressiveness are also important aspects of any system, and often impact on the overall security of the system. The risk model used in this application is very easy to use, requiring the user to specify just one parameter, E, the relative cost of false positives compared with identifying spam. The system allows for more expressiveness if required: the number of acts could be extended to prioritise messages depending on how likely they are to be spam, and the user could be allowed to configure the full preference-scaling function if they desired.

7.3.7 Conclusions from the Collaborative Spam Detection Application

The development and evaluation of the collaborative spam detection application has highlighted a number of important results for the SECURE project and trust-based decision making systems. Firstly, the analysis of the evidence-based trust model demonstrated the need for asymmetric trust dynamics, and the importance of a point of no redemption, or history-fading. Without these,

principals may initially perform a small number of positive interactions and then use the "credit" obtained to behave badly. A second result concerning the trust model is the need for a better method of representing recommendation integrity than a single scalar. Although it proved difficult to exploit the weakness in this metric in practice, it is still theoretically a vulnerability of the system.

Overall this application also demonstrated the utility of a trust- and risk-based decision-making engine for collaborative applications. The particular model used here is applicable to a number of variations of this application where it is possible to evaluate the quality of all the information given by principals. For example determining the quality of files in p2p file sharing networks, or entries in a collaborative database of compact disc meta-data such as freedb.org. In the latter case, a trust-based system would allow users to favour entries written by authors they had previously found to be sources of accurate and consistently formatted entries.

In contrast, the instantiation of the model used here would not be applicable to scenarios such as choosing a service provider since it is impossible to observe the results of those principals that were not chosen. Accordingly, although it reduced the number of interactions needed to find trustworthy peers in this application, the exchange of trust information was determined to be only marginally useful. Conversely, it is expected that such exchanges via recommendations would have a larger rôle to play in applications of this other type.

CHAPTER 8

Conclusions and Future Work

Global Computing is the vision of a massively networked infrastructure supporting a large population of diverse but cooperating entities. Similar to ubiquitous computing, entities of global computing will operate in environments that are dynamic and unpredictable, requiring them to be capable of dealing with unexpected interactions and previously unknown principals using an unreliable infrastructure.

These properties lead to new security challenges that are not adequately addressed by existing security models and mechanisms. The size of the global computing infrastructure means that security policy must encompass billions of potential collaborators and enemies. Mobility, and the possibility for disconnection from one's home network, requires the ability to make security decisions autonomously in an environment where identity conveys no *a priori* information about the likely behaviour of the principal, precluding the use of many existing access control systems.

The essential challenge of access control in these environments is to balance **trust** against **risk**: enabling access to resources for authorised principals is as important as providing protection from unauthorised ones. Previous research has shown how to quantify and model trust, yet the relationship between trust and privilege is unexplored. What does it mean to trust Alice to the n^{th}-degree? It is the thesis of this dissertation that the relationship between trust and privilege must be moderated using risk; in a dynamic environment, contextual risk information can transform the question of trust to the n^{th}-degree to one of whether the risk of trusting is beneficial to the entity taking the decision.

This dissertation proposes a model for quantifying and reasoning about risk in global computing. It then shows how this risk model can be integrated with trust and privilege to provide **trust-based access control**. To recap from chapter 2, the aim of TBAC is:

> For a given application domain, a computational trust system (such

> as SECURE), should be able to select an interaction partner (or partners) such that the risk of interacting with that principal is deemed to be acceptable to the decision-maker.

To meet this aim it is necessary to consider the trustworthiness of the principal(s) involved and the risk of entering into the interaction. In determining these factors there will be other risks: using scarce resources (such as network, CPU, energy or actual money) to establish trust (or distrust) may not alter the risk assessment sufficiently to justify the use of those resources. This dissertation shows how policies can be constructed to allow agents to make decisions autonomously based on these risk factors.

8.1 Summary of Research Contributions and Results

PDA Information Sharing: This dissertation began by developing a simple, but novel, access control model for information sharing between Personal Digital Assistants (PDAs) using trust and risk.

Risk Modelling: The work on information sharing in a ubiquitous computing scenario led to the analysis of risk in a variety of fields, including computer science, the social sciences, economics, and the insurance industry. This research produced a quantified formal model of risk for use in global computing.

Extending RBAC: It was then shown how trust and risk modelling could be incorporated into existing rôle-based access control (RBAC) models, such as OASIS.

