Application Security Models for Mobile Agent Systems

J. Todd McDonald [1]

Department of Computer Science
Florida State University
Tallahassee, FL, USA

Alec Yasinsac [2]

Department of Computer Science
Florida State University
Tallahassee, FL, USA

Abstract

Mobile agents are a distributed computing paradigm based on mobile autonomous programs. Mobile applications must balance security requirements with available security mechanisms in order to meet application level security goals. We introduce a trust framework to reason about application security requirements, trust expression, and agent protection mechanisms. We develop application security models that capture initial trust relationships and consider their use for mobile agent security.

Key words: Mobile agents, trust, security, application requirements, software protection, models, frameworks

1 Introduction

Mobile agents are autonomous programs with the capability of changing their execution location through a series of migrations and corresponding state updates. Mobile agent applications specify and enforce security requirements based on the unique interactions of agent servers (hosts) with static and dynamic agent components.

[1] Email:mcdonald@cs.fsu.edu. The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government

[2] Email: yasinsac@cs.fsu.edu. This material is based upon work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number DAAD19-02-1-0235

Traditionally, mobile agent security has focused on two forms of protection: keeping malicious parties from altering the agent and keeping malicious agents from harming other parties including potential hosts. Several surveys [1,2,3] categorize and describe attacks against agent systems along with mechanisms for defense.

Trust formulation has been given considerable thought both in distributed networking applications [4,5,6,7] and mobile agents [8,9,10,11,12]. Mobility as an application feature complicates trust because the receiving execution host must make distributed trust decisions in the face of little or no prior knowledge. Likewise, user agents must evaluate trust with hosts in different security contexts.

To date, other trust models for mobile agents have not addressed how to link requirements with appropriate agent protection mechanisms. Other trust models lack integration of generic security mechanisms or reasoning about initial trust relationships (what we term an *application security model*). We bridge this gap by posing a trust-based security framework for mobile agents with three novel features:

- Ability to link application security requirements with mechanisms based on trust
- Reasoning about trust properties for generic security mechanisms
- Application models for initial trust among principals in a mobile agent setting

The rest of this paper describes our trust framework and is organized as follows: section 2 discusses related works concerning trust and security requirements in the mobile agent paradigm. Section 3 presents our framework for expressing trust and security in mobile agent systems. Section 4 expounds three different application-level security models and section 5 summarizes our contributions.

2 Related Works

Our security framework novelly incorporates three notions: security requirements, agent security mechanisms, and trust. *Security requirements* are defined as the desire to guarantee one or more canonical qualities: privacy, integrity, availability, authentication, and non-repudiation. *Security mechanisms* for mobile agents enforce requirements and are categorized as either detecting or preventing malicious behavior.

Defining *trust* is as precarious as defining *agent*–and though researchers do not agree on either term they do discern the importance of both concepts in framing research. We define trust loosely in a previous work [7] as the expectation of behavior among parties and classify several different trust infrastructures in dynamic networks. Gambetta [13] defines trust as a subjective probability that is non-reflexive, changing, and context driven. Trust can be transitive and therefore delegated [10] or can be acquired by direct observation [14,15].

Trust management [5] is a framework for defining policies, actions, relationships, and credentials in terms of trust. Traditional trust management systems delegate permissions using certificates (credentials) that reduce to static decisions that do not scale well or allow change over time. Capra [15] points out limitations of several trust management frameworks: they are designed for centralized servers, have too high a computational overhead (*for mobile devices*), lack dynamic trust evolution, lack details about local policy usage, and lack subjective reasoning capabilities.

Early work in mobile agent security centered on policy management and malicious code protection based on credentials [16]. Distributed policy tools for mobile agents are developed in [10,17]. Trust cannot be hard-coded in applications that require decentralized control in large scale heterogeneous networks [6].

Mobile agents particularly need to separate application purpose from trust management issues if they are to scale well. Research efforts to implement trust expression in mobile applications include Lin et al. [11], SECURE [14], hTrust [15], and MARISM-A [9]. Our trust architecture incorporates several properties found in current work: derived and acquired trust (opinions from third parties), delegated trust, linking trust with security decisions, temporal considerations, and non-Boolean trust.

We fill in a key missing link in current models: the relationship between security requirements, trust, and agent protection mechanisms. Tan and Moreau attempted one of the first models of trust and belief specifically for mobile agent-based systems in [8]. This distributed authentication mechanism model is limited because it only uses trust expression for their extended execution tracing security mechanism. Conversely, our model gives the ability to account for a wide range of mechanisms and requirements.

3 Trust Framework

To describe our framework, we first define the principals that can be assigned trust properties, define next the nature of trust relationships between principals, and finally formulate what trust relationships can accomplish in mobile applications settings.

3.1 Principals in Mobile Applications

Three distinct groups of principals are found in mobile agent systems: *agents*, *hosts*, and *entities* (described in figure 1 in extended BNF[3]). We define an *agent* as a composition of static software (*code*) and a set of dynamic states (*state*) that represent the migratory results of the agent. Agents are described

[3] ISO/IEC 14977:1996(E)

by their migration path (*itinerary*), any unique identifiers (*id*), a log of agent or host activity (*log*), and a security specification (*policy*) that includes any historical trust information for other principals in the agent application.

