DEVELOPING A SECURITY MECHANISM FOR SOFTWARE AGENTS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Fatih TEKBACAK

July 2006
İZMİR

We approve the thesis of **Fatih TEKBACAK**

Date of Signature

. . . . **10 July 2006**
Assist. Prof. Dr. Tuğkan TUĞLULAR
Supervisor
Department of Computer Engineering
İzmir Institute of Technology

. . . . **10 July 2006**
Prof. Dr. Oğuz DİKENELLİ
Department of Computer Engineering
Ege University

. . . . **10 July 2006**
Assist. Prof. Dr. Bora KUMOVA
Department of Computer Engineering
İzmir Institute of Technology

. . . . **10 July 2006**
Prof. Dr. Kayhan ERCİYEŞ
Head of Department
İzmir Institute of Technology

. . . .
Assoc. Prof. Dr. Semahat ÖZDEMİR
Head of the Graduate School

ACKNOWLEDGEMENTS

I would like to express my thankfulness to my advisor, Asst. Prof. Dr. Tuğkan Tuğlular, for his guidance and encouragement. He was the one who supported me when I was in trouble with critical decisions. His valuable support, and confidence have been the driving force of this thesis work.

Furthermore, I had the pleasure of working with Önder Gürcan who cooperated me in many studies. I should also thank to Hüseyin Hışıl and Orhan Dağdeviren for their help in writing with LaTeX2e.

Finally, I should thank to my parents who always supported me throughout my education as well as in my graduate study.

ABSTRACT

DEVELOPING A SECURITY MECHANISM FOR SOFTWARE AGENTS

This thesis proposes a message security solution on multi-agent systems. A general security analysis based on properties of software agents is presented along with an overview of security measures applicable to multi-agent systems. A security design and implementation has been developed to protect communication among agents. And this implementation scheme has been applied to Seagent, a semantic web enabled multi-agent framework. Hence, a set of agent security mechanisms have been adapted for Seagent and have been implemented for message confidentiality, integrity, authentication and non-repudiation. Then these mechanisms have been tested for communication performance on Seagent.

ÖZET

YAZILIM ETMENLERİ İÇİN BİR GÜVENLİK MEKANİZMASI OLUŞTURULMASI

Bu tez çoklu etmen sistemleri üzerinde bir mesaj güvenliği çözümü önermektedir. Yazılım etmenlerinin özelliklerini temel alarak oluşturulan genel bir güvenlik analizi, çoklu etmen sistemlerine uygulanabilen güvenlik ölçütlerinin gözden geçirilmesi ile anlatılmıştır. Etmenler arasındaki iletişimdeki korumayı sağlamak için bir güvenlik tasarım ve implementasyonu geliştirilmiştir. Bu implementasyon mekanizması bir anlamsal web tabanlı çoklu etmen çatısı olan Seagent üzerinde uygulanmıştır. Bu nedenle Seagent için birçok etmen güvenlik mekanizması uyarlanmış ve mesaj gizliliği, bütünlüğü, doğruluğun kanıtlanması ve reddedilememe mekanizmaları implemente edilmiştir. Daha sonra bu mekanizmalar Seagent üzerindeki iletişim performansı için test edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This thesis is a combined study of message security solutions and multi-agent systems. By the progress of multi-agent systems, usage of these systems for different organizations have increased. Different organizational multi-agent platforms services in various domains have different requirements. One of these requirements as message security has been a conventional security problem where intruder agents not to read, modify, insert or delete messages. So, these problems exist in a semantic-web enabled multi-agent system, Seagent. Introduction and subsequent chapters propose a solution for security related operations on Seagent.

1.1. Background

Security risks exist throughout the agent life-cycle. These risks are present during agent management, registration, execution, agent-to-agent communication, user-agent interaction, and agent mobility.

In recent years, the Foundation for Intelligent Physical Agents (FIPA), an organization pursuing the promotion of technologies and interoperability specifications that facilitate the end-to-end interworking of intelligent agent systems in modern commercial and industrial settings, has become an influencial part of the agent community. The FIPA 98 Agent Security Management specification "(FIPA 1998)" outlined the requirements for secure intra- and inter-platform communication. FIPA 98 Security Management addressed mutual agent security issues for agent-to-agent interaction. This specification does not mandate the use of security features. Instead, it mandates how agents and agent platforms may interoperate in a secure fashion, if security is desired.

This specification allows security to be implemented at the message transport layer, through the use of security services available from a shared transport protocol at the agent platform. At the agent level, this specification relies on an asynchronous messaging model of communications. All information regarding the protection mechanisms employed to encapsulate a given message is provided with the message.

In 2003, FIPA Agent Message Security Object Proposal "(Vitaglione et al. 2003)" have been drafted. This proposal introduces the concept of per message security which means that each individual Agent Communication Language(ACL) message "(FIPA 2002b)" contains the security information required to process the embedded security safeguards. Agents are intended to process the security mechanisms themselves where appropriate so as to provided end-to-end (or peer-to-peer) security. With this approach specified FIPA Security Object "(Vitaglione et al. 2003, p.4.)" has been introduced to be used for encryption and signature scenarios "(Vitaglione et al. 2003, pp.5-8.)".

1.2. Problem Statement

With the principles for secure message transfer and expansion of security into the area of mobile agents, the multi-agent systems(MAS) security design considerations have been mentioned. Necessary security services are defined in order to provide entities within a MAS to assure confidence in the distributed system environment and to prevent it from the security attacks. By thinking about the FIPA compliant architectures, the problem is specialized as the security of communication between agent platform mediators such as Agent Management System(AMS), Directory Facilitator(DF) and the other agents in the platform.

The manner in which one or more security safeguards may be represented, associated with security threats, advertised, agreed and invoked, has not been exactly standardized within the agent community. These must be agreed in order to achieve end-to-end security. Agent security has been represented at a number of different layers in the ACL communication stack ranging from the use of secure transport protocols, new transport envelope fields, content ontologies, new security ACL fields etc.

According to "(Vitaglione et al. 2003)" as shown in Section 1.1., the unstandardized message security problem is tried to be solved. The problems that are not shown in "(FIPA 1998)" as public key infrastructures(PKI) and implementation methodology are explained in detail. Security design solutions take advantage of Security Object to be able to identify the message security features and protect from insecure communication.

The main security requirements come from the specific properties of agents and agent systems, such as the open and distributed characteristics of agent-based applications, the inherent social interactivity of agents, and the fact that agents may act autonomously while at the same time representing some user. However, Seagent security approach focuses on security mechanisms to implement security requirements according to agents' above characteristics. While Seagent security mechanisms are designed and implemented, it is interested in the following areas as "(FIPA 2001)":

1. Security aspects of FIPA-ACL messages, including both abstract messages and their concrete encodings: The FIPA ACL message payload has to be securely communicated between communicating agents. For example, the message with its specified encoding has to be encrypted. However, related parameters have to be added to the message envelope to understand the operations processed on payload.

