Co-operation in the digital age — engendering trust in electronic environments

A Seleznyov, M O Ahmed and S Hailes[†]

The pervasive environment implies a massive increase in the scale of systems, the heterogeneity of devices and diversity in services available, making the complex task of centrally managing the environment unfeasible. The scope and role of trusted third parties in facilitating trust is also reduced because of the high number of bilateral trust relationships, which cannot be predefined and managed statically. Moreover, the dynamic, mobile and asynchronous nature of many of the devices means that it is difficult to predict their state or context of operation from moment to moment. All this adds up to increased uncertainty and a need for a revision of the methods and concepts used to express and assess trust and provide assurance. This work addresses this need by defining realistic models of digital trust that are capable of dealing with the uncertainties inherent in the environment and that are aware of the contexts of interactions in evaluating trust.

1. Introduction

Mark Weiser had a vision of ubiquitous computing that was motivated by his belief that profound technologies disappear into the physical environment that surrounds us [1]. This vision has become a reality. Fewer than 2% of all microprocessors sold go into conventional PCs, the vast majority of the remainder becoming part of embedded systems, although few of these are currently networked. It is not, however, difficult to foresee that the efforts aimed at overcoming the inherent complexities in producing truly ubiquitous networked systems are most likely to bear fruit in the short to medium term.

Emerging ubiquitous infrastructures are characterised by huge numbers of autonomous, heterogeneous entities, interacting across diffuse organisational boundaries. Precisely the same lack of clarity in organisational structures that has led to the failure of public key infrastructure (PKI) outside highly homogeneous and carefully managed domains means that it is impossible to rely on specific or centralised security mechanisms that holistically administer trust in ubiquitous systems.

The increase in opportunity and capability for interaction naturally leads to an increase in the number of bilateral trust relationships between the entities that populate ubiquitous infrastructures, forming PGP-like webs-of-trust on a scale never before seen. Since this model reflects the type of organisational culture that results when economies are free, it will remain difficult to predict and define trust relationships between components with the degree of certainty purportedly on offer in neatly hierarchical PKI systems.

The trust management problem has been extensively studied and discussed, and many definitions have been proposed, but it has not been properly addressed by contemporary solutions [2]. The increased complexity of defining, managing and enforcing the policies of interaction for ubiquitous systems, means that many of the current technologies that are used to protect and provide assurance in traditional networked systems are inapplicable. The way they make their trust decisions is *ad hoc* and application specific, rendering them inflexible, un-scalable and un-portable [3]. Centralised control, relying solely on the *a priori* definition of the role, authority, and relationships between components, is undesirable because of the restrictions that must be imposed on the dynamism of systems to make behaviour easier to predict. Moreover, it is also unsustainable because of the imperfect and incomplete information inherent in such environments that results in uncertainties in the knowledge bases of components. In short, it is necessary to abandon the oversimplistic search for certainty and instead treat uncertainty in trust as a first-class concept, in precisely the same way as we do in everyday life.

Typically, the concept of trust is used in human society to deal with high-risk situations, in which little or no information about each other is available to the parties involved in the interaction [4]. While this concept has been widely addressed in distributed societies of agents [5—7], the appropriateness of the

[†] All authors are from University College, London.

solutions presented to provide flexibility and assurance in electronic environments is still questionable. While the capacity of virtual communities to model and reflect the organisation of groups of real individuals and their interactions, provides motivation to develop digital trust mechanisms, the human notion of trust is still too nebulous and complex for digital environments.

A further problem that rarely merits consideration in the academic literature is the problem of sustainability. There is a grave danger of academic conceit in proposing single solutions to the problem of trust management that are supposedly optimal on some criteria. The reality is that it is impossible to predict the future usage of such systems. Thus, for example, many existing solutions to the trust management problem do not take into account the idea of concept drift, assuming that trust relations and the semantics of the terms used to describe them are fixed for all time. However, local trust decisions are based on shared trust resources and they, in turn, are constantly changing. Having no control over these changes, a trust management system is forced to adapt over a range of time-scales, some of them relatively long term, and failure to do this can only increase error probability and decrease users' trust in the system as a whole. Thus, by hard coding trust decisions into trust management systems, contemporary developers make systems inflexible, denying them the ability to learn from past behaviour in order to adapt to changes and modify their behaviour.

The remainder of the paper is organised as follows — section 2 provides a background to the ubiquitous environment, discussing the premises of this work, and highlighting the characteristics of ubiquitous environments that raise the level of uncertainty. Section 3 discusses the notion of distributed trust management. Section 4 outlines the conceptual architecture of an Autonomic Distributed Authorisation Middleware (ADAM) and its implementation. Finally, section 5 concludes this paper.

2. Security issues in Ubicomp

New flexibility and interconnectivity increase opportunities for interactions unbounded by physical distance or organisational structure. However, realising this level of functionality significantly decreases the level of control one has over the environment, and, in particular, the electronic transactions that take place within it. We therefore face a dilemma — on the one hand, a global electronic environment provides more flexibility and functionality making it easier to co-operate and interact both between people and devices, while on the other hand, this flexibility leads to the uncertainty due to a lack of control over the environment, which is gradually transformed into a lack of trust in the environment, resulting in a reluctance to use it.

