Review of Multi Agent Base Security in Distributed Systems

Roodabe Heydari¹, Mehdi Effatparvar²

¹1. Department of Computer Engineering, Germi branch, Islamic Azad University, Germi, Iran
²2. ECE Department, Ardabil Branch, Islamic Azad University, Ardabil, Iran

Corresponding Author email: Roodabeh.Heidari@gmail.com

ABSTRACT: The multi-agent-systems paradigm has grown more and more popular to be a cause for realizing net-based solutions. This development is combined with an escalating relevance of security issues. In particular, the actual possibility diminished privacy along with assets is a major concern for both merchants and customers, in Internet-based commerce and without getting properly addressed, such very legitimate concerns hamper the continuing development of e-commerce. This paper discussion focuses of what techniques these agent systems purchase to eliminate several of the discussed security requirements.


INTRODUCTION

Designing large-scale distributed multi-agent systems that are employed in open environments, such as the Internet, creates new challenges, especially concerning security issues. Agents are autonomous, proactive, communicative, goal-directed, often capable of learning, and often mobile (Braun and Rossak, 2004). Mobile agents traverse the network to get into services and resources they should have the goals they pursue. Possibly mobile agent technology in sectors for example E-Commerce (AOS, 2005), E-Health (Moreno and Nealon, 2003) and E-Governance (Csetenyi, 2000) is well recognized. Of these sectors, security issues for example authentication, authorization, privacy, and copyright are the most importance. Data access control is mandatory: by moving agents to the spot from which data is stored, data access and processing can be achieved locally and controlled. Many security requirements ought to be addressed for large-scale distributed multivalent systems in open environments. The main objective in this chapter will lie upon security requirements specific for agent systems as an alternative to security requirements for distributed computer systems in general. Section 2 identifies the most recent security requirements for agent systems. This number of requirements is a baseline to be fulfilled for secure agent systems in open environments. Sections 3 through 7 discuss the security requirements and possible solutions in detail. The solutions are illustrated in the context on the Agentscape (AgentScape, 1998) agent platform. This platform has become chosen as it has been specifically created to be inside a large-scale, distributed, open environment. However, similar implementations of them solutions are possible in other agent platforms. The chapter closes with a breakdown of many well-known agent platforms, for example AgentScape (AgentScape, 1998), Ajanta (Karnik and Tripathi, 1998), SeMoA (Roth and Jalali-Sohi, 2001), and JADE (Belliﬁmene, 1999) featuring its security extensions JADE-S (Tomaiulo, 2003) and S-Agent (Gunupudi, 2006, Warnier, 2006). The discussion focuses of what techniques these agent systems purchased in order to resolve many of the discussed security requirements. The entire content of paper has become organized as follow: 2 Security in Multi-Agent Systems, 3 Naming and Authentication, 4 Secure MAS network management, 5 Secure MAS military application, 6 Secure MAS e-health, 7 Malicious Agents, 8 Discovered Multi-agent security, and 9 Conclusion.

Security in Multi-Agent Systems

only security requirements associated with distributed personal computer, but also multiagentsystem specific security requirements. It identifies the absolute minimum group of security requirements specific to multi-agent systems that should be fulfilled for doing itto perform securely in an open environment.

Principals in an Multi-Agent System

Conceptually, multi-agent systems are distributed, networked, computer systems through which agent owners run communicating agents that access resources on hosts, as both versions runs a realtor platform (i.e., in a situation of a realtor middleware) that is certainly in the management of a platform administrator. The
bold terms are the foremost principals in a multi-agent system (Gunupudi, 2006). All these principals face security threats. A secure multi-agent system must protect these principals along with communication to principals against security threats. Such as, secure communication is needed to protect the communication between two agents, but will also between two platforms and between a realtor as well as a platform or maybe the agent's owner. In a distributed system, like the Internet, communication is often over long distances and might be intercepted or monitored. In a closed environment, all principals in a realtor system are known earlier most likely trusted, therefore, security measures are often implicit. However, in an increasingly open environment, more explicit security measures are necessary to guard against security threats (Jennings, 2003; AgentScape, 1998; Karnik and Tripathi, 1998; Moreno and Nealon, 2003).