Hosts provide an execution environment for the agent. They encompass the underlying physical hardware, runtime services, and middleware necessary for agent migration and execution. Agents see a host as a collection of computational, communicational, informational, and management resources. Hosts also have security policies that support the trust formation and decision process.

Three classes of hosts are relevant to mobile computations: the *dispatching host* (DH) associated with the application owner that launches mobile agents, the *executing host*(EH) where mobile computations occur, and *trusted hosts*(TH) which have ability to change trust relationships among other principals based on services they offer.

Three entities have bearing on security relationships in mobile settings. The creator of the static agent code is the *code developer* (CD) while the code user is the *application owner* (AO). The (CD) and (AO) may be the same. The owner of a computer, the systems manager of a computer, and the user of a computer can be the same person, or can be separate individuals with different levels of trust.

For simplicity, we view the host owner, manager, and user as synonymous and apply the term *host manager* (HM) to refer to all three responsible parties. In human terms, we trust machines (hosts) and software (agents) in some cases because we trust the *manager* of the environment or the *developer* of the software. We equate the trust that we have in the host manager as the trust we have in any other host, realizing that the host manager for the dispatching host, the code developer, and the application owner can all be different entities.

```
      <agent> =   <code>, <state>+, <itinerary>, <id>, <log>, <policy>
       <host> =   <resource>+, <id>, <log, <policy>
<entity-type> =   <code developer> | <application owner> | <host manager>
     <entity> =   <organization>, <entity-type>
  <principal> =   <agent> |  <host> | <entity>
      <trust> =   </evel>,<foreknowledge>,<timeliness>
<application> =   <principal>+ , (<principal>,<principal>,<security requirements>+, <trust>)*
```

Fig. 1. Principals in Mobile Agent Systems

We define an application as the collection of all possible hosts that will be involved in the agent task and the set of uniquely identifiable agents that implement a user function. This intuition captures single agents and multiple collaborating agents including those with the same static code and different itineraries and those with different static code. We now define our notion of trust relationships that are shared among principals in our model.

3.2 Trust Relationships

One security task is to rightfully attribute observed actions within the system to a given party. The data state and code of a mobile agent is influenced by the code developer, the dispatching host, and all visited hosts–making attribution difficult. For simplicity, we equate the trust in the agent code with trust in the code developer and we will equate trust we have in the dispatching host as the trust we have in the application owner. In the spirit of [14,15], we define a trust relationship $\delta : P \rightarrow P \rightarrow S \rightarrow (L, F, M)$ as a mapping δ between two principals (P_x, P_y) and some number of security requirements (S) with 3 associated parameters: trust level (L), foreknowledge (F), and timeliness (M), depicted in figure 2.

We categorize trust levels (L) in a range from highly untrusted (HU/U) to highly trusted (HT/T), where in some instances trust is not determined (ND). Trust levels are non-Boolean and reflect a one-way subjective level of belief that one party will behave towards another party at some perceived level of malicious intent (HU, U, ND, T, HT). Trust can be discretely categorized negatively and positively as ranges between $[-1, 1] : (HT, T) > 0; ND = 0; (U, HU) < 0$ or as levels in the range $[0, 1]$.

Foreknowledge (F) is a statement of prior interaction between principals. Agents traveling in a dynamic free-roaming itinerary can encounter hosts that are unknown. Likewise, hosts are likely to encounter agents with no prior experience. Foreknowledge of a principle is described as either well-known (WK), known (K), or unknown (UK). *Well-known* principals have established histories while *known* principals are identified for possible interaction.

Timeliness (M) is categorized as *expired*, *stale*, or *fresh*. Timeliness may be established by mechanisms such as timestamps [15] and freshness can be used to make determinations on whether recommended or delegated trust decisions are reliable. Given the same volume of interaction, trust is higher when the interaction period is longer. When the elapsed time since the last interaction is short, higher confidence can be placed in more recent interactions.

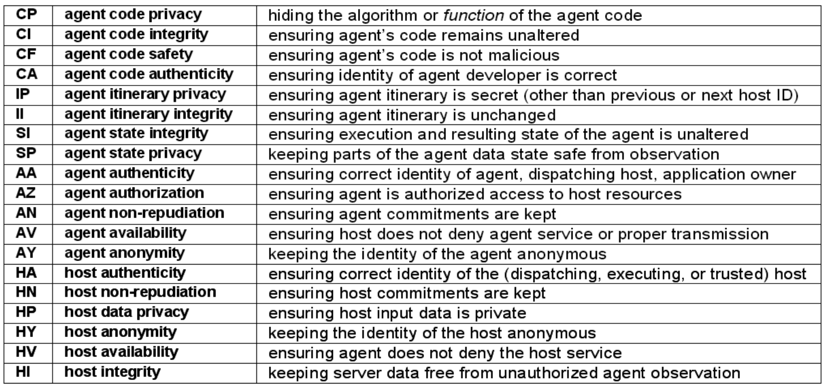
<δ> = <P_x>, <P_y>, <S>+ , <L>, <F>, <M>
<L> = <highly untrusted> | <untrusted> | <non-determined> | <trusted> | <highly trusted>
<F> = <well known> | <known> | <unknown>
<M> = <expired> | <stale> | <fresh>
<S> = <CP> | <CI> | < CF> | <CA> | <IP> | <II> | <SI> | <SP> | <AA> | <AZ> | <AN> | <AV> | <AY> | <HA> | <HN> | <HP> | <HY> | <HV> | <HI>