There are two key notions implicit in the discussion above — co-operation and trust. Although these are two different concepts, they cannot easily be separated. Without some degree of mutual trust there can be no co-operation, while a person does not need to trust anyone, if they do not intend to co-operate. To encourage co-operation, it is necessary to provide users with the means for making their own trust decisions. In the context of the digital environment, this means that users should have automatic tools that recognise and assess trust-warranting properties of other entities locally, without reliance upon omniscient and omnipresent services. These tools should be lightweight, so as to work in resource-constrained environments (such as *ad hoc* or sensor networks), and transparent enough to make sure that the burden introduced does not exceed the potential benefits. In this work, we propose ADAM (Autonomic Distributed Authorisation Middleware), a middleware system to provide users with such tools [8].

Before we discuss how trust management addresses the problem of situating decision-making and the features of ADAM that support the requirements discussed so far, we discuss how characteristics of the ubiquitous environment create the uncertainties that lead to the opportunities and problems discussed.

2.1 Physical heterogeneity

Within the ubiquitous electronic environment, physical variance is found within the capacity, structure, and space of devices and systems. The capacity of components refers to there being a variety of devices, with different capabilities. The variance in the structure of devices is expressed through the composition of systems. Typically, a loose coupling of traditional centralised networks providing the back-bone infrastructure to support numerous small, lightweight, and often mobile components that provide sensory input and actuation, which may interact in an *ad hoc* manner.

Within many systems, components are embedded in their operational environment and remain fixed in space, while others are mobile. Policies cannot therefore presume that components remain fixed in their location, have the same range of capacities nor adhere to the same structure; instead, different policies may be appropriate for different (classes of) location, capacity and structure.

The reader is referred to the work of Estrin et al [9] for a further discussion of physical heterogeneity.

2.2 *Behavioural heterogeneity*

Behavioural heterogeneity refers to the variety of operational characteristics displayed by components. These include differences in their task, scope, autonomy, policies of interaction and the utilities gained from interaction. Variance within the task and scope of components means that we cannot optimise them for a particular mode of operation.

Within global electronic environments, variance in scope is often expressed through the context of interaction within the operational environment. Devices are beginning to exploit information about their physical location, resources available, and the activity of the user to enhance the user experience. The CoolTown project which extends the Taligent framework [10] is an example of this. Designed for use by mobile users, CoolTown expresses components in terms of how their role affects places, people, and things. However, this is a static method, which does not provide mechanisms for dynamically configuring or changing these properties.

The variance in policy is obvious — components belonging to different principals are governed by the differing policies expressing the concerns of their owners. However, it is not feasible to predefine the sets of all possible interactions, nor to rely fully on trusted third parties to regulate interactions [11], complicating the process of negotiation to achieve faithful interaction.

Variance in utility arises because principals have incomplete information about the beliefs, desires and intentions of prospective trustees. This complicates the assessment of threats posed by prospective trustees. While differing levels of autonomy affects the complexity of the computational models used to create components, the predictability and control of the components means that highly autonomous components must be autonomic in their constitution and behaviour.

Because components are capable of interacting beyond their system boundaries, their behavioural constraints affect the emergence of global ubiquity. To facilitate transient association of components, while maintaining a high-level of assurance, it is necessary to provide for the identification and management of the relevant behavioural constraints possessed by the components.

2.3 *Scale*

The immense number of the devices that must interact to provide the required ubiquity[1] raises two issues:

[1] It is estimated that by the year 2009 there will be more than 3 billion active-network-capable devices using the Internet to operate.

- Invisibility

  To realise the vision of invisibility, devices must necessarily become smaller and more tightly embedded in their environments. While current trends continue to forecast ever-increasing capacity in ever-shrinking devices, we can still safely assume that size will limit capacity [9, 12]. Capacities such as battery, computation power, memory, and sensing capabilities limit the amount of useful work that a device is able to perform locally, thereby severely effecting the scope of security solutions [13]. With respect to this, Estrin et al remark: ‘Fidelity and availability will come from the quantity of partially redundant measurements and their correlation, not the individual component’s quality and precision’ [9].

- Management

  The increase in the number and heterogeneity of components in the environment leads to a dramatic increase in the complexity of managing the environment. Centralised methods do not scale well and hinder the dynamism of systems, making them unsuitable [3]. This problem is particularly acute within the management of the authentication and authorisation process.

2.4 *Embodiment*

The embodiment of components in their environment is among the main requirements to achieve ubiquity. Embodiment means that components are embedded in their operational environment, therefore physical access is limited. Issues such as the update of functionality or management policies cannot be performed easily, further highlighting the need for a degree of autonomic and self-regulating capabilities.