2. Security-related conversational interactions between agents: When the conversation started, first agent must send its public key. Then the other agent uses this information to realize a secure communication.

3. Abstract and concrete issues of integrity and authentication related to agents: When an agent sends its message, the other agent have to know that the message is correct and this message comes from the expected sender.

By taking into consideration these information, this thesis is especially interested in new transport envelope fields and new security ACL fields to be able to solve securing ACL messages. Because unsecure ACL messages reduce trust between agents and no standardized implementation is developed for communication security of agents. Hence, a secure and simple way for messaging is wanted to be designed for Seagent thinking about the performance challenges for agent message creation and communication.

1.3. Solution Approach

The considered main scenario is the transmission of a FIPA message between two agent platforms. It is wanted to focus on basic techniques, applied on the transmission of a single message, that allow several different higher levels to provide secure and trust functionalities.

The main requirements focused are "(Greenwood et al. 2003)":

1. Message confidentiality

2. Message integrity

3. Data origin authentication

Such requirements can be satisfied by using many different cryptographic techniques and protocols. The goal here is limited in defining basic mechanisms for including all the required security information into a the FIPA transport message, so that the recipients can process (verify, decrypt, etc..) the message properly.

The data formats defined must be independent by the security infrastructure/architecture adopted (e.g. PKI etc..), must be highly flexible in order to adopt most common cryptographic algorithms, and must be highly extensible in order to allow plugging of different formats for the transmitted security information.

Although they should not be mandatory, the existence of concepts like symmetric encryption (with secret keys), asymmetric encryption (private and public keys) and hashing algorithm to compute message digest are defined as the main security operations of the system.

In Seagent platform, message sending and receiving is managed by Dispatcher which is able to wait for incoming messages and send outgoing messages. It uses incoming message queue for incoming messages and outgoing message queue for outgoing messages. The incoming messages of Dispatcher come from an internal class after a set of operations of the agent platform. So incoming message security for Dispatcher is excluded. But the outgoing queue includes messages that are sent to the agents. However, the transported message to the agent has to be processed by Dispatcher according to security issues and these security issues have been reversely processed by agent that takes message.

Implementing above security issues as different security mechanisms and wrapping these mechanisms in a security service layer has been the main methodology requirement in Seagent platform. Message-based FIPA-compliant security service layer of Seagent provides the main security requirements for agent communication. The service layer provides a solution to add new mechanisms or new algorithms to the mechanisms without affecting the Seagent platform infrastructure. This approach increases granularity and makes the complex system structure simpler.

1.4. Thesis Organization

This thesis is organized as follows. This chapter includes the FIPA specified specs and the improvement of security concepts during its life cycle. Then the problems that can be compared are told. The mechanisms for these problems are explained in detail. However, the solution approach thinking about these issues on Seagent is determined.

Chapter 2 describes the agents and their characteristics. The behaviour of agents in multi-agent systems are identified. Then agent management services are shown. However, the transport message structure and communication model is identified. At last, Seagent platform architecture is basically defined.

Chapter 3 describes the main security concepts. In this chapter, the algorithms are told for providing expected security concerns in Seagent.

Chapter 4 describes firstly the security attacks against agents. Then the system's challenges and security requirements for agents are explained by the help of FIPA specified concepts. Then the attack scenarios are shown by using Seagent agent definitions. The interaction protocols are defined and Seagent interaction protocol is explained. Lastly, the previous approaches for agent security implementations are summarized to be able to have a basic information about the implementation scheme.

Chapter 5 describes the Seagent security. Design and implementation issues of Seagent security are explained in detail with diagrams. The related experimental results are measured and addressed for communication, encryption and signature scenarios.

Finally, chapter 6 gives the conclusion of this thesis work.

CHAPTER 2

AGENT PLATFORM SPECIFICATION AND SEAGENT

2.1. History and Definition of Agents

An agent is a computational process that implements the autonomous, communicating functionality of an application "(FIPA 2002a)". This is the FIPA definition for agents. But different scientists have different approaches to agent definitions.

In the definition of agents it is often mentioned that they can be described by mental attributes like knowledge and goals. Sometimes, this is used as the sole criterion: an agent is an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments. So agenthood is in the mind of the programmer "(Shoham 1993)".

Another well-ambitioned approach to define an agent is based on the concept of an autonomously acting entity: an autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future "(Franklin and Graesser 1996)".

One important influence of Agent-Oriented Technology is the rise of Distributed Artifical Intelligence at the beginning of the 1980s. The Distributed Artifical Intelligence branch deals with emergent high-level properties of complex distributed systems. Interaction and cooperation are important research topics in these intelligent systems "(Bond and Gasser 1988)". Another root of Agent-oriented Technology lies in parallel object-oriented programming (OOP), combining conventional OO-techniques with distributed systems.

The main branches of the concept of an agent are:

- **The agent as an object with competence:** Agents are seen as an extension to the OOP paradigm. Agents are recognized as complex objects with a higher degree of autonomy and flexible interactions "(Shoham 1993)".

- **The agent as part of a multi-agent system:** The interaction between agents are the focus of this conceptualization. Each agent is a specialized problem-solver, that communicates and cooperates with other agents to solve a common and complex problem "(Genesereth and Ketchpel 1994)".

- **The agent as an autonomous actor:** An agent has a view of its environment. It decides for itself about the goals to pursue and actions to be taken. Interactions with other agents are possible "(Franklin and Graesser 1996)".

- **The agent as a mental system:** Following this idea, an agent is a logical system for the description of a mental state including beliefs, capabilities, obligations, etc. This is problematic in so far, as those attributes sometimes differ considerably from the common understanding of these terms "(Shoham 1993)".

- **The agent as personal presentation:** The main motivation is to have the agent acting on behalf of its user without continuous control or interference of the user. Further, the agent shall adapt to his preferences "(Gilbert 1996)".

Agent-oriented techniques promise several advantages over traditional approaches, especially in the area of complex and highly distributed computing. As being highly parallel and multi-threaded entities, more efficient computing can be achieved.

Despite their relatively strict closure against its environment, agents are inherently communication-oriented. The properties that have been tried to be told yield increased flexibility and dynamics to the resulting overall system.

