# Securing Multi Agent Societies

Rahul Singh
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
USA
kingtiny@cs.cmu.edu

Katia Sycara
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
USA
katia@cs.cmu.edu

## ABSTRACT

When large Multi Agent Systems operate over open platforms such as the Internet, there is the possibility that some agents will endeavor to disrupt the MAS with the aim of achieving their own agendas at the cost of the others. For the protection of the agents in it's domain, each society of agents should make available certain services that allow agents to detect and avoid such attempts by malicious third parties. One of these is a rigorous method of identifying agents, so that each agent's actions can be referred to, recorded and evaluated. A robust identification mechanism can allow the construction of higher level-functions such as authentication, non-repudiation and secure communication. Agents can then use these basic functions provided by the MAS infrastructure along with their beliefs, current state of the MAS and the current goals to make decisions about trust. In this paper we present one architecture that allows agents to be named robustly, even in open environments, and build upon this scheme to provide MAS services that can allow agents to authenticate each other, digitally sign communication tokens and encrypt messages and data that is exchanged. We explore possible scenarios of the agent life in an open agent society and give an implementation independent architecture to secure agents and their environment.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; K.6.5 [**Management of Computing and Information Systems**]: Security and protection

## General Terms

Multi Agent Security

## Keywords

Multi Agent Systems, Security

## 1. INTRODUCTION

In open systems deployed over a platform such as the Internet the requirements for robust mechanisms for security are considerable due to the large number of exploits possible. Depending upon the type of system and the services provided the threats may vary. Foner presents a number of exploits [13] that may be carried out against the Yenta [12, 14] matchmaking system and explores possible means of counteracting them while Wong & Sycara [27] present a security architecture for the RETSINA [24] agent system with emphasis on agents and their deployers to allow for human accountability for the actions of their deployed agents. Finin et. al. [20, 26] describe a high level mechanism for security in communication protocols with a security infrastructure for KQML [11]. In addition to this there are a number of ongoing efforts for the development of mechanisms and architectures for security frameworks for the variety of distributed systems that are slowly emerging on the Internet. XKMS [15] intends to provide a mechanism for key management in the domain of distributed web services and S2ML [4, 5] provides a framework for marking up security assertions and policies in XML.

Most security frameworks developed focus on mechanisms for security and trust across human-agent boundaries and apply to agent systems that interact closely with human society. Imagine an agent society consisting of a number of agents that have no contact with any human beings at all save for ones that accept commands (goals) and those that return results. All the other agents in the system do not need to interact with any entity except their peers in the agent system. They are invisible to the humans issuing commands and receiving results. Hence any malicious activity on their part cannot directly affect the human domain. Any malicious activity on the part of an agent can only directly affect other agents in the system. The only recourse for the victim of such an attack is to record the disruptive nature of the attack, the name of the offending agent and to refrain from further interaction with that agent thus boycotting that agent. As long as all the members of the MAS adopt such a policy the actions of the attacker will have no effect on any agents in the system. Hence it is only by disrupting the MAS infrastructure that a malicious agent can affect the lives of the agents in the system.

In order to prevent agents from causing disruptions in an MAS two major components are required.

1. Agents should be aware of their peers and their actions and be able to identify threats and malicious activity so that they may be avoided in the future
2. The MAS infrastructure should have mechanisms to protect itself from attacks that can cause failures in the agent society

Security mechanisms at the infrastructure level can be provided as services to agents and be seamlessly integrated with other MAS services such as naming, location and discovery. Agents can then use these services along with their current beliefs, goals and the state of the environment to make assertions about trust.

## 2. THE CONCEPT OF TRUST

Trust is a term that is widely used in a number of contexts and most often associated with security and privacy. Gmytrasiewicz et. al [17] outline a mechanism of communication among agents that allows the participants to select the best messages to exchange based on their rational behaviour and expected utilities. The claim is that it sets the grounds for building trust within the context of a conversation. The actual definition of trust or a theory to prove its depth (oncs trust has been established) is however not presented.

Teng et. al. [25] show how Dempster-Schafer theory of evidence can be used to design a theory of trust and its propagation. While claiming that trust cannot be accurately measured quantitatively they nevertheless outline a number or variables and parameters to define trust. It is on the basis of these variables that they then build trust matrices to capture the level of trust between two parties that are interacting. The matrices are defined after trust has been established and they reflect that level of trust. There is no general theory presented that can determine the appropriate level of trust that an agent $a$ should put in agent $b$ when the transaction is initiated.

It may argued that a simple solution is to start with no trust and then build trust based on the history of previous transactions. This approach is flawed since there are examples (in human society) where trust is established based on factors other than past interactions and even when there have been no past interactions.