Trust-Based Access Control: A general model of trust- and risk-based access control was then proposed and implemented, based on the previously developed risk model and using the economic theory of decision making. Although designed principally for use with the SECURE framework, it was also demonstrated how it may be used with trust models other than SECURE. A key benefit to the SECURE approach is that explicit modelling of uncertainty and information is carried all the way through the decision-making process, enabling the access control policy to reason about metrics of quantity and quality of the information, as well as the information itself.

TBAC Architecture: The SECURE project provides a general framework for reasoning about trust and risk. This thesis shows how this framework can be instantiated into a more concrete architecture, demonstrating how the trust, risk, entity recognition and evidence management components would interoperate to allow the fulfilment of the stated aim of this thesis and permit trust- and risk-based decision making. It also explores a number of issues such as the risk of establishing trust through recommendations, and the risk of providing trust information to other principals.

Evaluation Methodology for Computational Trust Systems:
Langheinrich noted that [Lan03]:

> The currently developed solutions [using trust for access control in ubiquitous computing] make validation seem impossible, simply because authors never describe what constitutes a successful operation of the system.

This concern was echoed by the SECURE project's Industrial Advisory Board and project reviewers. This thesis addresses the problem by developing a comprehensive methodology that permits the evaluation of SECURE, and the work of this thesis, with respect to the aim outlined in section 2.5 and recapped above. Crucial to this is the definition of what it means to *attack* the system and thus what is considered malicious behaviour.

A prototype implementation of a peer-to-peer collaborative spam detection application allowed the detailed validation of the TBAC architecture using the developed evaluation methodology. This led to a number of results concerning the work described herein, and other aspects of the SECURE framework.

- The utility of trust-based systems in a global computing environment was established, since without trust the collaborative spam detection network could be subverted by peers operating on behalf of the spammers.
- Analysis of the dynamics of the SECURE trust model with respect to various behaviour patterns demonstrated the need for trust to change asymmetrically, that is, decrease faster than it increases. Without this condition an attacker can retain privileges, gained as a result of previously acting in a trustworthy manner, while misbehaving.
- It was further observed that (good) behaviour a long time in the past was not an accurate indicator for the current disposition of a principal, and therefore there was a need for "reputation fading" or some point of distrust beyond which redemption was impossible.
- Analysis of the recommendation integrity system used in the collaborative spam detection system confirmed the inadequacy of simple scalar trust metrics and the need for the more expressive formal models of trust described in chapter 2.
- Despite the weakness in the recommendation integrity metric, it was successfully demonstrated how the conveyance of trust information using recommendations can be used to bootstrap trust systems and facilitate the establishment of trust.
- Finally, the benefit of an end-to-end, threat-based evaluation was highlighted by the way it was possible to counter threats to, and vulnerabilities in, one component (such as trust or privacy) by applying suitable policies to another.

8.2 Further Work

This thesis, and the SECURE project in general, have developed a powerful framework for enabling trust-based secure collaboration and decision-making in global computing. However, the use of computational trust systems in this manner is a new field, and thus this initial framework represents an excellent basis for further research in the area. A number of potential extensions to the framework and avenues for further research are outlined below.

One area that the framework does not currently address in great detail is the notion of **time**. One of the results of the validation in chapter 7 was the need for "reputation fading", and thus it is hypothesised that trust models could be improved further by the ability to take time into account. It was also noted in section 6.7.1 that some forms of untrustworthy behaviour can only be deduced by taking the timing of events into account when analysing the evidence. Other researchers have already begun extending the SECURE risk model to incorporate the notion of time [CM05].

Explicit modelling of time in the trust and risk frameworks will mean additional factors that must be reasoned about in the access control manager. Attaching a notion of age or freshness to information will also require either extending the existing system, or developing new and more complex methods for handling the information quality and uncertainty metrics.

The current SECURE framework caters for **policy evolution** through the notion of dynamic trust and risk assessments that change in response to the difference between the expected outcome of an interaction and the actual outcome(s). A further extension would be the ability to measure the effectiveness of current low-level policy at implementing higher levels of requirements, and automatically evolving it accordingly to better implement the higher level policies. This would be a significant step towards self-managing security systems.