2.2. Characteristics of Agents and Multi Agent Systems

Autonomous agents and multi-agent systems(MAS) represent a new way of analyzing, designing, and implementing complex software systems. The agent-based view offers powerful tools and techniques that have the potential to considerably improve the way in which people conceptualize and implement many types of software. Agents are being used in an increasingly wide variety of applications ranging from comparatively small systems such as personalized email filters to large, complex, mission critical systems such as air-traffic control. At first sight, it may appear that such extremely different types of system can have little in common. It is the ease with which such a variety of applications can be characterized in terms of agents that leads researchers and developers to be so excited about the potential of the approach "(Jennings et al. 1998)". During these researches, researchers have investigated different characteristic issues. These could be identified as follows "(Borselius 2002)":

- **Situatedness** means that the agent receives sensory input from its environment and that it can perform actions which change the environment in some way.
- **Autonomy** means that agents are able to act without the direct intervention of humans (or other agents), and that it has control over its own actions and internal state.
- **Flexibility** can be defined to include the following properties:
  - **responsive:** agents ability to perceive their environment and respond in a timely fashion to changes that occur in it;
  - **pro-active:** agents are able to exhibit opportunistic, goal-driven behaviour and take the initiative where appropriate;
  - **social:** agents should be able to interact, when appropriate, with other agents and humans in order to complete their own problem solving and to help others with their activities.
- **Mobility** means that the ability for an agent to move across networks and between different hosts to fulfil its goals.
- **Rationality** means that the assumption that an agent will not act in a manner that prevents it from achieving its goals and will always attempt to fulfil those goals.
- **Veracity** means that an agent will not knowingly communicate false information.
- **Benevolence** means that an agent cannot have conflicting goals that either force it to transmit false information or to effect actions that cause its goals to be unfulfilled or impeded.

Each agent hides its functionality or task from the environment by only defining his relevance from its role. The autonomy refers to the agents ability in the sense, that it is not directly dependent on other agents or human users for providing its service or fulfilling its task. It may, and normally does make use of the service provisioning of other agents for his own working.

A multi-agent system is a system composed of multiple autonomous agents with the following characteristics "(Borselius 2002)":

- Each agent cannot solve a problem unaided.
- There is no global system control.
- Data is decentralised.
- Computation is asynchronous.

Computer platforms provide agents with environments in which they can execute. A platform typically also provides additional services, such as communication facilities, to the agents it is hosting. In order for agents to be able to form a useful open multi-agent system where they can communicate and cooperate, certain functionality needs to be provided to the agents. This includes functionality to find other agents or to find particular services. This additional functionality can either be implemented as services offered by other agents or as services more integrated into the MAS infrastructure itself.

Open multi-agent systems are usually envisioned as systems, communicating over the Internet, allowing anybody to connect to a platform on which agents are running. This means that the MAS lacks a global system control and that information in general is highly decentralised.

2.3. Agent Management Services

The FIPA defines three agent management services needed for an agent platform (AP) "(FIPA 2004)":

Directory Facilitator (DF): The DF is an information agent sharing information about registered services.

Agent Communication Channel (ACC): It is the standard communication channel between agents on an AP and between APs.

Agent Management System (AMS): The AMS is the actively managing entity on an AP. It controls the life cycle of agents and resource usage, including the ACC.

FIPA realized the need for platform integration and in the standard version of agent management specification, the extra functionality of extending the search to other platforms is added to the directory facilitator (DF) of the FIPA agent platform. DF, which provides yellow pages directory service to other agents in the platform, is a mandatory component of a FIPA compliant multi-agent platform. According to FIPA, multi-agent platform integration is achieved by DFs registering with each other.

2.4. FIPA Agent Communication Model

Figure 2.1 shows the FIPA message transport reference model

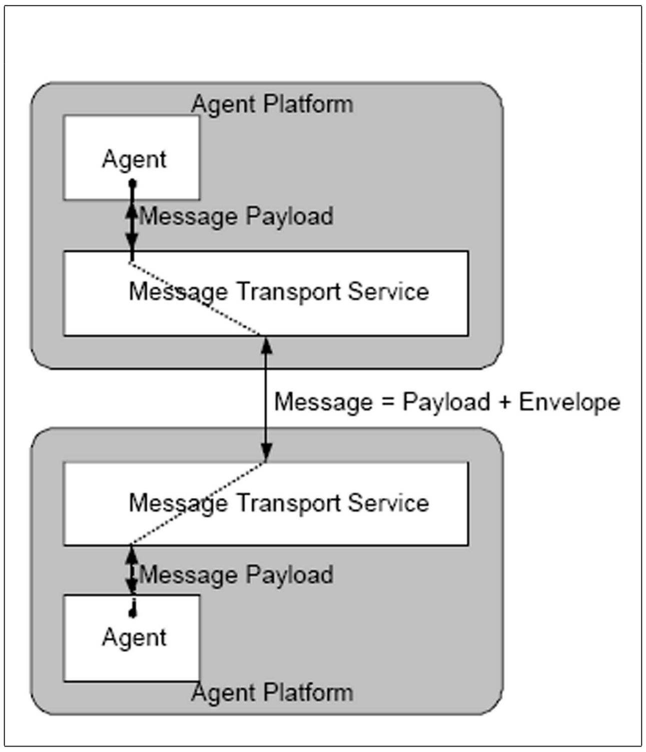

Agent Platform
Agent
Message Payload
Message Transport Service
Message = Payload + Envelope
Message Transport Service
Message Payload
Agent
Agent Platform

Figure 2.1. Message Transport Reference Model (Source: FIPA 2002e)

The reference model for agent message transport comprises three levels:

1. The Message Transport Protocol (MTP) is used to carry out the physical transfer of messages between two ACCs.
2. The Message Transport Service (MTS) is a service provided by the AP to which an agent is attached. The MTS supports the transportation of FIPA ACL messages between agents on any given AP and between agents on different APs.
3. The ACL represents the payload of the messages carried by both the MTS and MTP.

FIPA recognizes three options for an agent when sending a message to another agent residing on a remote platform illustrated in Figure 2.2:

1. Agent A sends the message to its local ACC using a proprietary or standard interface. The ACC then takes care of the transmission of the message to the correct remote ACC. The remote ACC will then eventually deliver the message.
2. Agent A sends the message directly to the ACC on the remote agent platform on which B resides. This remote ACC then delivers the message to B.
3. Agent A sends the message directly to agent B, using a direct communication mechanism. The message transfer, including buffering of messages and any error messages, must be handled by the sending and receiving agents.

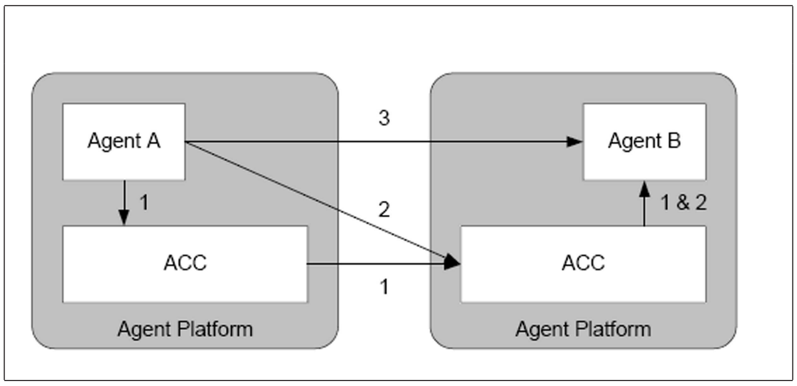


Figure 2.2. Comm. Methods between Different Platforms (Source: FIPA 2002e)

2.5. Agent Messages

The structure of a message is a key-value-tuple and is written in an agent-communication language, such as FIPA ACL. The content of the message is expressed in a content-language. Content expressions can be grounded by ontologies referenced within the ontology key-value-tuple. The messages also contain the sender and receiver names, expressed as agent-names. Agent-names are unique name identifiers for an agent. Every message has one sender and zero or more receivers. The case of zero receivers enables broadcasting of messages "(FIPA 2002a)".