The pressure towards critical mass has many drivers, but problems in usability act as a brake. Dynamism in the ubiquitous environment means flexibility and convenience of usage and this should attract users. However, dynamism and heterogeneity of environment also means complexity and lack of predictability, which creates uncertainty and, consequently, a pressure against embracing the technology for fear of the consequences of doing so. Empowering the user and restoring at least a perception of control, while ensuring that the system can actually function on a second-by-second basis, is the key to acceptability. One of the ways to approach this social challenge is to increase the awareness of potential users. Every person deals with uncertainty in everyday life; each real life situation or event involves uncertainty up to some point and people learn to live with this. In view of the real uncertainties inherent in the ubiquitous computing environment, it would be a mistake to attempt to misrepresent the nature of the system as one in which certainty obtains. Once some degree of uncertainty (and hence the possibility of error) is deemed to be acceptable to end users, it is possible to unconstrain the system, and to make use of incomplete and possibly incorrect information in making decisions about everything from service discovery through to the establishment of trust.

The ubiquitous environment, with its heterogeneity of devices and networks will inevitably lead to a non-negligible set of interactions that are relatively complex and not foreseeable *a priori*. This is analogous to real-life interactions between people, in which many interactions are among those we know well, but a significant number are with those about whom we have little information. In both cases it is not possible to predict all future situations and therefore, uncertainty is inherent in them. Before ubiquitous systems can become a reality, it is absolutely necessary for users to accept and even embrace uncertainty as a part of the decision-making process. Note that this definition includes the situation in which people trust the technology largely because they are unaware of its existence (intentionally so in invisible computing environments) — there is an implicit agnosticism about its effects: 'don't know, don't care, will not ask too carefully'. Nevertheless this all validates the need for autonomic management of systems, to provide solutions that are more flexible, robust and distributed [8].

3. Decentralised trust management

Trust management mechanisms address the above issues by situating the decision-making process in the local context of the interaction. Trust management is aimed at addressing the notion of digital trust holistically, while moving away from looking at the security needs of specific programs and processes.

The term decentralised trust management was coined by Blaze [14] to address the relationships between security policies, security credentials and trust relationships. This work led to the development of PolicyMaker, the basis of which is a compliance checking algorithm that uses a request, a policy, and a set of credentials to try to find a proof that the given credentials and request comply with the policy. A bit-wise response (0 or 1) is returned depending on whether the proof has been found or not. The main problem with this approach is that it is very difficult to devise a fast algorithm for finding proofs, as the underlying process is NP-hard.

Other policy-management-oriented approaches include Centaurus [15] and Vigil [16] developed by Kagal et al. The aim of the above-mentioned systems is to provide service-independent policy management in heterogeneous environments. Centaurus is based around a system of service managers, communications managers, clients and services. Service managers provide service discovery capabilities and act as proxies between clients and services, enabling remote execution of code for resourcing poor clients. Communications managers implement various communications protocols enabling services for heterogeneous devices. At the heart of this architecture are the security agents, responsible for authorising access to services within the group. Security agents carry all policy information regarding the groups they represent and reason about the credentials of prospective trustees. Centaurus ties all clients and services to service managers that handle all security information for them. Vigil extends the Centaurus architecture by introducing certificate controllers and role assignment managers.

The trust mechanism for Centaurus systems is based around a Prolog compliance checker that attempts to find a proof that a given assertion (a right for a prospective trustee) is valid. The focus of this framework is to realise a trust management system that reasons about policy by allowing and regulating capabilities such as delegation. However, it is unsuitable for the target environment of this work for a number of reasons:

- the use of goal-oriented planners means that the assessment of rights is computationally demanding and complex,
- the number of components means that it is large, and the reliance upon trusted security agents means that the attack-resistance of the trust mechanism is reduced to a single point of failure, i.e. the compromise of the security agent compromises the whole group,
- the system is moderately static and does not allow for the negotiation, or the dynamic change, of rights and roles.

The Simple Universal Logic-oriented Trust Analysis Notation (SULTAN), developed by Grandison [2], is intended to be a simple, comprehensive, framework to analyse and manage trust relationships, and is designed to underpin trust management. SULTAN uses a goal-oriented planner to refine rules, which often requires a network of policies (both authorisation and obligatory). This computational complexity renders the approach unscalable in complex situations, because the refinement process can become extremely difficult, if not intractable. Although a generic specification is proposed for SULTAN, there is no efficient algorithm to automatically perform trust management operations, and additionally, although the notions were defined, there is no direct (or indirect) built-in mechanism supporting trust recommendation (referring) rules. SULTAN, however, presents a more holistic approach to trust management by introducing risk management as part of the trust assessment process.

Distributed trust management systems face a number of obstacles.

- Perception of the environment

  Sensor-derived information is error prone and imperfect. Translating it to knowledge and truth is plagued with the problems of identifying the context from which information is obtained, important signalling features of the environment, and false-positive reactions.

- Flow control

  As the boundaries between systems become blurred, individual components cannot easily be segregated. Capacities such as mobile and distributed code further complicate this problem; therefore, information flow models, rather than traditional control flow models, are more applicable.

- System addressing

  We are no longer able to address systems through their component units but must address them in terms of sub-systems that may be unbounded in large open environments.