Eckel & Wilson [10] present results of experiments on human subjects where facial expression plays an important factor in the behaviour of the individuals interacting. They conclude that trust is something that depends largely upon (among other things) the beliefs of the participants.

In the context of computer science and e-commerce the term trust is most widely used in the field of security and privacy. Many research papers speak of managing trust, establishing trust through authentication and verifying trust in a variety of systems. All these approaches base their theories on the widely accepted security mechanisms developed by research in the field of cryptography. The CCITT standard X.509 [7] is extensively used over the Internet to authenticate various parties in dialogue, and in SSL [6] for secure communication. There is a general feeling that the mere possession and verification of public key certificates (i.e. X.509 certificates) is enough to establish trust between two parties in a communication. This is evidenced by the fact that the phrase "trusted third party" is frequently used to describe entities that have X.509 certificates that have been verified. But as Gerck [16] explains, the simple employ of these excellent techniques does not establish trust. If agents $a$ and $b$ both possess verifiable X.509 certificates then $a$ can be sure that any communication with $b$ is indeed with an entity whose identity is $b$. Similarly $b$ can establish beyond reasonable doubt that any documents that it received from $a$ did indeed originate from $a$. Additionally either party can also use information in each other's X.509 certificate to ensure that none of the conversation can be overheard by a third party. However, none of the above implies that $a$ trusts $b$ or that $a$ will not lie to $b$. In the context of a MAS it does not imply that an agent $a$ will not falsely advertise services that it does not intend to provide.

We believe that trust is a function of the high level cognitive processes that occur in an agent. Hence the level of trust that an agent establishes in another agent in the MAS will depend upon the current state of the agent, the state of the MAS, the current goal being achieved and other constraints that affect the life of an agent in its environment.

We now present a design for a security infrastructure to support a society of agents based on work previously conducted by Wong & Sycara[27]. We present an implementation independent architecture for enforcing security policies in a distributed MAS, so that agents can proactively and autonomously protect themselves from attack without human intervention.

## 3. AGENT NAMING IN OPEN SYSTEMS

An agent in a multi agent system is referred to by its name, much in the same way that humans are referred to by their names in society. An agent's name must be unique in within the domain of the MAS so that every agent may refer to every other agent unambiguously. Within a closed system with any number of agents this poses a minimal problem. The system designer merely ensures that every agent in the system is assigned a name that is unique. With no interactions with any other entity or system (since this is a closed system) agents may use the same names for themselves in any number of life cycles.

Open systems however have no human "arbitrator" that ensures that agents' ID's (or names) will be unique within every lifecycle. There are no guarantees that agents possessing a certain name will not come across another agent with the same name. Internet naming suffers from the same problem and uses DNS to assign unique names to websites. However the volatile nature of a MAS (especially when one considers the possibility of mobile agents) makes such a solution infeasible.

In a lookup based MAS such as RETSINA [24] we present a service centric model as a solution to the agent naming problem in open worlds such as the Internet. In this scheme every MAS consists of its domain of operation and every agent within this domain is uniquely named. This mechanism is facilitated by an Identity Service (IS) that runs within each MAS. The IS ensures that every agent within

its domain (the domain of the MAS) is assigned a unique name. The IS also doubles as a certification authority (CA) and issues Identity Certificates (public key certificates) to agents as proofs of identities. Interaction with the IS is carried out with the exchange of Identity Request Records (IRR) which are submitted to an IS for approval and certification. We formally define an Identity Request Record as follows.

*Definition 1.* Let $I_R$ be an Identity Request Record. Then $I_R$ is represented by a tuple defined as

$$I_R = (N_s, C_s, K_s) \tag{1}$$

where

- $N_s$ is the agent's self generated name that will be assigned to the agent if it is unique within the domain of the IS. If the name is not unique the IRR request fails. If this is null then the IS will generate and assign a unique name to the agent.
- $C_s$ is the optional common name of the agent and is usually human readable and not required to be unique. It may also be null.
- $K_s = (E_s, D_s)$ is the agent's generated assymetric key pair such that $\forall m, E_s(D_s(m)) = D_s(E_s(m)) = m$ [1]. If this is not specified the IS will generate it for the agent.

An agent that specifies none of the fields in the IRR is guaranteed an identity. It may also submit an IRR with certain fields specified thus putting constraints on the identity that it may be issued. If the value for $N_s$ is not unique within the domain of the IS then the request for an identity will fail and the agent will need to submit a new IRR. The response to a valid IRR is an Identity Certificate that establishes an identity for the agent. We formally define the response to a valid IRR as follows.