Another possible avenue of research would be the application of the SECURE framework to the problem of **security usability**. Chapter 1 described how greater security always has a cost, and anecdotally it seems that the cost is often in the form of a lower system functionality and usability. The model presented in chapter 3 showed how, for a limited application, security could be transparent and pervasive by being built around a user's habits and practices, instead of being bolted-on later and therefore becoming an obstruction to existing practices. If computers are to become pervasive tools that may be used by everyone, in the same way that anyone may learn to drive a car without understanding how an internal combustion engine works, then security must also become pervasive.

A major problem in this area is the use of software downloaded from the Internet: currently proposed schemes for eradicating spy-ware and other "malwares" often include the notion that code must be signed by a trusted software provider before it may be installed. However, like the hierarchical public-key infrastructure schemes of chapter 2, the need to obtain a certificate from an agency that can perform suitable verification checks requires resources that will deter smaller companies and individuals from providing free or low-cost software. There is also the question of *risk*: an individual may be happy to give

up some privacy in return for being able to use a piece of software he or she otherwise could not — but the same user may also have another machine they use for financial and other personal data that they wish to be "secure". The ability to convey trust information about software packages downloaded from the Internet, for instance recommendations from friends or colleagues considered to be "computer experts", and the ability to have the computer determine whether a package's request for administrator access is an acceptable risk, could be an important step towards making computer security usable and pervasive.

Bibliography

[AB00] Judy Feldman Anderson and Robert L. Brown. *Risk and Insurance*. Number 1-21-00 in Study Notes. Society of Actuaries (USA), 2000.

[ABB+03] Martín Abadi, Andrew Birrell, Michael Burrows, Frank Dabek, and Ted Wobber. Bankable postage for network services. In *Proceedings of the 8th Asian Computing Science Conference (ASIAN)*, pages 72–90. Springer, December 2003.

[ACGW99] Mike Ashley, Matthew Copeland, Joergen Grahn, and David A. Wheeler. *The GNU Privacy Handbook*. Free Software Foundation, v1.1 edition, 1999.

[AD01] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM'01)*, pages 310–317. ACM Press, November 2001.

[Ada95] John Adams. *Risk*. UCL Press, 1995.

[AH00] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.

[AM04] Xuhui Ao and Naftaly H. Minsky. On the role of roles: from role-based to role-sensitive access control. In *SACMAT '04: Proceedings of the ninth ACM Symposium on Access Control Models and Technologies*, pages 51–60. ACM Press, 2004.

[And01] Ross Anderson. *Security Engineering*. John Wiley and Sons, Inc., 2001.

[App02] Ashton Applewhite. Getting the Grid. *IEEE Distributed Systems Online*, May 2002.

[ARH97] Alfarez Abdul-Rahman and Stephen Hailes. Using recommendations for managing trust in distributed systems. In *IEEE Malaysia International Conference on Communication '97 (MICC'97)*. IEEE, 1997.

[ARH00] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *Hawaii International Conference on System Sciences 33*, pages 1769–1777, 2000.

[BB04] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[BBK94] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of trust in open networks. In *ESORICS '94: Proceedings of the Third European Symposium on Research in Computer Security*, pages 3–18. Springer-Verlag, 1994.

[BDK+05] Ciarán Bryce, Nathan Dimmock, Karl Krukow, Jean-Marc Seigneur, Vinny Cahill, and Waleed Wagella. Towards an evaluation methodology for computational trust systems. In *Proceedings of the Third Annual Conference on Trust Management*, volume 3477 of *LNCS*, pages 290–305. Springer-Verlag, May 2005.

[BFK98] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. Keynote: Trust management for public-key infrastructures. In *Proceedings of the 1998 Security Protocols International Workshop*, number 1550 in LNCS, pages 59–63. Springer-Verlag, 1998.

[BFK99] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In Jan Vitek and Christian Jensen, editors, *Secure Internet Programming*, pages 185–210. Springer-Verlag, 1999.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, page 164. IEEE Computer Society, 1996.

[BLB03] S. Buchegger and J. Y. Le Boudec. A robust reputation system for mobile ad hoc networks. Technical Report IC/2003/50, EPFL-DI-ICA, July 2003.

[BLMR04] Arosha K Bandara, Emil C Lupu, Jonathan Moffett, and Alessandra Russo. A goal-based approach to policy refinement. In *Proceedings of the Fifth IEEE International Workshop on Policices for Distributed Systems and Networks (Policy 2004)*, pages 229–239, Yorktown Heights, New York, June 2004. IEEE.