- Policy constraints

  If these components span organisational as well as physical boundaries (as is likely), a range of policy constraints will govern their behaviour, introducing further uncertainties and complicating the design of the control mechanism. Therefore, a compromise must be reached between the autonomy and manageability of suitable architectures.

4. ADAM

We have developed the ADAM (Autonomic Distributed Authorisation Middleware) architecture, aimed at the automation of the trust establishment and maintenance process. ADAM is based on forms of distributed knowledge acquisition and management to deal with the uncertainties introduced by the requirements of information access in ubiquitous environments. It uses self-organisation techniques to make the information that principals receive more relevant to their dispositions and to segregate malicious principals. Moreover, the explicit declaration of context means that decision taking can be situated close to interactions.

To date, many definitions of trust have been proposed, and we will not present a new definition in this work, but instead adapt an existing definition and concentrate on the management of trust relationships. The focus of this work is on building an efficient system that controls the full life cycle of a trusting relationship — starting from its establishment through to its revocation. The proposed system incorporates reactive elements that monitor the manipulation of network resources and respond when malicious activity is detected. However, these elements are beyond the scope of this paper.

For our work, we adapt the definition of trust in a way that treats it as '... a measure of willingness of a responder to satisfy an inquiry of a requestor for an action that may place all involved parties at risk of harm, and is based on an assessment of the risks and reputations associated with the parties involved in a given transaction'. As this definition states, all parties involved in interaction may be harmed by its consequences. By providing access to its resources, a principal may be explicitly harmed by the malicious actions of trustees. Third parties involved with the principal in existing trust relationships may implicitly be compromised by the relationship. Trustees are at risk, since obtaining incorrect or contradictory information compromises the integrity of their knowledge base and those of their trusting parties. These situations may arise as the result of intentional malicious activity or through legitimate error such as software bugs, network failures, and user errors. Within ADAM, the potential risk of undesirable outcomes are assessed by checking the credentials of participants and the history of their behaviour with respect to the value of the resource that potential users request; only then is a decision made about whether the risks likely to be posed through interaction are acceptable.

In establishing trust relationships, the following attributes are important — the participants, the scope (spatial and temporal restrictions applied to a current trust relationship) of the relation, the risk (as a degree of potential damage) associated with the relation, and security to describe the characteristics of trust. Each access control relation expresses sets of principals that use and provide resources in the environment, actions that may be applied to the resources, and policies governing the use of the resources.

ADAM is a multi-agent system that relies on autonomous agents to carry out users' requests on one side, while protecting resources/services on the other. Agents are policy-aware entities that use distributed knowledge management to collect evidence on the trustworthiness of prospective users and make decisions regarding whether to co-operate. Trusting decisions within ADAM are not binary (only to co-operate or to defect); for example, users may be offered more limited access than requested if the amount of or strength of the collected evidence about their trustworthiness is insufficient for the requested action (according to the local policy of the resource or service being requested). If the importance of the transaction is high, users may continue negotiating, providing additional information about themselves until either the resource manager is satisfied or the user deems the cost to be too high. All decisions regarding interaction are made by the ADAM's trust engine and are based on collected evidence about the trustworthiness of prospective participants and the local policy of a resource or service.

Within ADAM, interaction starts with a request for a service; requests are tuples, consisting of a resource, the set of actions to be performed and the identity of the user. In order to facilitate this, there must be a service/resource discovery mechanism that allows prospective trustees to look for services, view the actions available and the credential requirements of the services, and choose between them. ADAM does not itself perform authentication, it only authorises. When a user submits their request, an authorisation-agent is launched by the resource to collect evidence about the behavioural disposition of the identity presented. There are three main sources of information from which entities' trustworthiness can be derived [17]. The first source is from direct observations that are formed by recording outcomes of previous interactions with an entity. The second is the recommendations of trusted entities that allow the propagation of trust. The third source is in the reputation of an entity. Reputations are knowledge about an entity's behaviour derived from their history of interactions. They may be derived from the common knowledge of an organisation or a community, or based on specific predefined knowledge [18]. For our system, we use knowledge that is accumulated over time in communities of practice to assess reputations of network entities.

The process of interaction starts with the authorisation procedure, during which resource agents perform risk assessment and check whether the potential risk posed by the interaction is acceptable in terms of the local policy of the resource. The potential damage to the user's reputation and the potential for identity loss or loss of associated values are assessed, i.e. money may be charged if a credit card number is provided.

Resource agents analyse the evidence presented with respect to the potential damage the resource may incur if the requested actions are granted. It may therefore be the case that a user will be declined access when using one of their electronic identities and granted access when using another.

4.1 System architecture

In the previous section, we described the principles on which our system is founded. Next, we discuss ADAM's conceptual architecture in more detail, giving practical considerations about its implementation.

Authorisation decisions in ADAM are produced as the result of negotiations between two agents — user agents and authorisation agents. The former are implemented as mobile agents. They are aware of local policy on the user side and represent user interests in the negotiations. The user agent contains information about its legal user, their secret keys, and the certificates required for the user authentication (it may include some other information, such as credit card numbers, user names and passwords for different resources, etc). All information is encrypted and, to be activated, a user agent requires a correct PIN or password to be entered. A PIN constitutes part of a decryption key for user agents, and can only be activated by authorised people. It also denies access to user agents when they are inactive or travelling. The mobility of agents allows them to move across networks or between devices. For, example, for the user's convenience, an agent may be resident in the user's PDA.