A message is made up of a message envelope, containing transport information, and a message body comprising of the agent communication data or ACL message. An ACC should deliver the whole message, including the message envelope, to the receiving agent. However, it is possible for agent platforms to provide middleware layers to free agents from the task of processing the envelope.

As shown in Figure 2.3, a message is encoded into a payload suitable for transport over the selected message transport. An appropriate envelope is created that has sender and receiver information, which uses the transport description data appropriate to the transport selected. There may be additional envelope data to be included. The combination of the payload and envelope is termed as a transport-message.

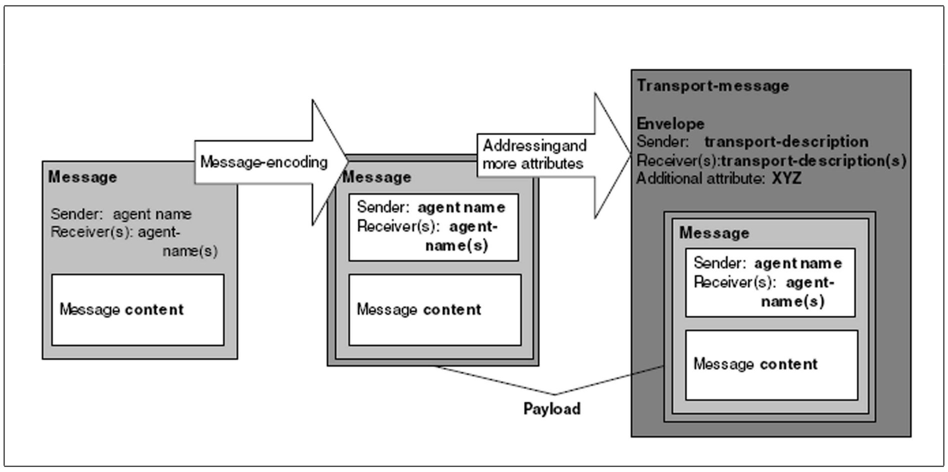

Message
Sender: agent name
Receiver(s): agent-name(s)
Message content
Message-encoding
Message
Sender: agent name
Receiver(s): agent-name(s)
Message content
Addressingand more attributes
Transport-message
Envelope
Sender: transport-description
Receiver(s):transport-description(s)
Additional attribute: XYZ
Message
Sender: agent name
Receiver(s): agent-name(s)
Message content
Payload

Figure 2.3. Transport Message Generation (Source: FIPA 2002a)

However, the envelope has transport-descriptions containing the information about how to send the message (via what transport, to what address, with details about how to utilize the transport). The envelope can also contain additional information, such as the encoding-representation, data related security, and other realization specific data that needs be visible to the transport or recipient(s).

In message validity, messages can be sent in such a way that any modification during transmission is identifiable. In message encryption, a message is sent in encrypted form such that non-authorized entities cannot comprehend the message content.

In the FIPA Abstract Architecture "(FIPA 2002a)", these features are accommodated through encoding-representations and the use of additional attributes in the envelope. For example, as the payload is encoded, one of the encodings could be to a digitally encrypted set of data, using a public key and preferred encryption algorithm. Additional parameters are added to the envelope to indicate these characteristics.

In Figure 2.4, the payload is encrypted, and additional attributes added to the envelope to support the encryption. These attributes must remain unencrypted in order that the receiving party is able to use them.

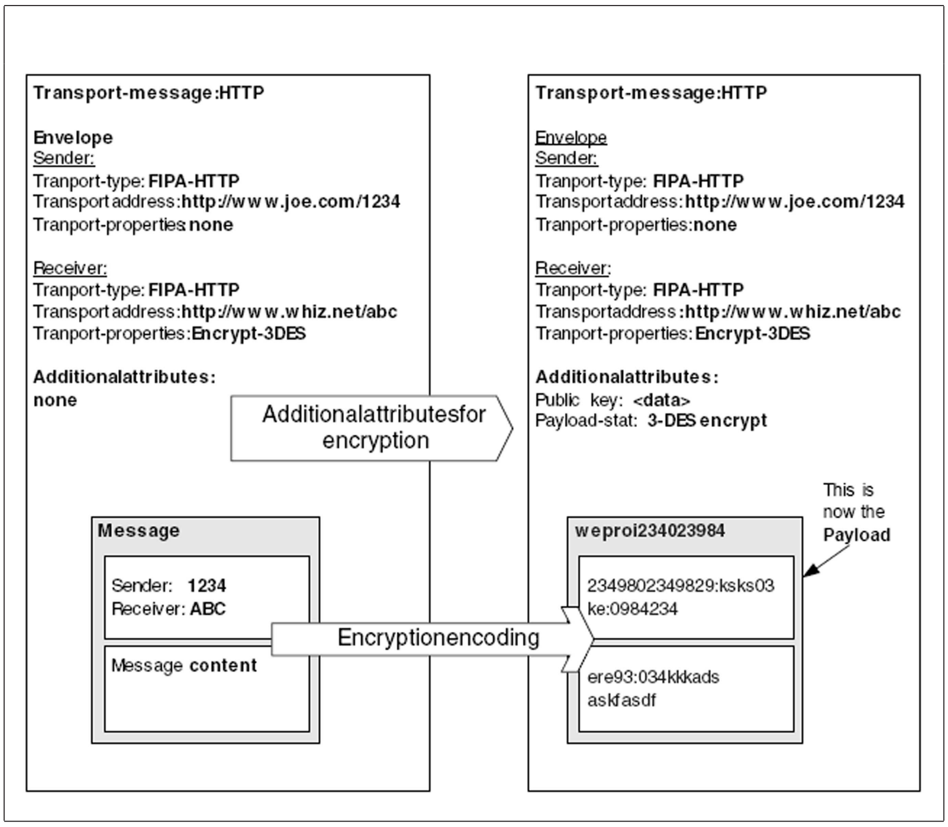


Figure 2.4. Encrypted Message Payload (Source: FIPA 2002a

After encrypting the message, the message payload includes an unrecognizable message and any intruder can not obtain an understandable message during communication.

These concepts will be explained in Chapter 4 with its new specification and implementation details.