[BMCO03] Jean Bacon, Ken Moody, David Chadwick, and Oleksandr Otenko. Persistent versus dynamic role membership. In *17th IFIP WG3 Annual Working Conference on Data and Application Security*, pages

344–357, Estes park, Colorado, USA, August 2003. IFIP, Kluwer, 2004.

[BMY01] Jean Bacon, Ken Moody, and Walt Yao. Access control and trust in the use of widely distributed services. In *Middleware 2001*, volume 2218 of *LNCS*, pages 300–315. Springer-Verlag, November 2001.

[BMY02] Jean Bacon, Ken Moody, and Walt Yao. A model of OASIS role-based acces control and its support for active security. *ACM Transactions on Information System Security*, 5(4):492–540, Nov 2002.

[BS04a] Moritz Y. Becker and Peter Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'04)*, pages 159–168. IEEE Computer Society, 2004.

[BS04b] Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management applied to electronic health records. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, page 139. IEEE, June 2004.

[CDKN04] Marco Carbone, Nathan Dimmock, Karl Krukow, and Mogens Nielsen. Revised computational trust model. EU IST-FET Project Deliverable, February 2004.

[Cha03] David W. Chadwick. An authorisation interface for the grid. E-Science 'All Hands' Meeting, September 2003.

[CM05] Daniel Cvrcek and Ken Moody. Combining trust and risk to reduce the cost of attacks. In *Proceedings of the Third Annual Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*. Springer-Verlag, May 2005.

[cme04] cmeclax. Nilsimsa codes. `http://ixazon.dynip.com/~cmeclax/nilsimsa.html`, November 2004. Accessed 22 November 2004 17:30 UTC.

[CNS03] Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A formal model for trust in dynamic networks. Research Series RS-03-04, BRICS, Department of Computer Science, University of Aarhus, January 2003. EU Project SECURE IST-2001-32486 Deliverable 1.1.

[CO02] David W. Chadwick and Alexander Otenko. The permis x.509 role based privilege management infrastructure. In *SACMAT '02: Proceedings of the seventh ACM Symposium on Access Control Models and Technologies*, pages 135–140. ACM Press, 2002.

[CSG+03] Vinny Cahill, Brian Shand, Elizabeth Gray, Ciarán Bryce, Nathan Dimmock, Andrew Twigg, Jean Bacon, Colin English, Waleed

Wagealla, Sotirios Terzis, Paddy Nicon, Giovanna di Marzo Serugendo, Jean-Marc Seigneur, Marco Carbone, Karl Krukow, Christian Jensen, Yong Chen, and Mogens Nielsen. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3):52–61, August 2003.

[DBE+03] Nathan Dimmock, András Belokosztolszki, David Eyers, David Ingram, Jean Bacon, and Ken Moody. Preliminary definition of trust-based access control model. EU IST-FET Project Deliverable, October 2003.

[DBE+04] Nathan Dimmock, András Belokosztolszki, David Eyers, Jean Bacon, and Ken Moody. Using trust and risk in role-based access control policies. In *SACMAT'04: Proceedings of the ninth ACM Symposium on Access Control Models and Technologies*, pages 156–162. ACM, June 2004.

[DBIM05] Nathan Dimmock, Jean Bacon, David Ingram, and Ken Moody. Risk models for trust-based access control (TBAC). In *Proceedings of the Third Annual Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 1–8. Springer-Verlag, May 2005.

[DBK+01] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with Chord, a distributed lookup service. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, pages 81–86, Schloss Elmau, Germany, May 2001. IEEE Computer Society.

[DDLS01] Nicodemos Damnianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The Ponder policy specification language. In *Proceedings of Policy 2001: Workshop on Policies for Distributed Systems and Networks*, volume 1995 of *LNCS*, pages 18–39. Springer-Verlag, January 2001.

[DHH+03] Boris Dragovic, Steven Hand, Tim Harris, Evangelos Kotsovinos, and Andrew Twigg. Managing trust and reputation in the xenoserver open platform. In *First International Conference on Trust Management (iTrust 2003)*, volume 2692 of *LNCS*, pages 59–74, Heraklion, Crete, Greece, 2003. Springer-Verlag.

[DKHP03] Boris Dragovic, Evangelos Kotsovinos, Steven Hand, and Peter R. Pietzuch. XenoTrust: Event-based distributed trust management. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pages 410–415, Washington, DC, USA, 2003. IEEE Computer Society.

[DM04] Nathan Dimmock and Ian Maddison. Peer-to-peer collaborative spam detection. *ACM Crossroads*, 11(2):17–25, Winter 2004.