The user agents not only simplify users' lives, they make network management easier since they automate certain tasks, such as password and certificate management. For example, to reissue a user password, the user agent is notified. After this, it moves to the network server responsible for managing users' profiles, where the password is changed in a secure environment, making it unnecessary to transmit sensitive information over the network. This method also has other benefits. Since the user does not need to remember their password, it is possible to choose strong passwords automatically without requiring any extra activity from users.

Some information is stored in user agents for users' comfort. It is necessary to remember only one PIN or password. Once the user agent is activated it can submit some user information on request, e.g. user names and passwords for other resources, certificates. However, clearly, some information, such as private keys, should never leave user agents. There is still a small probability that a malicious person manages to obtain information from an agent. We consider this to be rather smaller than the risk of finding out some or all of the PINs used to activate an agent. In Veijelainen et al [19], it was argued that, when correctly implemented, this kind of user information storage does not bring new security risks to those already present in computer networks.

Authorisation agents are meant to protect network resources by ensuring that only valid users obtain access to them. Agents are aware of local policy on the resource side and enforce policy rules and procedures. Also, after the authorisation procedure, agents enforce access control restrictions and monitor usage of resources in support of reactive security.

Consider the negotiation process in detail. Figure 1 shows the main phases through which ADAM must go in order to process each request. Initially, there are two interested parties that are potentially willing to co-operate — client and service. The former is looking for a service or resource to use for their needs. The latter is willing to provide this service. Firstly, the client needs to locate an appropriate server. Secondly, in order to co-operate, they must convince each other that they are sufficiently trustworthy to perform this transaction. These actions take place in a number of steps as numbered in Fig 1 and described below.

Each user has to activate their user agent by giving the correct PIN/password (1), as discussed above. A secure channel is established between the user terminal and the agent. If no agent with this user's information is found, a new agent is created and assigned the task of carrying out user requests. After this, in order to find an appropriate service, the client needs to perform service discovery. During this procedure, information about different services advertised in the network is gathered (2). This information includes types and descriptions of services and information that the client has to provide in order to be able to use them. Depending on local policies, different resources may have different requirements. Thus, the user agent must select the most suitable service that requires information the user is happy to provide.

The client may need to evaluate the chosen service (3) before using it. The evaluation is performed with the help of local policy (4). Additionally, the client may wish to check the quality of the service by collecting opinions of other clients (5). When this is done, the client may wish to continue, and makes a request (6, 7).

When the request is received on the service side, an authorisation agent is created to handle it. This performs risk assessment and checks whether the potential risk is acceptable in terms of the local policy of the resource (9). In doing this, it assesses the potential damage to the user in terms of the harm it is possible to do to the reputation associated with this identity, including identity loss, or loss of associated values (money may be charged if a credit card number is provided). It then compares this to the potential damage that could be sustained by the resource. It is therefore possible that a user will be declined access when using one of their electronic identities and granted access when using another.

Depending on the circumstances, the authorisation agent may request additional information to be provided by the user (7). However, this may contradict