2.6. Seagent

SEAGENT is a new agent development platform that is specialized for semantic web based multi agent system development "(Dikenelli et al. 2005)". The communication and plan execution infrastructure of SEAGENT looks like other existing agent development frameworks such as DECAF "(Graham et al. 2003)", JADE "(Bellifemine et al. 2001)", and RETSINA "(Sycara et al. 2003)". However, to support and ease semantic web based MAS development, SEAGENT includes the following built-in features that the existing agent frameworks and platforms do not have "(Dikenelli et al. 2005)":

- Agents created using SEAGENT handle their internal knowledge base using semantic web standards and the platform provides specifically designed interfaces to manage and query the internal knowledge without being dependent on a particular application programming interface.
- The directory service of SEAGENT is implemented in a way that the directory knowledge is held in semantic web standards and the directory service supports semantic matching of the agent capabilities to find the semantically similar agents.
- FIPA-RDF"(Fikes et al. 2003)" content language has been used to transfer semantic content in the agent communication language messages and OWL-QL is integrated to the FIPA-RDF content language to query the agents and services.
- SEAGENT introduces a new service for managing and translating ontologies. It provides a means to define mappings between platform ontologies and external ontologies. The translation process is based on these defined mappings.
- SEAGENT supports discovery and dynamic invocation of semantic web services by introducing a new platform service for semantic service discovery and a reusable agent behavior for dynamic invocation of the discovered services.

2.6.1. Platform Architecture

Seagent has a layered software architecture. Each layer has been specially designed to provide build-in support for MAS development on the Semantic Web environment. The overall architecture is shown in Figure 2.5.

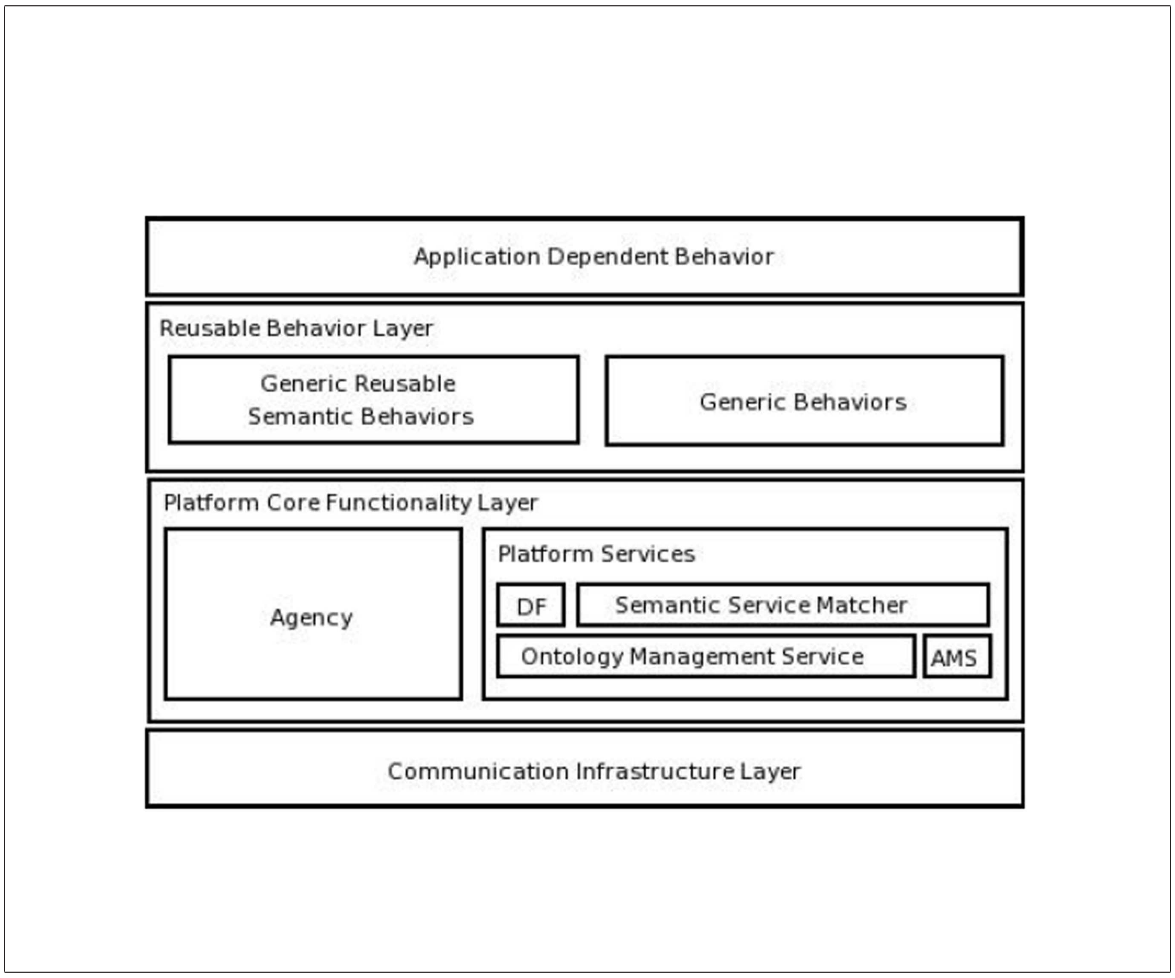

Application Dependent Behavior
Reusable Behavior Layer
Generic Reusable Semantic Behaviors
Generic Behaviors
Platform Core Functionality Layer
Agency
Platform Services
DF
Semantic Service Matcher
Ontology Management Service
AMS
Communication Infrastructure Layer

Figure 2.5. Seagent Platform Overall Architecture (Source: Dikenelli et al. 2005)

Because of interesting in Seagent message security, it would only be explained about Communication Infrastructure Layer. This layer has an aim of abstracting platforms communication infrastructure implementation. Seagent implements FIPAs Agent Communication and Agent Message Transport specifications to handle agent messaging. Although Communication Infrastructure Layer can transfer any content using FIPA ACL and transport infrastructure, Seagent platform only supports Seagent Content Language (SCL), a specific OWL ontology to define the ACL content, by default.

2.6.2. Security for Platform Architecture

According to Figure 2.6, agency, communication infrastructure layer and platform services are the main affected concerns or Seagent security. By thinking about the end-to-end security as an initial concern, the communication of agents with secure messages are determined.

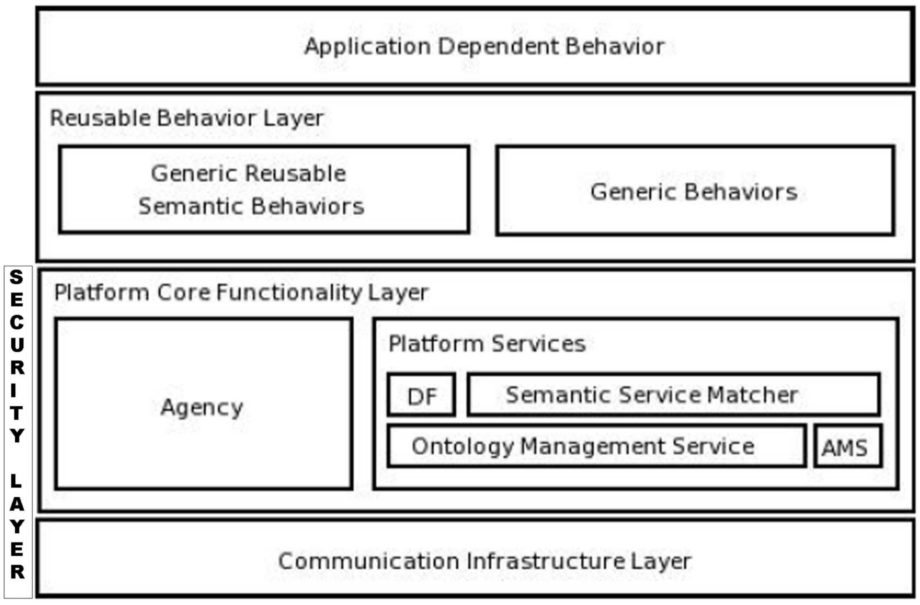

Application Dependent Behavior
Reusable Behavior Layer
Generic Reusable Semantic Behaviors
Generic Behaviors
SECURITY LAYER
Platform Core Functionality Layer
Agency
Platform Services
DF
Semantic Service Matcher
Ontology Management Service
AMS
Communication Infrastructure Layer