Security Threats for the Agent's Owner

A real estate agent performs its actions with respect to its owner, which is generally a legal entity, maybe a human or an institution: the agent owner. The principle security concerns for an agent owner are confidentiality and integrity of his agent, any data it carries, and any communication both to and from the agent (Guttman, 2001). Confidentiality is proportional to guarding the privacy of any agent’s owner. As an example, in the e-health environment, agents acting with respect to patients carry privacy-sensitive information that really should not be revealed to others. Accessibility to a realtor is a requisite for a realtor owner which has been implemented by a realtor owner himself, possibly supported by a platform. As an example, to bring about a realtor more fault tolerant, a realtor owner can start two (or more) copies of a realtor and send them to platforms, so that if one agent dies, additional can continue, keeping the agent offered to its owner. Alternatively, a realtor owner can trust a platform owner to try adequate measures to ensure the particular of an agent platform (Karnik and Tripath, 1998, Moreno and Nealon, 2003, Paterson, 2010, Gunupudi, 2006).

Security Threats for the Platform’s Administrator

A host’s administrator can run a realtor platform (i.e., an instance of the agent middleware) on his host. A real estate agent platform enables visiting agents to (paid) use of a host’s resources. The main security concerns for a realtor platform’s administrator (who is certainly not just like the host's administrator) are confidentiality, integrity, and availability of the agent platform, its resources, and any communication from and to the agent platform. A typical resource in a realtor system that may be the mark of availability threat is the lookup service (Guttman, 2001, Poggi, 2003, Tweedalea, 2007). The lookup service is a database that monitors the current locations of most agents in a realtor system. A real estate agent system needs these details, as an example, to deliver messages to agents sent from other agents. In an open environment, an attacker could start a realtor platform, join the agent community and subsequently fill the lookup service with false information regarding locations of agents. This attack renders the information in a lookup service useless and consequently paralyzes a realtor system as a whole. This type of attack is a questionnaire of a Denial-of-Service attack and illustrates the necessity of a safe lookup service which guarantees the correctness of its information. Without it, a platform administrator cannot guarantee the availability of the agent platform (Paterson, 2010).

Naming and Authentication

As mentioned above, identity management is an important security requirement in an open, distributed multi-agent system. The ability to name principals and authenticate them is an important part of identity management.

Naming

Before authentication is quite possible, principals in the multi-agent system must first have a very good (unique) identifier: a name. This name does not have to be human-readable; it’s really a meaningless string, given it is actually machine-readable. In theory, names is static, meaning they can’t shift the lifetime of a principal, or dynamic. For humans and organizations static names are an added logical choice, especially (mobile) agents in an agent system, dynamic names have their use. Just like, agent names could include a mention of the venue where an agent resides (location-dependent names), that produces locating the agent trivial. However, for most of this chapter it is actually assumed that principals have globally unique identifiers (GUIDs), that happen to be static names. The phrase global is not going to necessarily need mean the identifier is exclusive around the universe nevertheless it really suffices that the identifier is exclusive within an instance of any running agent system. It can also be assumed that in almost any agent system, something similar to GUIDs is required to principals. Another property of naming is actually principals might have dozens name. Just like, if an agent has multiple names, it can certainly utilizing names as pseudonyms. Pseudonyms can be used to implement anonymity (Warnier and Brazier, 2007): an agent can employ a different pseudonym each interaction with another agent. To illustrate, AgentScape (Nguyen, 2001) (see Section 8) actually has two naming schemes. First, agents are identified internally by GUIDs, that happen to be kept private in to the middleware.
Second, agents are externally visible through their (static) handles. Each agent might have dozens handle at any given time, which permits them to implement a style of anonymity as each handle is a pseudonym. Note that naming is simply not sufficient for authentication as there isn’t an mechanism to verify that a name corresponds to the correct principal(Hbner,2007, Bellifemine,2008).

**Linking an Agent and its Owner**

Generally in most situations, a representative must be uniquely and undeniably associated with its owner (e.g., an individual or organization). This link is part of authenticating a representative and is a good idea, like, to charge the owner if agents buy things on-line or help determine liability whenever agents misbehave. This section discusses, while agent based systems, how agents could be 'bound' recommended to their owner. As mentioned before, it will be assumed that a representative could be identified by way of a GUID. Conceptually, a representative comprises of meta-data, (executable) code, and data that a representative has ‘found’ on a specific host. The meta-data of a representative contains at the very least the following: the GUID of this agent, the url of this agent's owner, precluding a signed (by the owner) hash inthis agent’s code. The signature suggests that agent and owner are bound to each other. For authentication to ensure, it can be crucial the public key of a representative owner is stored during a PKI. By way of example, in Agents cape, when a representative is injected, the agent platform checks should the agent code is indeed signed. If verification is a winner the agent obtains a GUID precluding a handle is returned towards agent owner. Assuming the owner keeps this handle secret, to generate videos to communicate between agent and owner. Next, the injected representative is started through agent platform. In the event the agent misbehaves in some manner, the owner could be contacted and stay held responsible for ones agent’s actions. The agent injection procedure is similar in other agent systems(Ricci and Denit,2004; Ismal,2008; Novak,2003).