Fig. 2. Trust Relationships

3.3 Defining Security Requirements

Figure 3 defines mobile agent requirements derived from the traditional CIA model for describing security (confidentiality/integrity/availability). The list provides a comprehensive set of security requirements based on taxonomies found in [1,2,3]. Requirements dictate the security environment necessary for agents to operate at the executing host and the level of protection which hosts require from executing agents. To achieve a security requirement, a principal must either have a degree of trust towards another principal in the system in regards to a given security requirement or else a security mechanism must be in place to enforce that *particular* requirement.

Trust level, foreknowledge, and timeliness bind trust from two principals (an application owner, an executing host, a dispatching host, an agent, etc.) with one or more security requirements (elaborated in figure 3). Though we represent foreknowledge, trust level, and timeliness discretely, they can be converted to continuous ranges ($[-1, 1]$ or $[0, 1]$ for example) to accommodate different trust algorithms.



CP	**agent code privacy**	hiding the algorithm or *function* of the agent code
CI	**agent code integrity**	ensuring agent's code remains unaltered
CF	**agent code safety**	ensuring agent's code is not malicious
CA	**agent code authenticity**	ensuring identity of agent developer is correct
IP	**agent itinerary privacy**	ensuring agent itinerary is secret (other than previous or next host ID)
II	**agent itinerary integrity**	ensuring agent itinerary is unchanged
SI	**agent state integrity**	ensuring execution and resulting state of the agent is unaltered
SP	**agent state privacy**	keeping parts of the agent data state safe from observation
AA	**agent authenticity**	ensuring correct identity of agent, dispatching host, application owner
AZ	**agent authorization**	ensuring agent is authorized access to host resources
AN	**agent non-repudiation**	ensuring agent commitments are kept
AV	**agent availability**	ensuring host does not deny agent service or proper transmission
AY	**agent anonymity**	keeping the identity of the agent anonymous
HA	**host authenticity**	ensuring correct identity of the (dispatching, executing, or trusted) host
HN	**host non-repudiation**	ensuring host commitments are kept
HP	**host data privacy**	ensuring host input data is private
HY	**host anonymity**	keeping the identity of the host anonymous
HV	**host availability**	ensuring agent does not deny the host service
HI	**host integrity**	keeping server data free from unauthorized agent observation

Fig. 3. Agent/Host Security Requirements

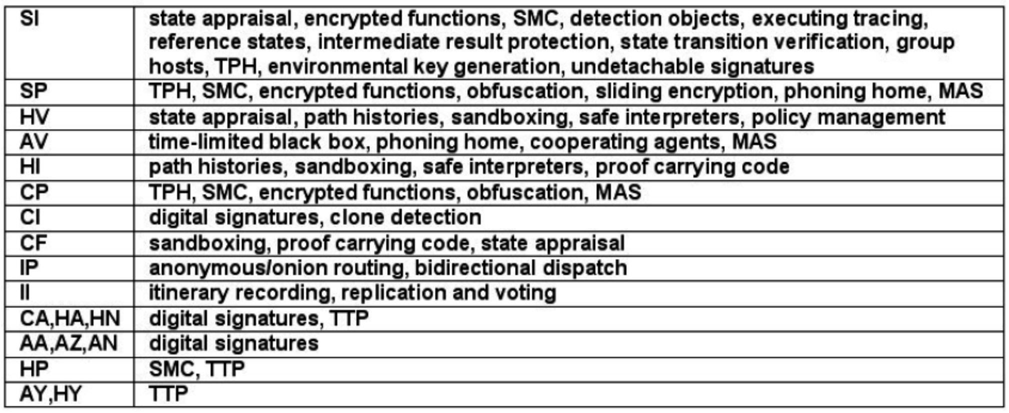
In our model, an application designer can specify varying levels of trust for different security requirements–giving freedom for the application to trust more in one security aspect but less in others. For example, principal A can trust principal B in regards to agent code privacy (reverse engineering), but not in regards to agent execution integrity (partial result alteration). We give three models in section 4 that describe classes of applications with varying levels of trust.

3.4 Defining Security Mechanisms

Security requirements formulate the desire to guarantee privacy, integrity, availability, authentication, or non-repudiation. Security mechanisms are targeted at enforcing one or more these requirements. Both application owners and potential agent execution environments have vested interest in the mechanisms that are used to enforce security–whether they prevent or detect malicious behavior and what aspect of protection they provide. No single security mechanism can address every security requirement for a mobile agent system. Some efforts have joined *mechanisms* and *requirements* at an application level [18] so that several mechanisms together enforce desired security levels.