Figure 2.6. Seagent Platform Overall Architecture with Security

There are user agents, service agents and middleware or middle agents such as name services and directory services. While there are certain entities in the system that for a variety of reasons such as performance and because of existing practices, some services are represented as agents and some are not. DF and AMS are implemented as agents and their communication services between other agents have to be secure. So they have to be private and public information to be able to provide security.

User agents and service agents start to communicate with each other after middle agent operations. During these conversations, after each of these agents have the other party's public key information, the secure messaging occurs successfully.

MAS systems are fundamentally message-based as Seagent, therefore threats to the communication, such as corruption of transmitted data and eavesdropping as explained in Section 3.3, need to be guarded against. A traditional safeguard for asynchronous communication in Seagent is to provide a secure envelope for each message sent, on a per message basis. The FIPA Message Transport service specification "(FIPA 2002e)" specifies an optional tag in the message transport envelope for security. This tag shows the message's security features and the other agent can understand what it should process on this message.

In Seagent, "Security" field of the message envelope is thought as per message basis. There is no abstraction to specify message security for a group of messages such as on a per session or on a per interaction sequence. Each individual message is processed individually. On the other hand, synchronous message security models such as the Secure Socket Layer (SSL) message specification do contain the concept of a message stream. FIPA MAS can exchange messages using a FIPA specified MTP (Message Transport Protocol) such as HTTP over a lower level security protocol such as SSL. However, SSL is at a much lower level of abstraction than the level of the agent communication interaction sequences.

CHAPTER 3

SECURITY CONCEPTS

3.1. Overview

Information security is a term used to describe the process of protecting information and services from misuse or destruction. Before the widespread use of data processing equipment, data and information valuable to an organization were made invulnerable primarily by physical and administrative means. With the advent of computer networks and the internet however, the design of automated tools which ensure security and privacy of information transferred across networks was inevitable. Especially the introduction of distributed systems and the use of networks and communication facilities for carrying data between different hosts or computers has introduced a major challenge for security. Then the concerns about security have been reconstructed.

Secure systems are designed so that the cost of breaking any component of the system outweighs the rewards. Cost is usually measured in money, time, and risk, both legal and personal. The benefits of breaking systems are generally control, money, or information that can be sold for money. The security of the system should be proportional to the resources it protects "(Knudsen 1998)".

To assess the security needs of an organization effectively and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. One approach is to consider three aspects of information security "(Stallings 2003)":

- **Security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A mechanism that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A service that enhances the security of the data processing systems and the information transfers of an organization.

The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

3.2. Security Concerns

Cryptography is the science of secret writing. It's a branch of mathematics, part of cryptology. Cryptology has one other child, cryptanalysis, which is the science of breaking (analyzing) cryptography.

At the application programming level, there are many options for making a program secure. Cryptography is the biggest tool for the application programmer. But it is important to realize that a cryptographically enabled program is not necessarily a secure one. Without a carefully planned and constantly scrutinized security strategy, cryptography won't do much good performance.

Correctly used, cryptography provides these standard security features "(Knudsen 1998)":

- **Confidentiality** assures that data cannot be viewed by unauthorized people.
- **Integrity** assures that data has not been changed without the sender's knowledge.
- **Authentication** assures that sender deals with correct receiver, not with imposters.

3.2.1. Confidentiality

With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. Confidentiality is also determined for the protection of traffic flow. This requires that an attacker not be able to observe the source and destination,frequency,length or other communication facilities.

A common way to protect information is to encrypt it at the sending end and decrypt it at the receiving end. Encryption is the process of taking data, called plaintext, and mathematically transforming it into an unreadable mess, called ciphertext. Decryption takes the ciphertext and transforms it back into plaintext. The mathematical algorithm that performs the transformations is called a cipher. Figure 3.1 shows how this works.

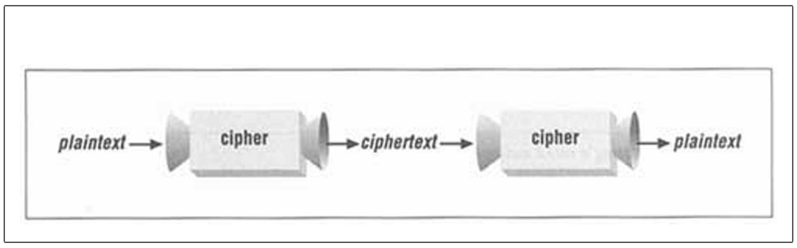
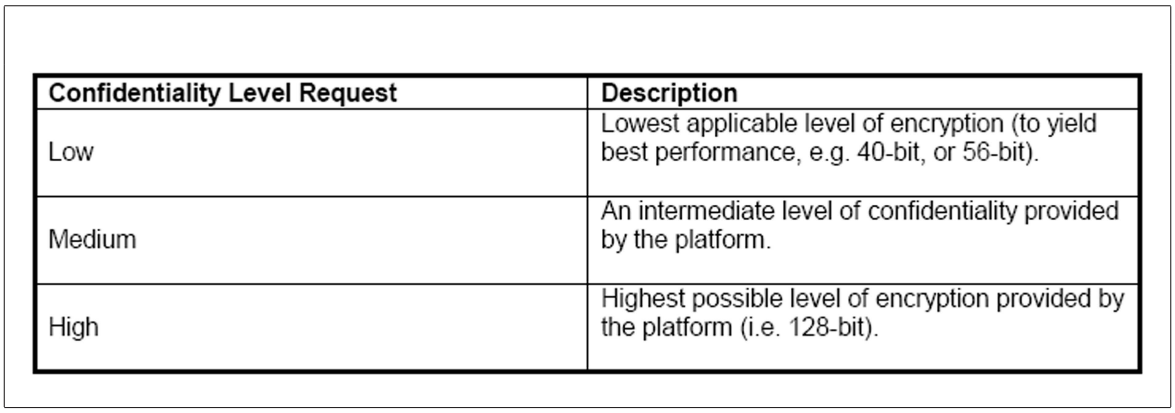


Figure 3.1. Operation of a Cipher (Source: Knudsen 1998)

Useful ciphers use keys to encrypt and decrypt data. A key is a secret value. If you encrypt the same plaintext using different keys, you will get different ciphertexts. Similarly, ciphertext can only be decrypted to the original plaintext using the proper key.