[Dou02] John R. Douceur. The Sybil Attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, number 2429 in LNCS, pages 251–260. Springer-Verlag, 2002.

[DZD+04] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica. Towards a common API for structured peer-to-peer overlays. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, volume 2735 of *LNCS*, pages 33–44, Berkeley, CA, February 2004. Springer.

[EFGK01] P. Th. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. Technical report, EPFL Lausanne, 2001.

[EFL+99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonsen. *RFC 2693: SPKI Certificate Theory*. IETF, September 1999.

[Eye04] David M. Eyers. *Active Privilege Management for Distributed Access Control Systems*. PhD thesis, University of Cambridge, 2004.

[FJK+01] Tim Finin, Anupam Joshi, Lalana Kagal, Olga Ratsimore, Vlad Korolev, and Harry Chen. Information agents for mobile and embedded devices. *Lecture Notes in Computer Science*, 2182:264–286, 2001.

[FKÖD04] Alberto Fernandes, Evangelos Kotsovinos, Sven Östring, and Boris Dragovic. Pinocchio: Incentives for honest participation in distributed trust management. In *Proceedings of Second International Conference on Trust Management (iTrust 2004)*, volume 2995 of *LNCS*, pages 63–77, 2004.

[FM04] Irini Fundulaki and Maarten Marx. Specifying access control policies for XML documents with XPath. In *SACMAT '04: Proceedings of the ninth ACM Symposium on Access Control Models and Technologies*, pages 61–69. ACM Press, 2004.

[FR01] Eric J. Friedman and Paul Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[FSG+01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information Systems Security (TISSEC)*, 4(3):224–274, 2001.

[FU01] Gonçlo L. Fonseca and Leanne Ussher. History of economic thought website, 2001. Accessed 22 November 2004 20:55 UTC.

[Gam00] Diego Gambetta. *Can We Trust Trust?*, chapter 13, pages 213–237. Department of Sociology, University of Oxford, 2000.

[GMCF95] J. Galvin, S. Murphy, S. Crocker, and N. Freed. *RFC 1847: Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*. IETF, October 1995.

[GOJ+02] E. Gray, 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, Department of Computer Science, Trinity College Dublin, December 2002.

[Gra02] Paul Graham. A plan for spam. Online. `http://www.paulgraham.com/spam.html`, August 2002. Accessed 11 February 2005 13:28 UTC.

[Gra03] Patrick Gray. Spam now more than half of all received email. Silicon.com: `http://networks.silicon.com/broadband/0,39024661,10004430,00.htm`, June 2003. Accessed 2nd February 2005 17:38 UTC.

[GS03] Tyrone Grandison and Morris Sloman. Trust management tools for internet applications. In *Proceedings of First International Conference on Trust Management, iTrust 2003*, number 2692 in LNCS, pages 91–107, Heraklion, Crete, Greece, May 2003. Springer-Verlag.

[GSSS02] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste. Project Aura: Towards distraction-free pervasive computing. *IEEE Pervasive Computing*, 1(2):22–31, 2002.

[HP03] Joseph Y. Halpern and Riccardo Pucella. A logic for reasoning about evidence. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 297–304, August 2003.

[HR92] Jack Hirshleifer and John G. Riley. *The analytics of uncertainty and information*. Cambridge surveys of economic literature. Cambridge University Press, 1992.

[Ing03] David Ingram. Trust-based filtering for augmented reality. In *First International Conference on Trust Management (iTrust 2003)*, volume 2692 of *LNCS*, pages 108–122, Heraklion, Crete, Greece, May 2003. Springer-Verlag.

[Ing05] David Ingram. An evidence based architecture for efficient, attack-resistant computational trust dissemination in peer-to-peer networks. In *Proceedings of the Third Annual Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 273–288. Springer-Verlag, May 2005.

[ISO99] ISO/IEC JTC1/SC22 Working Group. *ISO/IEC 9899 – Programming languages – C*, 1999.

[Jø96] Audun Jøsang. The right type of trust for distributed systems. In *NSPW '96: Proceedings of the 1996 workshop on New security paradigms*, pages 119–131, New York, NY, USA, 1996. ACM Press.

[Jø01] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, June 2001.

[JGK03] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing topologies of transitive trust. In *Proceedings of the Workshop of Formal Aspects of Security and Trust (FAST)*, September 2003.