*Definition 2.* Let $w$ be an Identity Certificate. Then $w$ is represented by a tuple defined as

$$w = (v, n, A_s, N_i, p, N_s, E_s) \tag{2}$$

where

- $v$ is the version of the identity certificate issued.
- $n$ is the serial number of the identity certificate.
- $A_s$ is the signature algorithm used for signing the certificate.
- $N_i$ is the unique name of the IS that is issuing the certificate.
- $p$ is the period of validity of the certificate.
- $N_s$ is the assigned unique name of the agent i.e. the subject of the certificate.
- $E_s$ is the public key of $N_s$ corresponding to $D_s$ such that $\forall m, E_s(D_s(m)) = D_s(E_s(m)) = m$.

The wary reader will notice that the concept of the IS and IRR are analogous to Internet Naming Authorities (NA) and X.509 Certificate Signing Requests (CSR). We have combined the NA and the CA into a MAS infrastructure entity that performs both functions. Identity Servers may also form a hierarchy within the domain, effectively splitting the MAS into subdomains. The name of the IS issuing a certificate is used in the $N_i$ field in definition 2 and thus uniquely identifies the MAS domain within which the agent name $N_s$ is used.

## 4. VALIDATING AGENT NAMES

All agents in the MAS receive an Identity Certificate $w$ in response to a valid IRR. The Identity Service issuing the certificate also possesses a valid identity (and a corresponding identity certificate[2]) which is used to populate the $N_i$ field in defintion 2. The issued certificate is a digitally signed document that can be used to verify the identity of the agent. Digital signatures [21] are based on the Public Key Cryptosystem (PKI)[21, 2] and allow a signed document to be verified for authenticity. Agents and MAS infrastructure components can use this property of Identity Certificates to ensure that a certificate presented as proof of identity is genuine. We formally define the predicate $sign(d, w)$ as follows.

*Definition 3.* Given any document $d$ define

$$sign(d, w) = (d, d') \tag{3}$$

where

- $(d, d')$ is the signed document consisting of the original document and its signature
- $d' = D_s(hash(d))$ is the signature on the document
- $hash(x)$ is a hash function (such as SHA [3]) that returns a unique hash of $x$
- $w = (v, n, A_s, N_i, p, N_s, E_s)$ is the identity certificate belonging to the signer of the document

The signature on a particular document $d$ can be verified using the $verify(d, d', w)$ predicate defined as follows.

*Definition 4.* Given any signed document $(d, d')$ define

$$verify(d, d', w) = true \Leftrightarrow hash(d) = E_s(d') \tag{4}$$

where

- $(d, d')$ is the signed document consisting of the original document and its signature

[1] $E_s(x) = m_1$ & $D_s(x) = m_2$ are the results of respectively applying the public and private parts of the asymmetric key $K_s$ to $x$ in order to obtain cyphertext $m_1$ & $m_2$.

[2] An IS may obtain an identity certificate from another IS or from a commercial Certification Authority (CA)

- $d^{'} = D_s(hash(d))$ is the signature on the document
- $hash(x)$ is a hash function (such as SHA [3]) that returns a unique hash of $x$
- $w = (v, n, A_s, N_i, p, N_s, E_s)$ is the identity certificate belonging to the signer of the document

Any agent that is issued an identity certificate $w$ possesses a signed document $(w, w^{'})$, where $w^{'}$ is the signature on the certificate. The IS issuing $w$ obtains $(w, w^{'})$ by generating $w$ and applying $sign(w, W)$, where $W$ is the identity certificate belonging to the IS. Any other third party can ascertain the validity of the certificate $w$ by applying $verify(w, w^{'}, W)$. So long as the third party can ascertain that $W$ is not forged (by recursively applying $verify(W, W^{'}, W^{''})$, where $W^{''}$ is the identity certificate of the parent IS or root CA) it can verify that the certificate $w$ is genuine.

# 5. THE MECHANICS OF SECURITY IN AN AGENT SOCIETY

Numerous papers [18, 27, 13, 20, 26] outline the possible threats to a multi agent system and we draw upon research in that area to look at the various methods of attack that need to be protected against. For completeness we briefly mention them here.

- Falsification or corruption of documents in the MAS infrastructure services.
- Eavesdropping of communication by malicious third party agents.
- Replay attacks by malicious agents with the aim of producing desired reactions from their peers or the MAS infrastructure.
- Corruption of exchanged messages by a third party, also known as a "Man in the Middle" attack.
- Spoofing attacks where an agent falsely masquerades as another agent.
- Denial of Service attacks where a malicious agent may overload the service provider thus making it inoperative.

Regardless of the architecture being developed, the major threats common to distributed systems and multi agent systems are some form of those listed above.

## 5.1 Fundamental Requirements of an MAS Security Mechanism

Counteracting the threats outlined in section 5 above requires the adoption of certain security policies and protocols. The details of the policy used usually depends upon the system being developed and takes the assumptions and constraints into account. However in order for any security implementation to be robust the three primitives of authentication, non-repudiation and secure communication must be implemented. We critically examine these primitives and present formal predicates that allow assertions to be made within the context of a Multi Agent System.