**Secure MAS networkmanagement**

These days, security is an integral part of network management services but for its long history we see a huge diversity in the protection aspects of network management. The IETF (Internet Engineering Task Force (IETF), online) has published some standards and RFCs based on securing the network and information. However these standards commonly are not MAS based hence because of the not enough capabilities cannot adopt offer secure MAS or can perform but at the expense of considerable extra complexities. Furthermore, the fabric and infrastructure of recent network applications inherently match secure MAS causethem to become suited to the integration of network management with distributed security models. As a way to facilitate the development of robust and highly adaptable communication systems, NEC has recently released a unit it really is a mixture off a ‘distributed agent-based system’ plus a ‘reconfigurable peer-to-peer overlay network’(Vaughan,2008). Onthe listof the challenges of the new development usually this system is tested against particular scenarios whereby earth is dynamic and changing fast with adverse conditions as those to military systems. In such environment, it’s hard to ensure the protection of communications. Similarly, for a properly competitive business environment,(Das,2008) prototype a built-in data center power management solution according to MAS with server management tools, sensors and monitors, besides an agent-based approach to get certain specified power and gratification objectives. The effects show that by cleverly turning off a lot of the servers under low-load conditions, they might claim over 25% power saving throughout the unmanaged case where none are equipped with incurring SLA penalties for typical daily and weekly periodic demands seen by way of the Web server farms. In such systems, security was planned for as misuse agents which could cause denial-of-service attacks to services made available from the information centers. Another network management distributed security development from Wang et al.(Wang,2008) where an ‘immune MAS’ is used to grant network intrusion detection. Their experimental results exhibit that the system not merely reduces effectively the rates of false-negative and false positive but will also it will adapt itself for a continuously changing network environment. Similarly,( Holloway,2009) propose a good and a lot more efficient self-organised entangled hierarchical architecture made up of multiple agents that decentralize the network security controller. Usually, it is used being a evolutionary approach according to swarms behavior for which a desired network security is defined, formalized and enforced by way of the technique of collaborative interactions with other agents(Tweedalea,2007, Moradian,2006).

**Secure MAS military application**

The intrinsic nature of the military application scenarios requires the systems to deal with security as a top priority factor. In effect, secure MAS can provide key security features and services in such scenarios. (Beautemen,2006) discuss detailed key factors that need to be taken into account when provisioning applications, tools, devices and infrastructure for military domain in the context of autonomous agents and MAS. A military application scenario of secure MAS is the use of critical decision making support systems in hostile environments to help in military tactics and military combat actions in the battle field. (Cila and
Malab,2010) provide a secure MAS architecture which matches the needs of future multi-dimensional warfare. This is a twolayer multi-agent architecture in which the first layer contains mission analysis agents, mission time scheduling agents, enemy situation analysing agents, own situation analysing agents, logistic agents and action generating agents. The agents need databases for the intelligence, environment, terrain, enemy tactics, techniques and procedures, own tactics, techniques and procedures and logistics information. Fig. 6 depicts the architecture for this military scenario in which all the databases and agents are interconnected by means of the security services provided by the secure MAS. Owing to the nature of these systems, the second layer of this architecture is a simulator tool in which all these agents can be intensively verified and validated before getting the system ready-to-use in the real battle field(Cranefields,2003,Papazoglou,2008, Bergenti,2009).