[JM96] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353 of *The Kluwer International Series In Engineering And Computer Science*, chapter 5. Kluwer Academic Publishers, 1996.

[KFJ03] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks, 2003*. IEEE, 2003.

[Kle01] J. Klensin. *RFC 2821: Simple Mail Transfer Protocol*. Network Working Group, April 2001.

[Kni21] Frank H. Knight. *Risk, Uncertainty, and Profit*, chapter 1. Hart, Schaffner & Marx; Houghton Mifflin Company, Boston, MA, Library of Economics and Liberty. 8 September 2004 edition, 1921.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the twelfth international conference on World Wide Web*, pages 640–651. ACM Press, 2003.

[Lan03] Marc Langheinrich. When trust does not compute – the role of trust in ubiquitous computing. Workshop on Privacy at Ubicomp 2003, October 2003.

[Lev95] Nancy G. Leveson. *Safeware: System Safety and Computers*, chapter 9. Addison-Wesley, 1995.

[LMW02] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *2002 IEEE Symposium on Security and Privacy*, pages 114–131. IEEE, May 2002.

[Mac87] Mark J. Machina. Choice under uncertainty: Problems solved and unsolved. *The Journal of Economic Perspectives*, 1(1):121–154, Summer 1987.

[Mau96] Ueli M. Maurer. Modelling a public-key infrastructure. In *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, pages 325–350. Springer-Verlag, 1996.

[MC01] D. Harrison McKnight and Norman L. Chervany. Conceptualizing trust: A typology and e-commerce customer relationships model. In *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS-34*, volume 7, page 7022. IEEE, January 2001.

[Mez04] Nicola Mezzetti. A socially inspired reputation model. In *Proceedings of 1st European PKI Workshop (EuroPKI 2004)*, volume 3093 of *LNCS*, pages 191–204. Springer-Verlag, 2004.

[Mos04] Tim Moses. *eXstensible Access Control Markup Language (XACML)*. OASIS Open, v2 edition, DEC 2004. NB: This is a standard produced by the OASIS Open consortium.

[NWD03] Tsuen-Wan "Johnny" Ngan, Dan S. Wallach, and Peter Druschel. Enforcing fair sharing of peer-to-peer resources. In *Proceedings of the Second International Workshop on Peer-to-Peer Systems (IPTPS'03)*, volume 2735 of *LNCS*, pages 149–159, Berkeley, CA, USA, February 2003. Springer.

[OAS03] OASIS Open. *Assertions and Protocol for the OASIS Security Assertion Markup Language*, v1.1 edition, September 2003. NB: This is a standard produced by the OASIS Open consortium.

[Ram31] Frank P. Ramsey. *The Foundations of Mathematics and other Logical Essays*, chapter VII: "Truth and Probability", pages 156–198. Kegan, Paul, Trench, Trubner & Co., London, 1931. (1999 electronic edition).

[Ras94] Eric Rasmusen. *Games and Information: An Introduction to Game Theory*. Blackwell, second edition, 1994.

[RD01] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[Sav72] Leonard J. Savage. *The foundations of statistics*. Dover, New York, second edition, 1972.

[SB02] G. Di Marzo Serugendo and Ciaran Bryce. SECURE: Application scenarios, December 2002. EU Project SECURE IST-2001-32486 Deliverable 5.1.

[SBC+97] Ravi Sandhu, Venkata Bhamidipati, Edward Coyne, Srinivas Ganta, and Charles Youman. The ARBAC97 model for role-based administration of roles: Preliminary description and outline. In *RBAC '97: Proceedings of the second ACM workshop on Role-based Access Control*, pages 41–50. ACM Press, 1997.

[SBD+03] Oliver Storz, Prasad Venkata Boddupalli, Nigel Davies, Adrian Friday, and Maomao Wu. Leveraging the grid to provide a global platform for ubiquitous computing research. Technical Report CSEG/2/03, University of Lancaster, UK, APR 2003.

[Sch02] Bruce Schneier. How to think about security. *Crypto-Gram Newsletter*, April 2002.

[Sch03] Bruce Schneier. We are all security consumers. *IEEE Security and Privacy*, 1(1):104, January 2003.

[SDB03] Brian Shand, Nathan Dimmock, and Jean Bacon. Trust for transparent, ubiquitous collaboration. In *First IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 153–160, Dallas-Ft. Worth, TX, USA, March 2003.