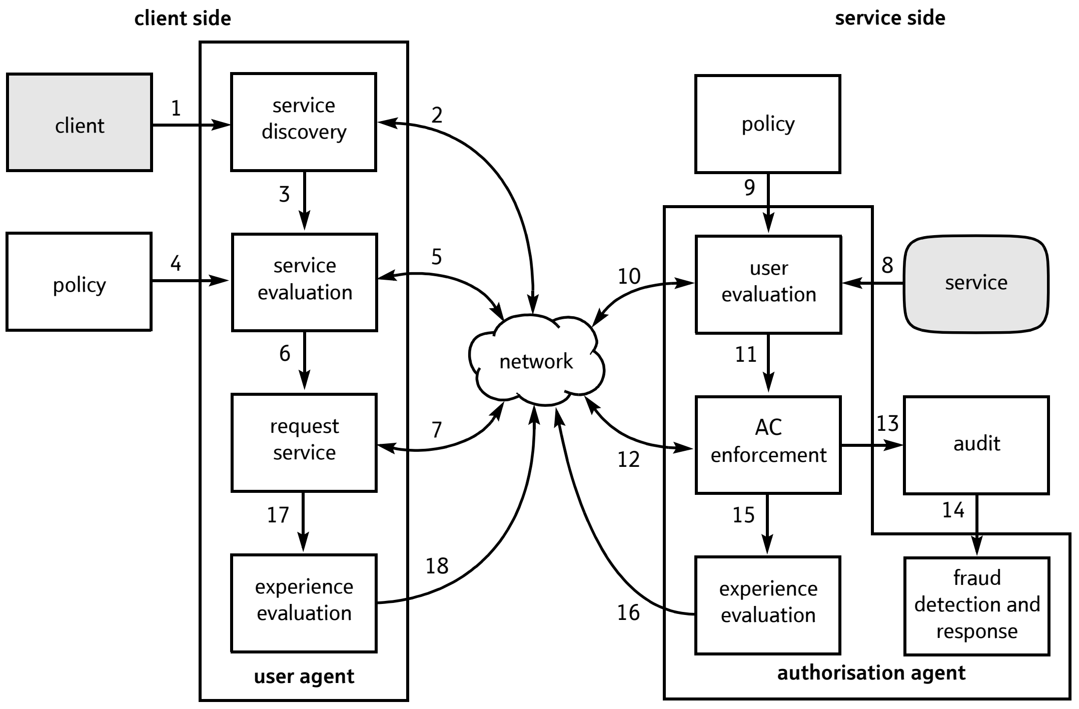

Fig 1 Authorisation process in ADAM.

user policy. For example, the user policy may not allow the release of some information from its local network.

When the client provides the information required by the service, the authorisation agent must collect evidence that the information provided is correct and that the user who requests the service is indeed the declared person (8).

At this point, the authorisation agent that processes the request has obtained the information on which it will base its authorisation decision. However, it does not yet know whether this information is trustworthy nor whether it has been sent by a legitimate source. The agent does not perform authentication itself. Instead, the authorisation agent delegates this task to third parties that have had previous experience of interactions with this user or different authorities (10). This runs an automatic credentials discovery (ACD) protocol, which forms the subject of a separate paper. However, its basis is that there are different sources available that can be used to verify the user identity, provide information about a user's reputation, and verify information provided by a user. In pervasive systems, these sources could be distributed over numerous networks and, as a result, might not be capable of working co-operatively. Consequently, the authentication agent must collect their recommendations and combine them to have a reasonable basis for the access control decision it will take. For example, the local profile management server may verify a password's hash sent along with the request; or a request signed by a user's private key may be verified if one of the parties provides the corresponding public key; or there are authorities who can verify credit card numbers. We would like to note that local policies and the availability of information dictate the number of steps in this process and proof of identity required to obtain access.

After a request for user credentials is made, the authorisation agent must collect pieces of knowledge about the user in a secure and private manner. This information must be transformed from heterogeneous opinions into homogeneous data that can be automatically combined and thus allow a decision about user reputation to be made (11). It is worth noting that the result of negotiations between agents is not binary. The negotiations themselves are regulated by a set of fuzzy rules that are dynamically created and reflect local policies. Thus, the authorisation agent may decide that a user's credentials are inadequate to authorise the requested action, e.g. 'read/write', but are adequate to allow another, say 'read'. The client may accept or decline the offer, or be willing to give some extra information to obtain the desired service (for example, some companies require a deposit or a card number if a client does not have a credit history). After a user's credentials have been collected and evaluated, the authorisation agent decides whether or not to perform the action. If 'yes', the agent creates an association that is given to the client (12). The agent enforces access restrictions by controlling this association. Over the lifespan of an association, authorisation agents perform continuous auditing as a basis for the later (re)assessment of the user's reputation. Audit trails are also used for reactive fraud detection and response (14). Agents perform both misuse and anomaly detection and notify interested parties about any problems. When the action has been completed, the authorisation agent classifies its experience as positive or negative (15) and disseminates updates to user credentials (16). After this, the agent is destroyed, invalidating the client's association. At the client, the user agent evaluates user experience (17) and disseminates service credentials when appropriate (18).

While we have no space here to explore this further, it is unreasonable to assume that recommenders always provide accurate testimonies; the system can be subverted both maliciously and as a result of the use of different knowledge management methods or policies. Thus evaluations of testimonies (fraud detection) and agents' ratings are used to maintain overall system integrity by favouring better recommenders.

4.2 Implementation

It is a requirement of any implementation of this architecture that it be sufficiently flexible to support heterogeneity of resources and transactions in the environments in which it is going to operate. Consequently, ADAM is being implemented as distributed lightweight component-based middleware. Figure 2 outlines its structure.

The system consists of independent components that can easily be re-implemented and replaced. Multiple instances of some components allow the system's behaviour to reflect underlying physical heterogeneity (in Fig 2 there are five components marked that are most likely to have multiple instances). Although the detailed interfaces for each component are outside the scope of this paper, the components themselves can be described at a high level.

- Component manager

  The component manager is the system core and enables the remaining component instantiations to communicate. Since it provides the essential integration between all other components, it must necessarily be lightweight and small. It is primarily intended as a mechanism to allow components to communicate, but it is also responsible for the