A large body of literature can be found that details proposed security mechanisms for mobile agent systems. Though we are limited by space, thorough analysis of agent security mechanisms can be found in [1,2,3,12]. Host-based mechanisms protect a host from malicious agents and include sandboxing, safe interpreters, code signatures, state appraisal, proof carrying code, path histories, and policy management. Agent-based mechanisms protect the agent from malicious activity outside of itself and several commonly referenced mechanisms include encrypted functions, detection objects, replication with voting, reference states, time-limited execution, digital signatures, phoning home, anonymous routing, trusted third parties (TTP), secure multi-party computation (SMC), multi-agent systems (MAS), intermediate data result protection, undetachable signatures, environmental key generation, execution tracing, and tamper-proof hardware (TPH).

Protection mechanisms can allow agent transactions in spite of unknown or poor trust environments. Wilhelm *et al.* [19], for example, make a strong argument that the presence of trusted hardware and appropriate execution protocols can effectively shield private application data from observation on untrusted remote servers and mitigate lack of trust. A principal can have different trust levels for different requirements, e.g., Alice may trust Bob to execute her agent without reverse engineering it (an expression of code privacy), but may not trust Bob to execute the agent without looking at previous results from other hosts (an expression of state privacy). When the desired trust level is not adequate between parties in the agent application, the presence of security mechanisms is used to enforce specific security requirements so that agent-based tasks can be accomplished.



SI	state appraisal, encrypted functions, SMC, detection objects, executing tracing, reference states, intermediate result protection, state transition verification, group hosts, TPH, environmental key generation, undetachable signatures
SP	TPH, SMC, encrypted functions, obfuscation, sliding encryption, phoning home, MAS
HV	state appraisal, path histories, sandboxing, safe interpreters, policy management
AV	time-limited black box, phoning home, cooperating agents, MAS
HI	path histories, sandboxing, safe interpreters, proof carrying code
CP	TPH, SMC, encrypted functions, obfuscation, MAS
CI	digital signatures, clone detection
CF	sandboxing, proof carrying code, state appraisal
IP	anonymous/onion routing, bidirectional dispatch
II	itinerary recording, replication and voting
CA,HA,HN	digital signatures, TTP
AA,AZ,AN	digital signatures
HP	SMC, TTP
AY,HY	TTP

Fig. 4. Agent Security Requirements and Mechanisms

Figure 4 lists various agent security mechanisms and the corresponding requirements they are intended to enforce. Though not an all-inclusive list of mechanisms, principals rely on such mechanisms to protect themselves. Certain mechanisms are preventative in nature, not allowing malicious behavior a priori. Other mechanisms rely on a posteriori information to detect whether unauthorized actions occurred to either the agent or the host. Some mechanisms readily fit into both categories and the clear delineation is not important. In general, preventative mechanisms are desired over detection mechanisms when available because they are stronger, but may come with more overhead or complication. Detection is normally a less stringent means of security because protection has already been violated in some way–but at least is caught.

3.5 Linking Requirements and Mechanisms with Trust Enhanced Decisions

Trust decisions in pervasive scenarios are based on information from two primary sources: personal observations (previous interactions) and recommendations from other parties (transitive or delegated trust). Spy agents [20] can be used to build and maintain a trust profile by validating behavior in small interactions. Trust-earning actions can build relationships which in turn are used to decide which security mechanisms need to be used to achieve the level of security desired by the application owner.

We use trust in the mobile agent environment to affect security mechanism selection, agent itinerary, policy decisions, and code distribution. For example, if the application owner (AO) has non-determined (ND) or low (U) trust toward any of the prospective hosts, the owner may require *detection* mechanisms to guarantee agent state integrity or agent state privacy. If no trust (HU/U) exists at all, the AO may require more stringent *preventative* mechanisms to enforce agent state integrity/privacy. If the AO has full trust (T/HT) in prospective hosts, no security mechanism may be required in order for the agent to be dispatched and executed on those particular hosts.

To link agent security mechanisms with application requirements, initial, recommended, and first-hand trust is processed and the framework renders a mechanism-based decision that meets the security objectives of the principals involved. Highly trusted and trusted principals will tend to yield no requirement for security mechanisms. Non-determined trust will tend to require $detection - oriented$ mechanisms while untrusted relationships will tend to demand $prevention - oriented$ mechanisms. Migration decisions may also be determined based on trust level.

Figure 5 depicts the inputs and outputs to our trust determination process for mobile applications with the outcome being selection of one or more mechanisms that will meet bidirectional requirements between principals. An appropriate (set) of host-based mechanisms and an appropriate (set) of agent-based mechanisms is one of the outcomes of the trust algorithm. We propose that the *initial* set of trust relationships can be generalized based on the application environment of the mobile agent and represented by an application model. The initial trust component of the trust-based decision seen in figure 5 is defined by a set of relationships (the application model) which is context-dependent on the application.