Secure MAS e-health

Healthcare applications typically require loosely coupled heterogeneous components, a dynamic and distributed management of data and remote collaboration among diverse entities and users. The agent approach has shown a potential to be deployed in a wide range of applications in health care services. Agents can maintain the autonomy of the collaborating participants, integrate disparate operating environments, coordinate distributed data, such as patient records held in different departments within a hospital or in several hospitals, clinics and surgeries (Aldea,2001), improve patient management through distributed patient scheduling using some co-operating intelligent agents, provide remote care monitoring and information for groups such as the elderly and chronically ill, undertake hospital patient monitoring, supply diagnosis decision-support and enable intelligent human computer interfaces to adapt to medical data upon users requirements. The e-health systems normally require handling of a very sensitive large amount of medical data. These include patient’s medical history, diagnosis, test results and various personal details. Therefore all usual properties such as classic confidentiality, authentication, integrity and nonrepudiation should also be guaranteed in any agent-based health-care system. The use of cryptographic methods is also important to protect the access to data while it is being transmitted among agents,( Li and Hoang,2009)propose a role-based access control scheme to protect confidential e-health data from unauthorised access. The proposed scheme is dynamic in terms of the interactivness between three main elements defined as: role, interaction and organisation. The e-health system is perceived as MAS with different personnel playing different roles, requiring varying interactions among themselves, under different organisational contexts. The role is defined as a peer-to-peer model, capable of receiving requests from other roles as well as initiating requests to other roles of the system. A role that initiates a request to another role is defined as an initiator role. The role that receives a request is called a ‘reactor’. Each role is associated with a security property or a dependency, which defines the constraints to ensure successful satisfaction of the dependency(Bresciani,2003, Aldea, 2001, Hoang,2009).

Malicious Agents

This section focuses on the complementary problem: malicious agents. Just as agent owners want to protect their agents against potentially malicious hosts,so do platform administrators want to protect their hosts against potentially malicious migrating agents. Malicious agents typically attempt to gain access to resources on a host they are not authorized to use. Such access includes attempts to access private data of the host, private data of other agents,or to use additional computational resources that have not been negotiated. Fortunately, there are a number of techniques that a platform administrator can apply to reduce the threat of malicious agents and control their access to a host’s resources. This section discusses a few of these techniques and subsequently focuses on the subject on how to configure and manage access to resources for agents(Hbnerj,2007,Cranefield,2003,Tweedalea,2007).

Sandboxing Agents

Most solutions to securing hosts from malicious agents entail monitoring every action that a dealer attempts even on a host. Whenever a dealer is really a call to the middleware API, it’s intercepted by a security manager. The security manager checks the device policy to check which offer which action which includes migration and resource access, should really be allowed or denied. Like, a bunch could decide that it does not allow agents make use of remote web-services (i.e., not running on your regional host). Every make an effort to contact a remote web-service will undoubtedly be blocked by the security manager. Many agent platforms are Java-based (Paterson,2010),whereas in the Java the primary solutions towards securing mobile code will be to execute any remote code in the protection domain or sandbox. A sandbox limits the set of operations how the remote code may call. Like, sandboxing typically restricts network access as well as authority to access your regional file system. Java provides agent system programmers the know how to define sandboxes with a security manager and/or custom class loaders. In Java your sandbox is enforced and implemented through the underlying JVM, for interpreted scripting languages which includes Python and Safe-Tcl the sandbox is implemented through the interpreter. For C or C++ (binary code) agents are ‘jailed’(SHI,2008). Sandboxing and
jailing are supplied solutions with which agents are run in contained environments limiting the amount of damage they are induce to the systems on that they run. An alternate is to simply run agents of trusted owners. Whom to trust depends on system administrator. Through this solution, agents are simply trusted cons signed by a professional software manufacturer, whom a computer owner trusts not to provide malicious agents. The simplicity with this scheme is additionally its weakness: the security on the system lies in the fact that the signer is trustworthy. The weakness with this system was already shown as digital signing certificates have been completely issued to people masquerading to be an associated with a common software maker (Dinverno, 2001). Furthermore, smaller than average open source software makers might not have the financial capability to purchase such signing certificates.

 Needless to say, digital signatures is usually in addition to sandboxing to develop a better quality security solution. Finally, a elaborate security approach is the use of proof-carrying code (Szypzczyk, 2009) (applied to the mobile agent paradigm described in (Tweedalea, 2007)). Agents carry a machine verifiable proof together that specifies their expected and acceptable behavior. Each host gives you a theorem proverb make sure that an agent's code indeed adheres to its specification. Specifying security permissions can be an elaborate job, more likely to mistakes. Theremainder of this section discusses how lots of people of roles and sets of predefinedpolicies simplify this task. Security policies allow users of agent systems to take care of the security popular features of the multi-agent system within their choice. Developers of agent systemsbe able to ship a number of security policies withregards to their software. Like, an effective default policy is certainlyone that won't prevent users from performing vital tasks, but will protect the host against the most common security issues. In contrast, 'high security' policies should reallybe utilized in security critical environments. Such policies are incredibly restrictive. Below a security policy framework is discussed and illustrated within AgentScape (Ismail, 2008, Bergenti, 2009).