### 5.1.1 Authentication

Prior to any communication, agents must be able to authenticate each other in order to ensure that that the communication partner is indeed the one that it claims to be. We formally define the authentication predicate $auth(a_i, w)$ as follows.

*Definition 5.* Given any agent $a_i$ in the MAS, an Identity Certificate $w = (v, n, A_s, N_i, p, N_s, E_s)$, a private key $D_s$ corresponding to $E_s$ and a random string $s$ we can define $auth(a_i, w)$ as

$$auth(a_i, w) = true \Leftrightarrow E_s(D_s(s)) = s \quad (5)$$

Definition 5 says that if a principal $N_s$ can successfully decrypt any random string $s$ encrypted by $a_i$ using the public key $E_s$ belonging to principal $N_s$, then $N_s$ is said to have been authenticated by $a_i$. This definition of $auth(a_i, w)$ is based on the challenge response protocol given in figure 1. In this procedure if $a_i$ wants to authenticate the subject $N_s$ associated with the Identity certificate $w$ then $a_i$ generates a plaintext challenge string $s$ (usually a random number) and sends it to $N_s$. $N_s$ encrypts $s$ using $D_s$ (the secret part of $K_s$) to obtain cyphertext $s_1$ and returns it to $a_i$. $a_i$ then applies $E_s$ to $s_1$ to obtain $s_2$. If $s_2 = s$ then the authentication suceeds and $a_i$ can be sure that $N_s$ is the subject associated with the identity certificate $w$.

$a_i \rightarrow N_s$ : $s$, where $s$ is the challenge string.
$N_s$ : $s_1 = D_s(s)$, by applying the secret part of the asymmtric key $K_s$ to $s$
$N_s \rightarrow a_i$ : $s_1$
$a_i$ : $s_2 = E_s(s_1)$, by applying $E_s$ the public part of $K_s$ obtained from $w$. The authentication suceeds if $s = s_2$.

**Figure 1: Challenge Response Authentication Protocol**

$a_i$ may also choose to validate the authenticity of the certificate $w$ using the verify predicate defined above.

### 5.1.2 Non-Repudiation

Parties involved in an exchange must be able to ensure that communication tokens (messages, documents etc.) do indeed originate from the entity that claim to generate them. This can then allow agents to ensure that certain documents that they expect from their peers do indeed originate from them. Non-Repudiation can be achieved by using the predicates $sign(d, w)$ and $verify(d, d^{'}, w)$ defined in section 4. Figure 2 shows the interaction where agent $N_s$ sends a document $d$ to agent $a_i$ which can then verify the signature on the document. Here the transmitting party, $N_s$ digitally *signs* the document $d$ by appending $d^{'} = D_s(hash(d))$ to $d$. The receiving party can verify the signature as belonging to $N_s$ by generating $hash(d)$ and comparing it to $d^{''} = E_s(d^{'})$. This technique ensures that the document $d$ did indeed originate from $N_s$ and was not tampered with in any way enroute to the receiver.

$N_s$ : $(d, d^{'}) = Sign(d, w)$, where $d$ is the document being signed, $d^{'}$ is the signature and $w$ is the identity certificate belonging to $N_s$

$N_s \rightarrow a_i$ : $[(d, d^{'}), w]$

$a_i$ : $result = verify(d, d^{'}, w)$

**Figure 2: Protocol Enforcing Non-Repudiation**

#### 5.1.3 Secure Communication

Mechanisms must be available to ensure that no communication is overheard by a third party. A popular technique is to encrypt all communication, a method used by SSL[6]. More advanced techniques further randomize the commnunication channel in order to prevent attacks that use techniques such as traffic analysis. If a standard protocol implementation such as SSL[6] is not used then the communicating partners may use each other's public keys to encrypt messages they exchange. In cases where it is computationally infeasable to use public key encryption to encrypt messages (eg. when the number or size of the messages exchanged is very large), the agents may use the asymmetric key system to exchange symmetric keys such as DES [19] and then build and maintain a secure communication channel for the duration of the dialogue. In situations where asymmetric encryption is infeasable, methods such as the Diffie-Hellman [9] key exchange may be use to exchange keys that can be used for maintaining a secure communication channel.

### 5.2 MAS Infrastructure Components for Security

In an MAS such as RETSINA [24] the infrastructure plays an important role in the operation of the agents that it supports. Agents usually register themselves with the infrastructure so that their peers in the MAS may know about their existence and interact with them. Typical examples of infrastructure components are an Agent Name Server (ANS) used by many MAS architectures such as RETSINA [24] and JATLite [1]. An ANS is a repository that maps agent names to physical locations that are network dependent. When agent $a$ wishes to communicate with agent $b$, $a$ will request $b's$ network address from the ANS and then attempt to connect to $b$ and initiate a dialogue.