3.2.1.1. Confidentiality Levels

According to FIPA Agent Security Management Specification "(FIPA 1998)", an agent can request a low, medium, or high level of confidentiality. These confidentiality levels are provided so that the agent is not burdened with the responsibility of deciding on specific security mechanisms, but is only responsible for determining the sensitivity of the data which it produces. These levels are shown in Figure 3.2.

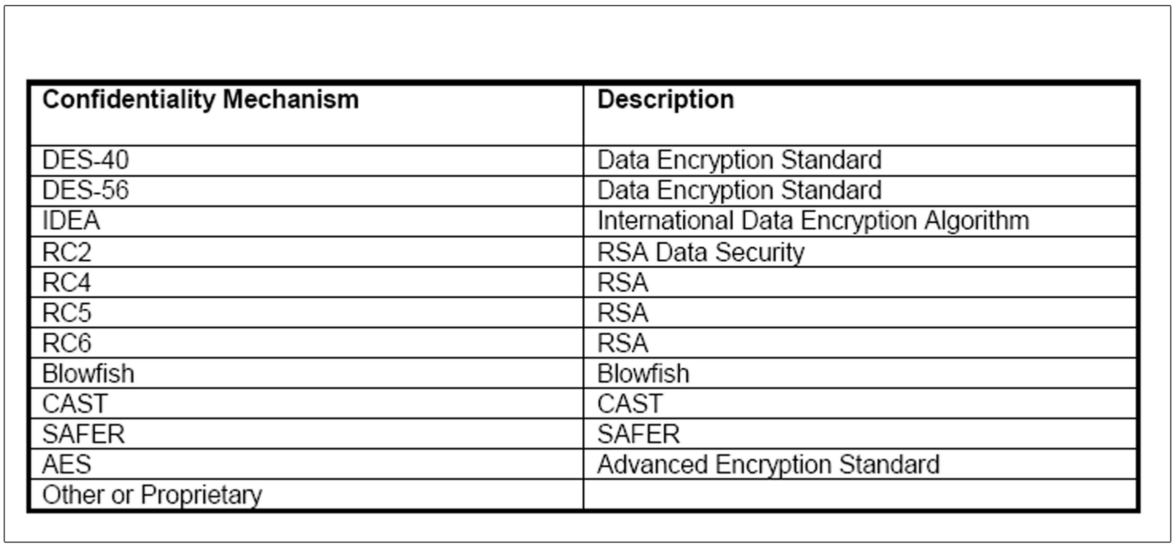


Confidentiality Level Request	Description
Low	Lowest applicable level of encryption (to yield best performance, e.g. 40-bit, or 56-bit).
Medium	An intermediate level of confidentiality provided by the platform.
High	Highest possible level of encryption provided by the platform (i.e. 128-bit).

Figure 3.2. Confidentiality Levels (Source: FIPA 1998)

3.2.1.2. Confidentiality Mechanisms

Confidentiality mechanisms that could be used by agent security applications are shown in Figure 3.3. Alternatively, an agent can request a specific security mechanism to ensure confidentiality. According to chosen confidentiality level, confidentiality mechanism can be more specific.



Confidentiality Mechanism	Description
DES-40	Data Encryption Standard
DES-56	Data Encryption Standard
IDEA	International Data Encryption Algorithm
RC2	RSA Data Security
RC4	RSA
RC5	RSA
RC6	RSA
Blowfish	Blowfish
CAST	CAST
SAFER	SAFER
AES	Advanced Encryption Standard
Other or Proprietary	

Figure 3.3. Confidentiality Mechanisms (Source: FIPA 1998)

3.2.2. Symmetric Encryption Algorithms

Symmetric encryption, or secret-key encryption, uses a secret key to encrypt a message into ciphertext and the same key to decrypt the ciphertext into the original message.

Symmetric cryptography, which includes symmetric encryption, requires the sender and receiver to agree on a shared secret key. Symmetric cryptography is in general more efficient in terms of computing resources than asymmetric cryptography.

In symmetric cryptography, the key has to be kept secret, and you have to trust your recipient to keep the key a secret also. If someone else obtains the key, you and your recipient have to agree on a new key in a secure manner.

Operation of the symmetric key encryption scheme is shown in Figure 3.4.

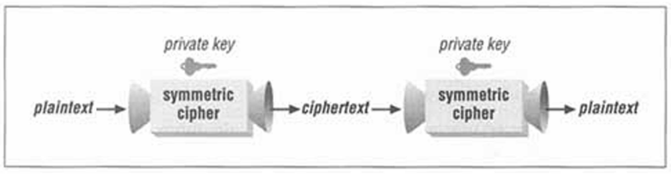


Figure 3.4. Operation of Symmetric Encryption (Source: Knudsen 1998)

Symmetric ciphers come in two varieties. Block ciphers encrypt and decrypt fixed-size blocks of data, usually 64 bits long. Stream ciphers operate on a stream of bits or bytes. A block cipher can be made to work like a stream cipher, using the appropriate mode. The use of computers in cryptography has led to the creation of block ciphers, in which a message is broken into blocks. The cipher encrypts or decrypts one block at a time. If data has been encrypted between the client and server with a block cipher, server has to wait until the client typed enough characters to fill a block. In this case, a stream cipher is better suited to the task.

3.2.2.1. Data Encryption Standard(DES-56)

DES encrypts a plaintext bitstring x of length 64 using a key K which is a bitstring of length 56, obtaining a ciphertext bitstring which is again a bitstring of length 64. The algorithm proceeds in three stages "(Stinson 1995)":

1. Given a plaintext x, a bitstring x_0 is constructed by permuting the bits of x according to a (fixed) initial permutation IP. It is written that $x_0 =$ IP (x) $= L_0R_0$, where L_0 comprises the first 32 bits of x_0 and R_0 the last 32 bits.

2. 16 iterations of a certain function are then computed, L_iR_i.