**Security Manager**

To enforce resource access control, every action of a real estate agent must first be authorized by an RBAC system of the action can be executed. Whenever a real estate agent attempts to perform a security relevant action, a Security Manager checks if the agent is authorized to perform this action. This check is a two-step process. First, the Security Manager determines the GUID of the agent and determines the role, or roles, of which the GUID is a member. Second, the Security Manager determines if one or more of these roles is authorized to perform the requested action. It is worthwhile to see that not only a platform's administrator, but in addition a real estate agent owner needs to trust the security manager. After a real estate agent owner has negotiated for resources and possibly paid for access, a real estate agent owner expects the security manager to grant access as negotiated. Similar to monitoring of Service Level Agreements (SLA) a reliable third party module can be utilized to monitor and log the communication between client (agent) and supplier (host) (Papazoglou, 2008, Bresciani, 2003).

**Security Policies**

While security (Vaughan, 2008, Wang, 2008) can be a major concern for resource and location administrators, it is not always the case that these principals are either particularly interested, or trained to define their own security policies. For this reason, it is advisable to have a set of predefined default policies. These predefined policies range from simple, non-restrictive policies, used for agent systems deployed in a well-known environment, to stronger, restrictive policies, where agent systems operate in a more hostile environment. These two extremes are described in the context of the following two case studies: a closed world and a hostile world. In a simple closed world environment, locations are controlled by well-known entities and are all trusted. Communication between locations is cryptographically secured and each location is known and trusted by every other location. The major threat to the middleware is that of malicious agents. Agent owners must be authenticated. Once authenticated, agents are authorized to perform any and all actions. Therefore, the authorization mechanism is not used for access control, but is instead used for auditing purposes: whenever an agent performs a security relevant action, it is logged for possible later examination by the location administrator. While a simple system is common in small, closed environments, the provision of services on the web, with the associated access of these services by software agents demonstrates that such an environment cannot be assumed. In a hostile environment locations are controlled by entities that are not always known by every principal. Agents are authenticated by their initial location as before, but the authorization mechanism is now used to enforce location-specific restrictions. The security manager monitors usage of specified resources and ensures that all accesses are restricted by the negotiated limits. Any breaches of these limits are logged and execution of the agent responsible is immediately suspended. Migration is only authorized between the original 'home' host—the host where the agent started—and remote hosts. Therefore, migration from one remote host to another forces an agent to first return to the home host. This is enforced to prevent malicious hosts attempting to inject or read data developed from a prior migration. For example, the result of a price check from
a prior website should not be available when performing a price check at a competitor's site. Within a hostile environment, not only locations and hosts may want to constrain the actions of agents, but also agent owners may want to restrict the actions their agents are allowed to perform on their behalf. These actions include the ability to negotiate, migrate, inject, access resources, purchase items on the web, etc. In summary, the security architecture outlined in this section and illustrated within the AgentScape agent system provides a flexible means to define and manage agent access to specific functionality. Flexibility is provided in two areas: firstly, hosts and locations have the ability to control access to resources that they control. Secondly, owners can constrain their agents from performing actions that, while they are authorized by the locations and hosts, are not desirable to the owner. For more information see (Moradian, 2006).