Agents may also register their capabilities with a middle agent [28] thus allowing their services to be discovered. In the RETSINA infrastructure an agent builds an advertisement which describes the capabilities of the agent and the services that it offers and then registers this information with a matchmaker [28, 8] such as LARKS [23, 22]. The matchmaker then maps agent capabilities to agent names and serves requests for service providers. The ANS and the matchmaker together allow agents to find other agents that provide a required service.

There are a number of security issues that arise from such an architecture. Wong & Sycara [27] outline the issues specifically related to ANS and matchmakers and give protocols for securely interacting with them thus preventing agents from falsifying their registrations and maliciously tampering with the registrations of other agents. The security is based on the Agent Certification Authority (ACA) which is incorporated into the MAS. We extend the ACA to also serve agent identification requests as shown in section 3 and formally include it as an MAS infrastructure component that manages agent identities. An IS is a conceptual entity and can be implemented either as part of the ANS or as a separate service or a set of services. Identity Servers can also form a heirarchy and manage subdomains in an MAS.

### 5.3 Interaction Protocols for MAS Infrastructure Components

A Multi Agent system depends upon the infrastructure for its integrity and the seamless interoperation of the agents it supports. Agents register with infrastructure services as they come alive and the interaction with these services must be robust and secure. We now examine the protocols required for robustly interacting with MAS services such as an ANS, and discovery services such as matchmakers.

#### 5.3.1 Simple ANS Registration Protocol

Figure 3 gives the simple ANS registration protocol adapted from [27]. Here an agent $a_i$ interacts with an Identity Service, $I_S$ and and ANS $A_i$. From the point of view of the agent, the interaction consists of two broad steps namely, (i) An agent obtains a unique name for itself and (ii) uses this name to register itself with the ANS. The ANS on the other hand merely has to ensure that the name supplied by the agent is unique within the certification domain. This prevents ambiguities when looking up agents in the MAS. The ANS registration protocol given in [27] required that the ANS actually verify that the agent requesting a registration record be running at the physical location it claims. We have relaxed this assumption without opening any security holes in the system due to the availability of the $auth(a_i, w)$ predicate defined in section 5.1.1. Consider an agent $a_1$ running at physical location $h_1 : p_1$. If $a_1$ registers with an ANS and falsely claims to be running at location $h_2 : p_2$ then no agent will be able to contact it since all lookup requests to the ANS will return $h_2 : p_2$ as the contact point for $a_1$.

If agent $a_2$ is running at $h_2 : p_2$ then any agent, $a_3$ wishing to contact $a_1$ will actually contact $a_2$. This discrepancy will be detected when $auth(a_3, w)$[3] fails. It will thus never be in the interest of an agent to register itself using a fictitious physical address.

$a_i \rightarrow I_S$ : $I_R$, where $I_R$ is an Identity Request Record as defined in section 3.

$I_S \rightarrow a_i$ : $w$, where $w$ is an identity certificate as defined in section 3.

$a_i \rightarrow A_i$ : $(m)$, where $m = register[w, h : p]$. $h : p$ is the physical network address that $a_i$ is running at.

$A_i$ : $result = Verify(w, w^{'}, W)$, where $w^{'}$ is the signature on the certificate $w$ and $W$ is the certifcate belonging to $I_S$.

$A_i \rightarrow a_i$ : $result$.

**Figure 3: Simple ANS Registration Protocol**

[3] $w$ is the identity certificate belonging to $a_1$

### 5.3.2 Simple ANS Unregistration Protocol

An ANS maintains records that map agent names to physical network locations. It is essential that agents do not modify these records unless they have the authority to do so. Hence only agents that have registered themselves on the ANS should be allowed to unregister themselves. In order to enforce this policy the ANS should authenticate agents that try to unregister from the ANS. This can be done using the simple ANS unregistration protocol given in figure 4. This protocol essentially requires an agent to prove that it is indeed the one that originally registered itself with the ANS and now wishes to unregister and shutdown. The unregistration request will succeed if and only if the following two conditions are met.

1. The agent wishing to unregister has the same name as the name in the registration record on the ANS
2. The agent can authenticate itself and prove that it is indeed the valid holder of that name, i.e. $auth(A_i, w)$ succeeds.