Given our definition of principals, trust, and applications, we support a

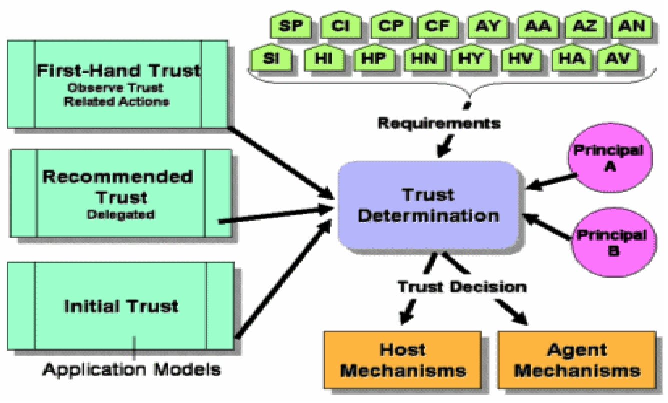


Fig. 5. Trust Decisions for Mobile Agent Security

framework to exercise security. Figure 6 pictorially summarizes the components of our trust framework for mobile agent applications using standard UML generalization and compositional notation. Trust is enforced or established by security mechanisms; applications link the trust expectations of principals through security requirements to a trust level, foreknowledge, and timeliness. An application environment can be used to generalize the adversarial nature that exists among principals: we call this generalization an *application level security model* which describes an initial set of trust relationships between application parties.

When considering the agent life cycle and the initial binding of trust at various stages we formulate the following notions:

(i) creation and development of code bind trust to a code developer

(ii) ownership of an agent binds trust to an application owner

(iii) dispatching an agent binds trust to a dispatching host,

(iv) execution of an agent binds trust to all prior hosts an agent has visited plus its dispatcher

(v) migration binds trust to the next host in the agent itinerary

(vi) termination binds trust of the entire application to the entire set of execution hosts and the network environment

Our model allows trust to be earned or degraded based on actions observed over time. Given a set of trust relationships that exist between principals, several trust-based decisions are supported by the architecture: which agent/host security mechanism to use, which hosts an agent can include in the itinerary, which parts of the agent code are executed by the host, which agents are allowed host resources, which principals can share policy information, whether trust levels can increase (whether you are allowed to recover from negative trust), and whether or not trust recommendations from other parties should be given merit to include how much weight they are given.

Observable trust-related actions can change trust levels among mobile

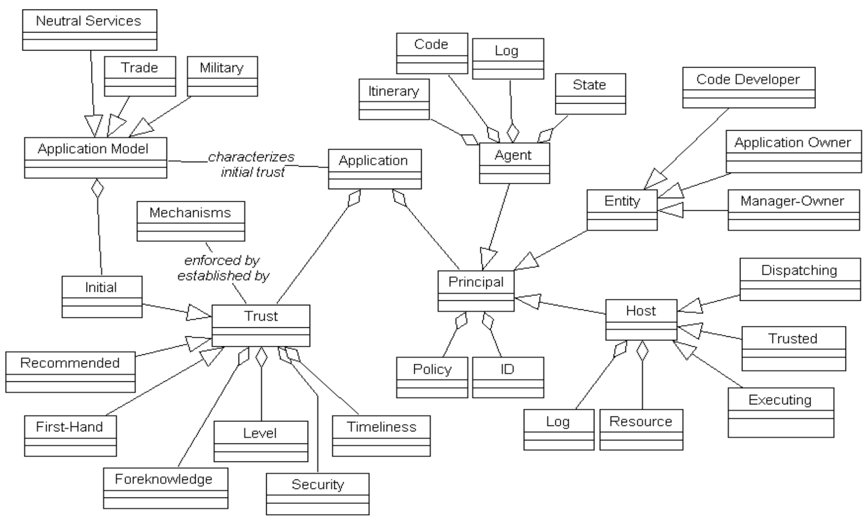


Fig. 6. Trust Framework

agents and hosts. Trust relationships evolve from initial trust according to a predefined set of rules–which represent a security policy. In our previous work on trust in ad-hoc networks [7], four different categories of trust acquisition are formulated which we apply in the mobile application context: trust-earning actions over time, trust-earning actions by count, trust-earning actions by magnitude, and trust-defeating actions. We address the issue of trust-based actions and the effects of trusted third parties on mobile agent execution in [12].

4 Application Level Security Models

Models come in many shapes and sizes. In all cases, they are used to focus and detail a particular aspect of a problem. *Security* models help test completeness of a security policy or can verify whether an implementation fulfills a set of requirements. *Application* models for multiple agent systems describe how agents accomplish tasks based on an underlying pattern such as *publish/subscribe*, *courier/broker*, and *supervisor/worker*. The application context determines security responsibilities of principals and limits trust award to occurring only through specific interactions.

In our security framework we establish trust three ways: *initial* trust, *acquired* trust, and *recommended* trust (see figure 5 and 6). Over time, trust will change based on observed actions and delegated decisions–which we do not cover for space limitations. Application scenarios dictate how principals are derived and how they act towards each other. We can use scenarios to set boundaries on whether trust can be acquired over time–whether we can promote principals from untrusted (U) to non-determined (ND) or from non-determined (ND) to trusted (T).