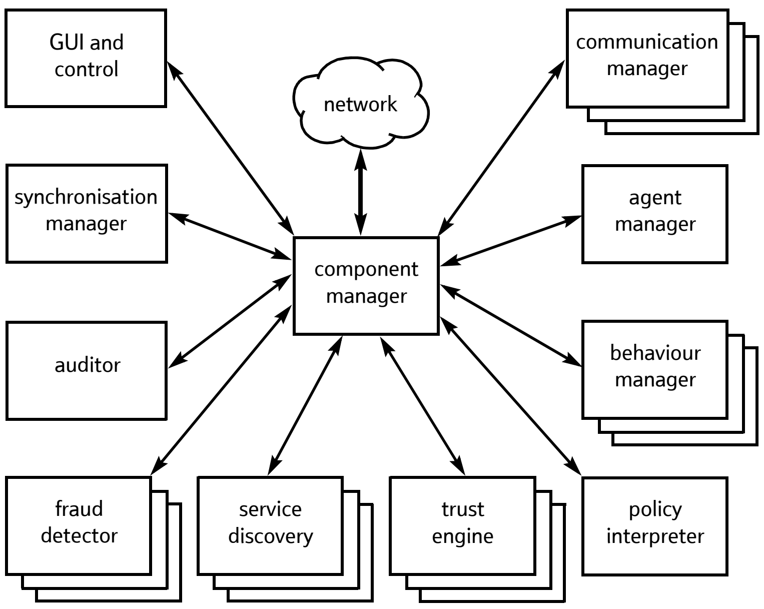

Fig 2 ADAM component organisation.

management of those components — for loading and unloading components. At present, for ease of design, XML is used for internal message description. It is expected that components will sometimes be unavailable due to failures, breaks for updating, connection loses, etc. Thus, we explicitly allow for the possibility of asynchronous communications between components and we further allow components to be distributed for cases in which a single node does not have the required power to support the entire architecture, but where a small collection of neighbouring nodes might.

- Communication manager

  The communication manager is a component responsible for support of method invocation and messaging with outside parties, including responsibility for interaction with name services and for message routing, particularly in *ad hoc* networking environments.

- Synchronisation manager

  The synchronisation manager is responsible for maintaining locally reliable time and date information to ensure freshness of messages and for audit purposes. It is not necessary to have the absolute time; indeed, Lamport's logical clocks might provide sufficient information to allow for the ordering of messages within sessions.

- Audit

  The audit component is used for collection and storage of information about transactions for debugging, non-repudiation, and for fraud detection purposes. This can log information using different levels of verbosity, as determined by the fraud detector or GUI and control components. The audit component allocates and maintains the space required to store the audit log file, including the garbage collection of data as it is no longer needed.

- GUI and control

  The GUI and control component is needed to control the system, to permit the explicit change of components and the system configuration, and for debugging purposes. The GUI element allows for visualisation of the operation of different components.

- Service discovery

  The service discovery component is responsible for discovery and evaluation of networked services. Users provide criteria for service evaluations and searches. Multiple instances of this component may implement different service discovery mechanisms.

- Agent manager

  The agent manager maintains the agents created to perform different tasks for ADAM. It is responsible for space allocation, creation, deletion, and migrations of agents. It maintains current system state by continuously polling agents to check whether they are alive. It is responsible for maintenance of links to external agents in case existing agents (not controlled by the system) need to be connected to the system.

- Policy interpreter

  The policy interpreter takes organisational/ environmental and user-specified policy and translates it into a universal underlying format. During this procedure it must resolve conflicts between policies and filter unnecessary policy rules. The result of conversion is used by the behaviour manager. The policy interpreter may interact with GUI and control components to facilitate user-defined policy specification.

- Trust engine

  The trust engine provides the core mechanism for trust-based decision making, along the lines described in detail above.

- Behaviour manager

  The behaviour manager uses encoded policy provided by the policy interpreter and abstract code from the trust engine to refine agents' behaviour. There might be more then one component providing abstract code. This code specifies behaviour of created agents. Thus, by registering more than one instance of the component it is possible to implement a variety of behaviours for agents (e.g. data collection agents, intrusion detection agents).

- Fraud detector

  The fraud detector provides misuse (and possibly anomaly) detection mechanism to detect abuse of given privileges. Its multiple instances may implement different fraud detection techniques.

5. Conclusions

This paper presents a conceptual description of the distributed access control system (ADAM), which is aimed at automation of a trust establishment process by performing distributed knowledge acquisition and management. The architecture is based upon two groups of agents — mobile user agents protecting user interests and authorisation agents protecting network resources. The access control decisions are results of negotiations between them. Local policies are translated into sets of fuzzy rules and the negotiations aimed at finding consensus between these sets.

The system allows automated trust establishment by gathering information about network entities and, later, maintenance of trust by constantly controlling information flow and manipulations with network resources. Several foundational aspects of ADAM make it different from the other trust management systems:

- it is designed to work in a range of networks, allowing automatic trust establishment and maintenance between entities situated in different network and administrative domains — this provides additional flexibility and allows ADAM to function in an ambient computing environment,
- it allows each user to have multiple electronic identities — it only authorises, it does not authenticate, the authentication task being delegated to separate parties,
- ADAM authorises transactions (actions), based on the history of the identity presented, with respect to the risk these actions pose to the resource if sanctioned, rather than the users.

Overall, the ADAM system facilitates automatic trust establishment and maintenance independently of the type and topology of underlying networks. This provides considerable flexibility and allows ADAM to function in pervasive environments.