$N_s \rightarrow A_i : m$, where $m = unregister[w]$ and $w = (v, n, A_s, N_i, p, N_s, E_s)$.
$A_i : result = ((N_s == N_s^{'}) \wedge (auth(A_i, w)))$, where $N_s^{'}$ is the name in the registration record.
$A_i :$ Unregister name $N_s$, if $result == True$
$A_i \rightarrow N_s : result.$

**Figure 4: Simple ANS Unregistration Protocol**

## 5.4 Counteracting Threats

The security infrastructure and mechanisms developed in the previous sections can be used to counteract the threats outlined in section 5. While the actual nature of an attack will depend upon the system being protected, the general concepts underlying the solutions will be similar.

In the context of multi agent systems, malicious or malformed agents may disrupt MAS activities by corrupting data records held by an MAS infrastructure component. A typical example of this is the editing or the deleting of registration records from lookup services such as the ANS. This can be prevented by the enforcement of a security policy that ensures that only agents that have registered a particular name may unregister it. The simple ANS unregistration protocol given in figure 4 enforces this policy by authenticating agents prior to unregistration. If an agent $N_s$ possessing identity certificate $w$, wishes to unregister itself, the ANS $A_i$ will process the unregistration request only if $N_s$ can authenticate itself, i.e. $auth(A_i, w)$ returns true. The ANS additionally ensures that the name of the agent generating the unregister request is the same as the name stored in the registration record. The unique naming mechanism provided by the IS will ensure that no two agents get the same name. Therefore an agent should never need to unregister a name that it does not own. Capability description services such as matchmakers can use a similar procedure to ensure that registered capability description documents are not accidently or maliciously edited or removed.

Agents can build and maintain secure communication channels by using standard protocols such as SSL[6]. Higher levels of security can be achieved by generating and exchanging (symmetric) session keys using the asymmetric keys embedded in identity certificates owned by the communication partners. Generating new keys for every dialogue ensures that one compromised session key will not affect the security of all conversations. Additionally, symmetric key encryption is more efficient and can be advantageous when the number or size of the messages is very large.

Replay attacks can be prevented by embedding nonces within messages exchanged by the communicating parties. Every transmitted message is given a newly generated random string that is echoed by the reply. A conversation between two agents thus forms a chain, where every message includes the nonce from the preceeding message and a new nonce to be included in the next message. A replayed message will have a nonce that is from a previous sequence of messages and thus any agent that tries to masquerade as another by resending messages that it has overheard will be foiled when the nonces do not match.

Digital signatures can be used by communicating agents to ensure that messages are not tampered with in any way enroute to the receiver. An agent $a_1$ wanting to send a message $m_1$ to agent $a_2$ can digitally sign the message and send $(m_1, m_1^{'}) = sign(m_1, w)$ to $a_2$. $a_2$ can verify the signature on the message by applying $verify(m_1, m_1^{'}, w)$ hus ensuring that $m_1$ has not been modified by a third party. The digital signature $m_1$ can also be used by $a_2$ to prove that $a_1$ did indeed send the message.

Spoofing attacks can be prevented by rigourous authentication prior to any communication. A good example of a spoofing attack is as follows. Consider an agent $a_i$ with identity certificate $w$ running at physical network address $(h : p)$, where $(h : p)$ denotes the host:port pair of the agent's network address. If $a_i$ disappears without unregistering itself another malicious agent $a_j$ may take over its position at $(h : p)$ and masquerade as $a_i$. An agent $a_3$ wishing to communicate with $a_i$ will actually initiate a connection with $a_j$ but the spoof will be detected when $auth(a_3, w)$ fails.

Denial of Service (DoS) attacks come in a variety of ways and are domain specific. In the simplest DoS attack an agent may overload a service provider by flooding it with queries thus making it unavailable to other agents in the system. This type of attack can be prevented by using policies that ensure that system resources are adequately distributed among the various clients requesting services. An overloaded agent may ignore queries from a DoS attacker if the number of queries received over a certain period of time exceeds some threshold.

# 6. AGENT LIFECYCLES IN AN OPEN MAS

The security mechanisms developed in the previous sections provide services that allow agents to obtain identities, authenticate themselves and digitally sign messages. We now examine certain cases of the lifecycles of an agent and its interactions with the MAS and show the procedures by which an agent may be securely situated in the MAS.

### 6.0.1 Agent Initialization

Every agent $a_i$ that intends to live in the MAS must first obtain an identity for itself. It generates an Identity Request Record $I_i$ consisting of the three fields $N_i$, $C_i$ and an asymmetric key pair $K_s = (E_S, D_s)$. It then sends this information to the IS overseeing its domain and receives an Identity Certificate $w_i$. It uses the simple ANS registration protocol given in figure 3 and registers with any ANS in its domain. It can also register its capabilities with a matchmaker if it needs to advertise its services. It now possesses an identity certificate associating it with a unique identity and it is registered with an ANS and a Matchmaker. In cases where ANS registrations are leased and expired after a certain period of time, an agent merely has to re-register with the ANS using its identity assigned by the IS. It also has the option of changing its identity.