In this paper we introduce the notion that mobile agent applications can have initial trust relationships (between agents and hosts for example) that are based on a common trust environment. This initial trust is the starting point for trust-enhanced security decisions. We provide three examples of real world environments that reflect mobile agent applications that share common trust assumptions. These initial trust relationships couple the security requirements and trust levels of various participants. As a result, agents in an application can initially determine which security mechanisms will be used and hosts can initially specify a required set of security mechanisms.

4.1 The Military Model

The military model is based upon the notion that a wall or ”Maginot Line” exists between friendly and adversarial entities. Within friendly borders, entities are known to each other as authenticated, highly-trusted principals. At some point, however, a given principal may be taken captive by an adversary. This captured entity (whether a host or agent) may continue to function passively in order to discover information or leak secrets on behalf of the capturer. Captured entities may become overtly malicious by delaying and denying service, corrupting friendly communications, and attacking privacy and integrity of group operations.

The military model formulates common application characteristics between principals and helps focus security requirements. For instance, although hosts might be ad-hoc or mobile, they are verified by a managerial entity within the environment. Figure 7 illustrates how an initial set of trust relationships capture this notion: every dispatching host/agent originator (DH/AO) has a ”known” and ”trusted” relationship with every executing host (EH) to begin with. This is indicated by ”K/T” in the row for DH/AO that intersects the EH column in figure 7. In the military model, trust relationships are implicit as long as the identity of the principal can be verified. Principals are commonly known beforehand, so more trust is granted for both authenticated agents and hosts.

The military model fits requirements and trust relationships where the use of trusted third parties, trusted hardware, group security operations, multiple security levels, multiple trust levels, and distinct organizational structures exist. This environment is found in many corporate infrastructures (as well as the military itself) where a trusted computing base is financially possible or mandated. Implicit trust among principals allows hosts to efficiently work in cooperation with agents to provide mutual prevention and detection services.

The military model also suggests a common set of agent characteristics, where agents are designed, developed, and deployed by a centralized authority. In industry, development can be delegated to outsourced teams or an information technology department with in-house programmers. The military model reflects programming environments where only authorized mobile agent applications are used and agents act as peers. Other initial trust models can reflect agents that take on adversarial roles with one another. Of course even corporate divisions have proprietary and sensitive information that may require protection.

In the military model, agents may still have requirements for integrity and privacy, but their identity, safety, authorization, and authentication can be verified within a circle of trust. The military model also places less emphasis on distinction between executing and dispatching hosts. Agent servers are used interchangeably in some cases as the dispatcher and in other cases as the execution host. Initial trust in this model is indicative of many real world computing paradigms where agent-based applications accomplish group collaboration, systems management, and information gathering. The key feature is that a centralized management domain exists.

Figure 7 summarizes the military model initial trust relationships and describes how δ is initialized. Two simplifying assumptions are illustrated: the application owner (AO) and the dispatching host (DH) are equivalent for trust purposes and the code developer (CD) has equivalent trust to the agent (A). The matrix also depicts, for example, the dispatching host (DH) / application owner (AO) initially knows and trusts (K/T) all executing hosts (EH). It also illustrates how the AO knows and trusts all agents (A/CD) it will use.

Based on this initial trust relationship set, the trust algorithm dynamically determines acquired or recommended trust. Acquired trust mechanisms (the acquisition of negative trust) facilitate discovery of infiltrators. These relationships also determine the security mechanisms required by each host or agent. The underlying assumption of the military model is that some agents or hosts will fall under "enemy" control. Two primary security-related tasks consume the majority of time in the military model: 1) protecting insiders from outsiders and 2) detecting whether or not an agent or host has been compromised or captured.

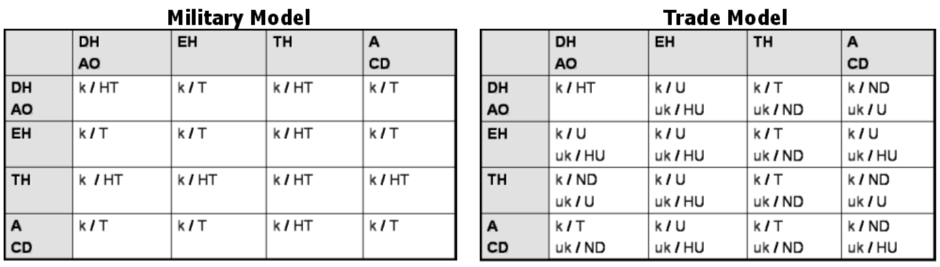
The issue of the latter security task becomes detection of anomalous or malicious behavior and removal of malicious parties from the circle of trust. This scenario best represents application environments where there is a peer (non-adversarial) relationship within the community of trust that applies to both agents and hosts. As figure 7 illustrates, the initial trust relationships among all principals in the system begin at a known and trusted level and when trusted servers are used (TH), they are "highly trusted" (HT).