[SDBJ04] Jean-Marc Seigneur, Nathan Dimmock, Ciarán Bryce, and Christian Damsgaard Jensen. Combating Spam with TEA, Trustworthy Email Addresses. In *Proceedings of the Second Annual Conference on Privacy, Security and Trust (PST'04)*, pages 47–58, Fredericton, New Brunswick, Canada, October 2004.

[SFJ+03] Jean-Marc Seigneur, Stephen Farrell, Christian Damsgaard Jensen, Elizabeth Gray, and Chen Yong. End-to-end trust in pervasive computing starts with recognition. Technical Report TCD-CS-2003-05, Trinity College, Dublin, 2003.

[SGF02] Gary Stoneburner, Alice Goguen, and Alexis Feringa. Risk management guide for it systems. Technical Report SP800-30, National Institute for Science and Technology, January 2002.

[Sha04] Brian N. Shand. *Trust for resource control: Self-enforcing automatic rational contracts between computers*. PhD thesis, University of Cambridge, UK, 2004.

[Sim82] Herbert A. Simon. *Models of bounded rationality*, volume 1. MIT Press, 1982.

[SJ05] Jean-Marc Seigneur and Christian Damsgaard Jensen. The claim tool kit for ad-hoc recognition of peer entities. *Journal of Science of Computer Programming*, 54(1):49–71, January 2005.

[SL03] Andrei Serjantov and Stephen Lewis. Puzzles in p2p systems. In *Proceedings of the 8th Cabernet Radicals Workshop*, October 2003.

[SS75] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, September 1975. Online version `http://web.mit.edu/Saltzer/www/publications/protection/` Accessed 2005-03-23 14:45 UTC.

[Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[TD03] Andrew Twigg and Nathan Dimmock. Attack-resistance of computational trust models. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 275–280, June 2003.

[Tem] Brad Templeton. Proper principles for challenge/response anti-spam systems. `http://www.templetons.com/brad/spam/challengeresponse.html` Accessed 7 February 2005 15:36 UTC.

[vNM47] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, second edition, 1947.

[vRD04] Guido van Rossum and Fred L. Drake, Jr. *Python Reference Manual*. Python Software Foundation, 2.3.4 edition, May 2004.

[Wee01] Stephen Weeks. Understanding trust management systems. In *IEEE Symposium on Security and Privacy*, pages 94–105, 2001.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific American*, pages 94–104, September 1991.

[WGZP04] X. H. Wang, T. Gu, D. Q. Zhang, and H. K. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of of Workshop on Context Modeling and Reasoning (CoMoRea'04)*, pages 18–22. IEEE, March 2004.

[Wik04] Pell's equation. Wikipedia: The Free Encyclopedia (`http://en.wikipedia.org/wiki/Pell's_equation`), 2004. Accessed 9 November 2004 16:40 UTC.

[WL02] William H. Winsborough and Ninghui Li. Towards practical automated trust negotiation. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002)*, pages 92–103. IEEE Computer Society Press, June 2002.

[WO04] Jingzhu Wang and Sylvia L. Osborn. A role-based approach to access control for XML databases. In *SACMAT '04: Proceedings of the ninth ACM Symposium on Access Control Models and Technologies*, pages 70–77. ACM Press, 2004.

[WV03] Yao Wang and Jilita Vassileva. Trust and reputation model in peer-to-peer networks. In *Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03)*, pages 150–159. IEEE, September 2003.

[XL02] Li Xiong and Ling Liu. Building trust in decentralized peer-to-peer electronic communities. In *Proceedings of the 5th International Conference on Electronic Commerce Research (ICECR-5)*, October 2002.

[Yao02] Walt Teh-Ming Yao. *Trust Management for Widely Distributed Systems*. PhD thesis, University of Cambridge, 2002.

[Yao03] Walt Teh-Ming Yao. Fidelis: A policy driven trust management framework. In *Proceedings of the First International Conference on Trust Management (iTrust 2003)*, volume 2692 of *LNCS*, pages 301–317, Heraklion, Crete, Greece, May 2003. Springer-Verlag.

[YS02] Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *1st International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 294–301. ACM, July 2002.

[ZZZ+03] Feng Zhou, Li Zhuang, Ben Y. Zhao, Ling Huang, Anthony D. Joseph, and John Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proceedings of ACM/IFIP/USENIX Conference (Middleware 2003)*, volume 2672 of *LNCS*, pages 1–20. Springer-Verlag, August 2003.