**Discovered Multi-agent security**

utilization of a multi-agent Warnier, 2007; AOS, 2005; Dinnervanom, 2001) style platform simplifies implementation of the MAS. Upon today’s software techniques this is provided by way of a middleware to facilitate deploying the agents, controlling communication between the agents and maintaining the mandatory security measures for the agents. Owing with their location in the system agents can conveniently handle most of security functions as well as the background processes within the MAS. Let us believe that the security system is deployed to handle a distributed intrusion detection system. Each agent is responsible for analysing the network behaviour regarding its own machine while sharing the info on the platform with one another so that they may collaborate in detecting a possible intrusion activity. This secures the platform from any possible malicious agent attack. A practical method to make this happen is to measure security through having a sandbox, defined as a protected environment enabling genuine agents to protect the platform against malicious agents. The next level of protection is for platform-to-agent security which is as security service that the platform makes for the agents. Essentially, analogy of this really is some sort of privacy computation such as encryption that protects the info connected with a realtor against any possible manipulation such as workflow, privacy and integrity of data. A next level of protection is for agent-to-agent security in which a realtor tries to protect itself against actions made by other agents’ misbehaviour such as denial of service, spying or pretentious actions with respect to other agents. Finally, platform-to-platform security protection is devised for securing interactions between different platforms. In spite its natural capability, MAS, however, has been effectively used previously to produce applications fully secure. Like, experts such as Nguyen et al. (Fernandes, 2006), Poslad et al. (Karnik, 1998), Mana et al. (Moreno, 2003) and (Garrigues, 2010), clearly point out that not enough proper security is in most MAS applications. One is Aglet (Aglet project page is available at http://aglets.sourceforge.net/ and the most recent version premiered in 2002), the well-known IBM's platform, which neither provides a protection for protecting the agents nor devises any strong authentication and authorization. It offers a basic user-password authentication process for the platform to spot the agents, which in turn, enables them to get into the platform anonymously together with an easy access control based on the two possible roles ‘trusted agents’ for agents developed by the server and ‘untrusted agents’ for agents developed by the external servers. Another example is JACK (AOS, 2005), a commercial MAP for building the distributed multi-agent reasoning system (Gunupudi, 2006). This platform doesn’t handle any security services and fully depends on Java’s internal security. It uses Java security policies for file access control. (TuCSoN olRici, 2004, Gunupudi, 2006) is a free open-source MAP and S-Moises’ of Hbner et al. (Israil, 2008), a middleware that can be used to create structured MAS and Agent Service, by Vecchiola et al. (Fernandes, 2006), a totally free open source framework for developing MAS, which all lack the provision of strong security measures. They even do not facilitate an effective protection for the agents within the platform and lack strong protection for the platform against malicious agents. Some do not provide any authentication and some lack security for the communications. Generally, a protected MAP is required to handle basic security measures such as authentication, authorization and accounting services plus the usual privacy and integrity of data through the communications. Further requirements for these systems are inclusion of new MAS-based measures such as trust and vulnerability factors. An authentication, for example, located within the MAP can help to protect the infrastructure. A protected infrastructure then enables a protected system as well as protecting agents against malicious processes injected by fictitious agents. Moreover, this method can be used as a foundation for authorisation and trust management system. A dependable authorisation and trust management system can enable managing the agents’ privileges and control their behaviour as well as maintaining their privacy. For providing a protected MAP, JADE (Bellifemine, 2008), well-known development
framework fora realtor in Java, then extended into JADE-S(Guttman, 2001) now comes with an increase of recent developments of SAgents(Jennings, 2003) and ExJADE-S(Vitabile, 2009) enhancing the security facets of the platforms. All of these extensions incorporate strong authentication systems adopting certificate management, distributed access control mechanism and an accounting system. Additionally, they require keeping a course on agents’ behaviour while maintaining data integrity, trust management, encryption and confidentiality of the communications. (Ismail, 2008) proposes a method to tackle address security issues of a protected MAP directly. In this technique the platform implements a clear authentication integrated within the authorisation service. The systems supports a vibrant exchange beneath the evolutionary access rights and claims modularity, portability, non-repudiation and high performance. It runs on the digital signature for authentication of the agents and a brand new algorithm to regulate the exchange of access rights on the list of agents. (Fernandez, 2006) combine the peer-to-peer networking features to construct a protected MAP. This platform provides a strong authentication service based on X.509 certification enhanced with authorisation, secure transport and secure execution for the agents. The platform consists of an overlay network through a distributed hash map, called Pastry network, used to distribute the security services across the network. Owing to the nature of the overlay network this platform can also provide a fault-tolerant environment for all agents, a desirable feature in a protected critical system. In order to secure the communication in existent insecure MAPs, the X-System (Novak, 2003) provides different security architecture. Basically, the secure communication and authentication is through a public key infrastructure based on the extension of the FIPA (Gunupudi, 2006) protocol. This well-known inter-operation protocol shares messages between the agents and the MAPs thus, providing FIPA compliant security architecture for the MAP.

**CONCLUSION**

Security in multi-agent systems is a major concern, particularly in multi-agent systems deployed in a large-scale, distributed, and open environment. Finding a balance between restricting access to resources and allowing enough openness to let the whole system function efficiently and effectively is the challenge. Based on our review we have discovered one closely interrelated trends for developing new solutions that is secure MAS, This solution is better than preview solutions.

**REFERENCES**


Auton J. Agents Multi- Agent Syst. 2004. 9, (1–2), pp. 5–53