The role of trusted third-parties and trusted hardware, as well as coalition security mechanisms, becomes focused on identifying principals that have violated the circle of trust or are attempting to gain access to the circle of trust. A strong military model may require that all executing hosts be equipped with tamper-proof hardware. Other application scenarios are better suited for expressing e-commerce interactions, discussed next.

4.2 The Trade Model

A second toy example we present is the trade model: it captures the intuition of a competitive interaction among actors that are all bargaining for resources. Such an environment could also be termed an *economic* model, a *buy/sell* model, or a *supply/demand* model where economic benefits are in view. This application scenario is indicative of the Internet model of computing where e-commerce mobile agents might be deployed. It describes applications where disjoint communities of mobile agent dispatchers want to use services or obtain goods from a set of host commodity or service providers. Agent literature routinely represents such a model as an agent dispatched to find an airline ticket among a group of airline reservation servers-accomplishing the transaction autonomously while finding the best price within the user constraints.

Figure 7 illustrates the initial trust relationships for security requirements in the trade model and depicts the adversarial relationship among principals. In this scenario several trust facets are expressed: 1) buyers (application owners) do not trust sellers (hosts) to deal honestly with them; 2) sellers do not trust other sellers to work for their best interest; 3) buyers do not trust sellers to act non-maliciously; and 4) buyers are in competitive relationships with other buyers for the same goods and services. Initial relationships between dispatching hosts/application owners (DH/AO) and executing hosts (EH) thus have an implicit untrusted (U) relationship for parties that are known (K) and an implicit highly untrusted (HU) relationship for parties that are unknown (UK)-seen in the figure 7 matrix. Executing hosts in the matrix (EH) have untrusted relationships (U/HU) with other executing hosts, whether known (K) or unknown (UK). The buyer adversarial relationship is expressed by the initial trust between agents/code developers (A/CD) as being non-determined (ND) or highly untrusted (UH) in the case of known/unknown parties.



Military Model

	DH AO	EH	TH	A CD
DH AO	k / HT	k / T	k / HT	k / T
EH	k / T	k / T	k / HT	k / T
TH	k / HT	k / HT	k / HT	k / HT
A CD	k / T	k / T	k / HT	k / T

Trade Model

	DH AO	EH	TH	A CD
DH AO	k / HT	k / U uk / HU	k / T uk / ND	k / ND uk / U
EH	k / U uk / HU	k / U uk / HU	k / T uk / ND	k / U uk / HU
TH	k / ND uk / U	k / U uk / HU	k / T uk / ND	k / ND uk / U
A CD	k / T uk / ND	k / U uk / HU	k / T uk / ND	k / ND uk / HU

Fig. 7. Military and Trade Model Initial Trust Relations

The largest number of perceived mobile agent application possibilities typically fall into the trade model in terms of security requirements. In this context, principals are not necessarily known before interaction takes place and there is, in most cases, no trust or foreknowledge between users that want to execute agents and hosts that would like to execute agents. This model relies more on acquired or delegated trust decisions and reflects that executing hosts are as equally distrusting of agents and other executing hosts.

Application owners see hosts as implicitly untrusted in the sense that there is economic benefit to gain if hosts alter the execution integrity of their agents or maliciously collude together.

4.3 The Neutral Services Model

As a third notion to capture application level security requirements, we define the neutral services model with the intuition that a service (or set of information) is acquired by one or more agents. Providers of services do not themselves have an adversarial relationship, but they may be viewed as having disjoint communities of trust. The primary difference in the neutral services model and the trade model is that communities of hosts exist with no adversarial relationship among themselves. These communities are essentially neutral in terms of their commitments to each other–neither friendly nor hostile.

This model is suited for application environments designed around information or database services. Information providers typically have no economic gain from altering the results or influencing the itinerary of agents that they service. Hosts provide services honestly in the sense that they would not alter the path or intermediate data results of an agent or induce denial of service. Service providers can and in most cases do charge a small fee for the use of their service, however. What might be of interest to a dispatching application owner in this model is whether or not its agent is billed correctly. In this respect, if information providers charge for their service, it is to their benefit to alter the execution integrity of an agent so that the agent is charged for more than was legitimately received.

Adversarial relationships exist between agents from the "client" community and hosts in the "server" community, but there is not necessarily a trust or distrust of hosts within a given community. Neutral hosts see no benefit from altering an agent that might be carrying results from other hosts or from preventing them from visiting other hosts. Hosts in this realm are in essence a "one-of-many" information provider. This paradigm may not fit a search engine model where a mobile agent visits and collates search results from let's say Google, Yahoo, and Alta Vista. In that case, it may be of interest to one of these engines (who get benefit from every agent hit since advertisers might pay more for a more frequently visited search engine) to alter the itinerary or search results of other hosts. It might also benefit a search engine in this example to maliciously alter search results of other engines carried by the agent to be "less useful" so that their results look better to the application owner. For cases like this, the trade model would fit better in terms of security requirements.

The type of protection that is needed in the neutral services model revolves primarily around the execution integrity of the agent. To that effect, hosts that bill customers for usage might be tempted to cheat and wrongly charge agents for resources they did not use. Likewise, agents may want to falsely convince a host that no service or information was gathered, when in fact it was. Trusted relationships between neutral third parties are also more conducive in this environment and trusted third parties may interact with various communities of service providers themselves on behalf of other users.