### 6.0.2 Communication with Peers

Any agent $a_i$ that wishes to communicate with another agent $a_j$ must first lookup the address of $a_j$ with the ANS $A_i$ in its domain. Assuming that $a_j$ is either registered with $A_i$ or with another ANS known to $A_i$ the physical address of $a_j$ is returned to $a_i$. Before any dialogue can be established however, $a_i$ must authenticate $a_j$. In order to do this it requests $a_j$'s identity certificate $w_j$. $a_i$ can verify the authenticity and validity of the certificate by applying $verify(w, w^{'}, W)$, where $w^{'}$ is the signature on the certificate and $W$ is the certificate belonging to the IS that issued $w$. $a_i$ can be sure of $a_j$'s identity if the predicate $auth(a_i, w_j)$ succeeds. If the authentication procedure fails then $a_i$ can infer that the ANS registration record for $a_j$ returned from ANS $A_i$ is actually false. In this case it may decide to terminate the connection.

### 6.0.3 Agent Shutdown Procedure

Before an agent $a_i$ disappears from an MAS it must unregister its entries from infrastructure services such as the ANS. In order to do this it follows the simple ANS unregistration protocol given in figure 4. This procedure is for completeness and situations where this is not followed do not lead to any security breaches. Any registrations that exist after the agents have shutdown will point to physical addresses that have no agents running and hence no communication can be established with these points. In the cases where other agents take up these physical locations authentication prior to communication (see subsection 6.0.2 above) will detect the anomaly.

### 6.0.4 Proxy Registrations

There may be cases where an agent wishes to run at multiple physical locations (for example when firewalls prevent access to it from certain networks) and it thus needs to register itself with the ANS and maintain multiple ANS registration records. In this case an agent $a_i$ can use multiple iterations of the simple ANS registration protocol and register itself at locations $(h_1 : p_1), (h_2 : p_2), ..., (h_n : p_n)$ where every $(h_i : p_i)$ represents a physical network host:port address pair. This can allow any agent in the MAS to contact $a_i$ at any of the locations it is registered and running at. The situation when $a_i$ falsely registers itself as running at address $(h_i : p_i)$ is not a security threat since any agent wishing to contact $a_i$ at $(h_i : p_i)$ will find nothing. A legitimate agent running at $(h_i : p_i)$ will not be affected because it will maintain its own ANS registration record that points to its location. Agents wishing to contact this legitimate agent may communicate with it at will. Authentication prior to communication will ensure that agents communicate only with those that they intend to.

### 6.0.5 Multiple Identities

The naming scheme using an IS as an MAS infrastructure component allows agents to obtain and maintain multiple identities with no effect on the security of the MAS. Every agent $a_i$ that requires multiple identities $N_{i_1}, N_{i_2}, ..., N_{i_n}$ is issued identity certificates $w_{i_1}, w_{i_2}, ..., w_{i_n}$. Each certificate binds the agent's identity to a seperate public key thus allowing the agent to maintain multiple ANS registration records. It may choose to run at only one location, in which case each ANS record will point to the same physical address. It may also choose to run at multiple locations and hence any name/address combinations are possible. This functionality may be required when an agent wishes to run under different domains certified by different identity servers.

### 6.0.6 Communication Across Certification Domains

Consider agents $a_1$ and $a_2$ that possess certificates $w_1$ issued by IS $I_1$ and $w_2$ issued by IS $I_2$ respectively. Furthermore let $W_1$ and $W_2$ be the identity certificates belonging to $I_1$ and $I_2$ respectively. $a_1$ and $a_2$ must authenticate each other before any dialogue is initiated and the identity certificates $w_1$ and $w_2$ must be verified by $a_2$ and $a_1$ respectively. $a_1$ and $a_2$ can be sure of the validity of the certificates $w_2$ and $w_1$ if predicates $verify(w_2, w_2^{'}, W_2)$ and $verify(w_1, w_1^{'}, W_1)$ succeed. There is however the additional implicit assumption that the signer certificates $W_1$ and $W_2$ are valid. In order for this assumption to be valid there must exist a common IS lying in the certification chains of $W_1$ and $W_2$. In terms of domains, it is necessary for both agents $a_1$ and $a_2$ to be certified by identity servers that manage subdomains in the the same parent domain. In the case when this does not hold true (i.e. the certification domains are disjoint), the solution lies in one of the agents obtaining two identity certificates (and hence two identies), one for each domain or blindly trusting the IS that certifies the agent being authenticated.