References

1 Weiser M: 'The computer for the twenty-first century', Scientific American, 256, No 3, pp 94—104 (1991).

2 Grandison T: 'Trust specification and analysis for Internet applications', PhD Transfer Report, Imperial College of Science, Technology and Medicine, Department of Computing (2001).

3 Grandison T: 'Trust management for Internet applications', PhD theses, Imperial College of Science, Technology and Medicine, Department of Computing (2003).

4 McKnight D H and Chervany N L: 'The meanings of trust', University of Minnesota (1996).

5 Abdul-Rahman A and Hailes S: 'Supporting trust in virtual communities', 33rd Hawaii International Conference on System Sciences (2000).

6 Chopra K and Wallace W: 'Trust in electronic environments', 36th Hawaii International Conference on System Sciences, pp 331—340 (2003).

7 Marsh S: 'Formalising trust as a computational concept', PhD Department of Computer Science and Mathematics, University of Sterling (1994).

8 Seleznyov A and Hailes S: 'A conceptual access control model based on distributed knowledge management', IEEE, 18th International Conference on Advanced Information Networking and Applications (2004).

9 Estrin D, Culler D, Pister K and Sukhatme G: 'Connecting the physical world with pervasive networks', pp 59—69 (2002).

10 Kindberg T, Barton J, Morgan J, Becker G, Caswell D, Debaty P, Gopal G, Frid M, Krishnan V, Morris M, Schettino J and Serra B: 'People, places, things: Web Presence for the Real World', in Proc WMCSA2000 (2001).

11 Eustice K, Markstrum S, Ramakrishna V, Reiher P, Kleinrock L and Popek G: 'Enabling secure ubiquitous interactions', 1st International Workshop on Middleware for Pervasive *ad hoc* Computing (2003).

12 Stajano F: 'Security For Whom? The Shifting Security Assumptions of Pervasive Computing', Springer-Verlag, Proceedings of International Security Symposium (2002).

13 Stajano F and Crowcroft J: 'The butt of the iceberg: hidden security problems of ubiquitous systems', in Basten T, Geilen M and deGroot H (Eds): 'Ambient Intelligence: Impact on Embedded System Design', Kluwer Publishers (2003).

14 Blaze M, Feigenbaum J and Lacy J: ‘Decentralized trust management’, IEEE Symposium on Security and Privacy (1996).

15 Kagal L, Korolev V, Chen H, Joshi A and Finin T: ‘Centaurus: a framework for intelligent services in a mobile environment’, The 21st International Conference on Distributed Computing Systems Workshops (ICDCSW ’01) (2001).

16 Kagal L, Undercoffer J, Perich F, Joshi A and Finin T: ‘A security architecture based on trust management for pervasive computing systems’, Grace Hopper Celebration of Women in Computing (2002).

17 English C, Wagealla W, Nixon P, Terzis S, McGettrick A and Lowe H: ‘Trusting collaboration in global computing’, The First International Conference on Trust Management, Springer-Verlag, Vol 2692, pp 136—149 (2003).

18 Mui L, Mohtashemi M and Halberstadt A: ‘A computational model of trust and reputation’, 35th Hawaii International Conference on System Sciences (2004).

19 Veijalainen J, Seleznyov A and Mazhelis O: ‘Security and privacy of the PTP’, in Makki K, Pissinou N, Makki K and Park E (Eds): ‘Mobile and wireless Internet: protocols, algorithms, and systems’, Kluwer Publishers, pp 165—190 (2003).

Alexandr Seleznyov is a research fellow in the Computer Science Department, University College, London.

He received his 1st class MSc degree in Computer Engineering in 1997 from Kharkov State Technical University, Ukraine, and PhD from the Department of Computer Science and Information Systems, University of Jyväskylä, Finland in 2002.

His main research interests include network security, distributed artificial intelligence, multi-agent systems, mobile networks, sensor networks and electronic transactions.

Mohamed Ahmed holds a BSc in Computer Science and AI and an MSc in Robotics and AI, both from the Department of Computer Science, at the University of Essex, UK.

He is currently pursuing a PhD at the Department of Computer Science, UCL. He works within the area of security in ubiquitous networking environments.

His research focuses on adapting traditional models of trust and assurance to be more dynamic and adaptive in prescribing rights to users and proactive in response to anomalies.

Stephen Hailes was an undergraduate at Trinity College Cambridge and then a PhD student in the Cambridge University Computer Lab. Following this, he joined the Department of Computer Science at University College, London, as a Research Fellow.

Shortly afterwards he was made a lecturer, and then, in 2000, a senior lecturer. He has been Director of Studies since January 2003.

He is interested in all aspects of mobile systems and security. But most par-ticularly in *ad hoc* systems, pervasive/ambient computing environments, and how to secure these areas. He is currently the PI of the MARS project, which is a collab-orative project in conjunction with BT looking at building middleware components for trust management in pervasive environments, and the CoI on SENIT, an EC Framework 6 Integrated Project looking at securing ambient networks. His past projects include 6WINIT, an EC Framework 5 project, EPSRC-funded projects (PIMMS, MOSQUITO, and MEDAL), and UKERNA-funded projects such as ACOL and TITAN.