5 Conclusions

When application development is in view, it is helpful to have methods that help transform requirements into implementation. We present a trust model to support mobile agent application development that links trust relationships and expression with both security requirements and security mechanisms. We further define the notion of an application security model that can seed our mobile agents trust framework.

We give three examples of such models and characterize how initial trust relationships can be generated based on the trust assumptions between parties involved in mobile agent interactions. The usefulness of such models is that developers and researchers can reason about security requirements and mechanisms from an application level perspective and integrate trust-based decisions into the mobile agent security architecture.

References

[1] Jansen, W., and T. Karygiannis, *National Institute of Standards and Technology, Special Publication 800-19-Mobile Agent Security*, August 1999.

[2] Bierman E., and E. Cloete, *Classification of malicious host threats in mobile agent computing*, Proc. of 2002 Annual Research Conf. of the S. African Inst. of Comp. Scientists (2002), Port Elizabeth, 141–148.

[3] McDonald, J. T., A. Yasinsac, W. Thompson, *Taxonomy for Defining Mobile Agent Security*, submitted to ACM Computing Surveys, May 2005, URL: http://www.cs.fsu.edu/research/reports/TR-050329.pdf.

[4] Yahalom, R., B. Klein, and T. Beth, *Trust relationships in secure systems-A distributed authentication perspective*, Proc. of IEEE Symposium on Research in Security and Privacy (1993), 150–164.

[5] Blaze, M., J. Feigenbaum, and J. Lacy, *Decentralized Trust Management*, Proc. IEEE Conference on Security and Privacy, Oakland, CA, May 1996.

[6] Grandison, T., and M. Sloman, *A Survey of Trust in Internet Applications*, IEEE Comm. Surveys **4**th Quarter (2000).

[7] Burmester, M., and A. Yasinsac, *Trust Infrastructures for Wireless, Mobile Networks*, WSEAS Transactions on Telecommunications **3**:1 (2004), 377–382.

[8] Tan, H. K., and L. Moreau, *Trust Relationships in a Mobile Agent System*, G. Picco (ed.), 5th IEEE Intl Conf. on Mobile Agents, LNCS **2240** (2001), Atlanta, Georgia, Springer-Verlag, 15–30.

[9] Robles, S., J. Mir, and J. Borrell, *MARISMA-A: An Architecture for Mobile Agents with Recursive Itinerary and Secure Migration*, Fischer, K., and D. Hutter (eds.), Proc. of SEMAS'02, Bologna, Italy, 2002.

[10] Kagal, L., T. Finin, and A. Joshi. *Developing secure agent systems using delegation based trust management*, Fischer, K., and D. Hutter (eds.), Proc. of SEMAS'02, Bologna, Italy, 2002.

[11] Lin, C., *et al.*, *On the Design of a New Trust Model for Mobile Agent Security*, 1st Intl Conf. on Trust and Privacy, LNCS **3184** (2004), 60–69.

[12] McDonald, J. T., and A. Yasinsac, "Trust in Mobile Agent Systems," Technical report TR-050330, Dept. of Comp. Science, Florida State University, 2005.

[13] Gambetta, D. *Can We Trust Trust?*, Gambetta, D. (ed.), Trust: Making and Breaking Cooperative Relations (1990), Basil Blackwell, Oxford, 213–237.

[14] Cahill, V., *et al.*, *Using Trust for Secure Collaboration in Uncertain Environments*, IEEE Pervasive Computing **2**:3 (2003), 52–61.

[15] Capra, L., *Engineering Human Trust in Mobile System Collaborations*, Proc. of the 12th Intl Symposium on Foundations of Sftwre Eng., Nov. 2004.

[16] Bellavista, P., A. Corradi, C. Federici, R. Montanari, and D. Tibaldi, *Security for Mobile Agents: Issues and Challenges*, chapter in "Handbook of Mobile Computing" (2004).

[17] Antonopoulos, N., K. Koukoumpetsos, and K. Ahmad, *A Distributed Access Control Architecture for Mobile Agents*, Proc. of Intl Network Conference, Plymouth, UK, July 2000.

[18] Claessens, J., B. Preneel and J. Vandewalle, *(How) can mobile agents do secure electronic transactions on untrusted hosts?*, ACM Transactions on Internet Technology (2003).

[19] Wilhelm, U., S. Staamann, and L. Buttyan, *On the Problem of Trust in Mobile Agent Systems*, IEEE Network and Distributed Systems Security Symposium, San Diego, CA, 1999, 11–13.

[20] Kalogridis, G., C. J. Mitchell, and G. Clemo, *Spy Agents: Evaluating Trust in Remote Environments*, Proc. Intl Conf on Security and Management, Las Vegas, NV, 2005.

[21] Esfandiari, B., and S. Chandrasekharan, *On How Agents Make Friends: Mechanisms for Trust Acquisition*, Proc. 4th Wkshp on Deception, Fraud and Trust in Agent Societies (2001), 27–34.