## 7. CONCLUSIONS

As agent systems become bigger and more ubiquitous, agents will achieve more autonomy, slowly becoming transparent to the human users. Agents in such large systems must be independent and must proactively protect themselves from attack from malicious or malformed agents. MAS infrastructure services should include services such as robust agent naming, which can be used to build higher level services for authentication, non-repudiation and secure communication. It is on the basis of these mechanisms of security along with beliefs, intentions, currently unsolved goals and the state of the MAS that agents will be able to spawn high-level cognitive processes that can lead to the development of trust among peers in a Multi Agent Society.

## 8. ACKNOWLEDGMENTS

This research was funded in part by the Air Force Office of Scientific Research grant F30602-98-2-0138 and DARPA grant F49620-01-1-0542 to Carnegie Mellon University.

## 9. REFERENCES


[1] Java Agent Template, Lite (JATLite), http://java.stanford.edu/.

[2] Public Key Infrastructure (PKI), http://www.ietf.org/html.charters/pkix-charter.html.

[3] Secure Hash Algorithm (SHA), http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[4] Security Assertion Markup Language (SAML), http://www.oasis-open.org/committees/security/.

[5] Security Services Markup Language (S2ML), http://www.s2ml.org/.

[6] The SSL Protocol Specification Version 3.0, http://www.netscape.com/eng/ssl3/.

[7] CCITT (Consultative Committee on International Telegraphy and Telephony) Recommendation X.509, The Directory - Authentication Framework. *CCITT Blue Book*, 8:48–81, 1988.

[8] K. Decker, K. Sycara, and M. Williamson. Middle Agents for the Internet. In *International Joint Conference on Artificial Intelligence (IJCAI)*, January 1997.

[9] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[10] C. C. Eckel and R. K. Wilson. The Human Face of Game Theory: Trust and Reciprocity in Sequential Games. *Trust and Reciprocity: Interdisciplinary Conceptual and Empirical Lessons*, 2000.

[11] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.

[12] L. Foner. A Multi Agent Referral System for Matchmaking. In *First International Conference on the Practical Applications of Intelligent Agents and Multi Agent Technology*, April 1996.

[13] L. Foner. A Security Architecture for Multi Agent Matchmaking. In *First International Conference on Multi-Agent Systems (ICMAS '96)*, 1996.

[14] L. Foner. Yenta: A Multi-Agent Referral Based Matchmaking System. In *The First International Conference on Autonomous Agents*, February 1997.

[15] W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, and J. Lapp. XML Key Management Specification (XKMS), http://www.w3.org/tr/xkms/.

[16] E. Gerck. Overview of Certification Systems: X.509, CA, PGP and SKIP, http://www.mcg.org.br/certover.pdf.

[17] P. J. Gmytrasiewicz and E. H. Durfee. Toward a Theory of Honesty and Trust Among Communicating Autonomous Agents. *Group Decision and Negotiation"*, 2(3):237–258, 1993.

[18] Q. He, K. Sycara, and T. W. Finin. Personal security agent: Kqml-based pki. In *ACM Conference on Autonomous Agents*, 1998.

[19] N. B. of Standards (U.S.). Data Encryption Standard (DES). *Federal Information Processing Standards Publication 46*, April 1977.

[20] M. Rabi, T. Finin, A. Sherman, and Y. Labrou. Secure Knowledge Query Manipulation Language: A Security Infrastructure for Agent Communication Languages, citeseer.nj.nec.com/196978.html.

[21] R. Rivest, A. Shamir, and L. Adelman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[22] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record (ACM Special Interests Group on Management of Data)*, 28(1):47–53, March 1999.

[23] K. Sycara, J. Lu, M. Klusch, and S. Widoff. Matchmaking among Heterogeneous Agents on the Internet. In *Proceedings of the 1999 AAAI Spring Symposium on Intelligent Agents in Cyberspace*, March 1999.

[24] K. Sycara, M. Paolucci, M. V. Velsen, and J. A. Giampapa. The RETSINA MAS Infrastructure. Technical Report CMU-RI-TR-01-05, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 2001.

[25] Y. Teng, V. V. Phoba, and B. Choi. Design of Trust Metrics Based on Dempster-Shafer Theory, citeseer.nj.nec.com/461538.html.

[26] C. Thirunavukkarasu, T. Finin, and J. Mayfield. Secret Agents – A Security Architecture for the KQML Agent Communication Language. In *Intelligent Information Agents Workshop* held in conjunction with *Fourth International Conference on Information and Knowledge Management CIKM'95*, Baltimore, 1995.

[27] H. C. Wong and K. Sycara. Adding Security and Trust to Multi Agent Systems. In *Proceedings of Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies*, pages 149 – 161, May 1999.

[28] H. C. Wong and K. Sycara. A Taxonomy of Middle-Agents for the Internet. In *Proceedings of the Fourth International Conference on Multi Agent Systems*, pages 465 – 466, July 2000.