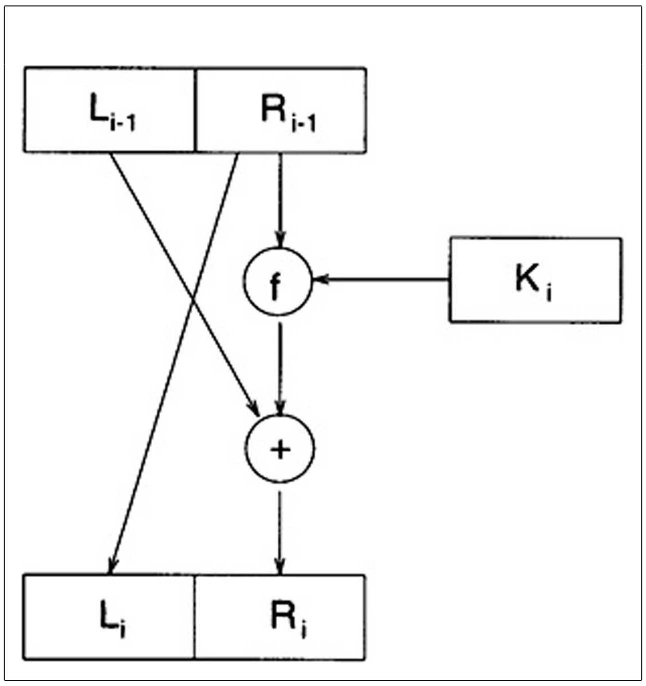


Figure 3.5. One round of DES Encryption (Source: Stinson 1995)

$1 \leq i \leq 16$, according to the following rule:

- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

where $\oplus$ denotes the exclusive-or of two bitstrings. f is a function that has been described below, and K_1, K_2, . . . , K_{16} are each bitstrings of length 48 computed as a function of the key K. (Actually, each K_i is a permuted selection of bits from K.) K_1, K_2, . . . , K_{16} comprises the key schedule. One round of encryption is depicted in Figure 3.5.

3. It has been applied the inverse permutation IP^{-1} to the bitstring $R_{16}L_{16}$, obtaining the ciphertext y. That is, y = $IP^{-1}(R_{16}L_{16})$.

The function f takes as input a first argument A, which is a bitstring of length 32, and a second argument J that is a bitstring of length 48, and produces as output a bitstring of length 32. The first argument A is expanded to a bitstring of length 48 according to a fixed expansion function E. E(A) consists of the 32 bits from A, permuted in a certain way, with 16 of the bits appearing twice. Then E(A) $\oplus$ J is computed. Eight 6 bit string result is processed with S boxes, 4*16 array whose entries come from the integers 0 - 15. The resulting bitstring C = $C_1C_2C_3C_4C_5C_6C_7C_8$ of length 32 is permuted according to a fixed permutation P. The resulting bitstring P(C) is defined to be f(A, J). The f function is depicted in Figure 3.6. Basically, it consists of a substitution (using an S-box) followed by the (fixed) permutation P. The 16 iterations of f comprise a product cryptosystem.

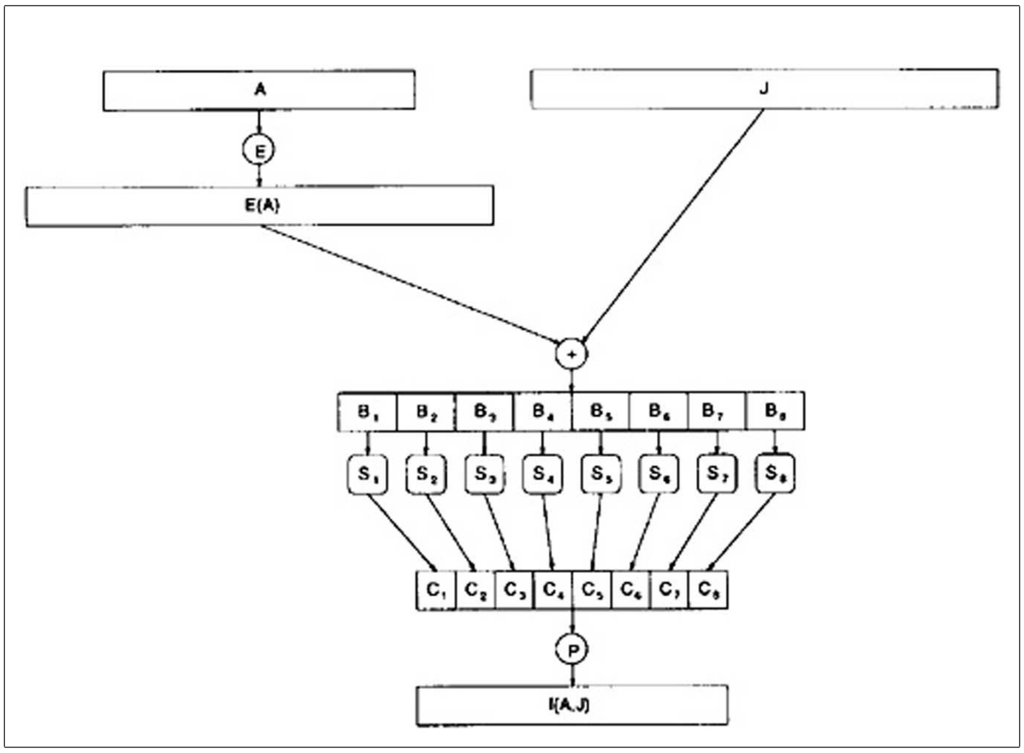

A
J
E
E(A)
+
B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8
C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8
P
f(A,J)

Figure 3.6. The DES f Function (Source: Stinson 1995)

3.2.2.2. Triple DES

Triple DES has been the answer to many of the shortcomings of DES. Since it is based on the DES algorithm, it is very easy to modify existing software to use Triple DES. It also has the advantage of proven reliability and a longer key length that eliminates many of the shortcut attacks that can be used to reduce the amount of time it takes to break DES.

Triple DES is simply another mode of DES operation. It takes three 64-bit keys, for an overall key length of 192 bits. The procedure for encryption is exactly the same as regular DES, but it is repeated three times. The data is encrypted with the first key, decrypted with the second key, and finally encrypted again with the third key as in Equation 3.1 "(Stallings 2003)".

$$C = E_{K_3}[D_{K_2}[E_{K_1}[P]]] \tag{3.1}$$

where

C=ciphertext

P=plaintext

$E_K[X]$=encryption of X using key K

$D_K[Y]$=decryption of Y using key K

Decryption is simply the same operation with the keys reversed as in Equation 3.2.

$$P = D_{K_1}[E_{K_2}[D_{K_3}[C]]] \tag{3.2}$$

Consequently, Triple DES runs three times slower than standard DES, but is much more secure if used properly. The procedure for decrypting something is the same as the procedure for encryption, except it is executed in reverse. Like DES, data is encrypted and decrypted in 64-bit chunks. Unfortunately, there are some weak keys that one should be aware of: if all three keys, the first and second keys, or the second and third keys are the same, then the encryption procedure is essentially the same as standard DES. This situation is to be avoided because it is the same as using a really slow version of regular DES.

Although the input key for DES is 64 bits long, the actual key used by DES is only 56 bits in length. The least significant (right-most) bit in each byte is a parity bit, and should be set so that there are always an odd number of 1s in every byte. These parity bits are ignored, so only the seven most significant bits of each byte are used, resulting in a key length of 56 bits. This means that the effective key strength for Triple DES is actually 168 bits because each of the three keys contains 8 parity bits that are not used during the encryption process.

3.2.2.3. Advanced Encryption Standard(AES)

AES is a symmetric block cipher with a block length of 128 bits and support for key lengths of 128,192 and 256 bits "(Stallings 2003)". It is fast and compact with a simple mathematical structure that eliminated much of the cumbersome overhead of 3DES. AES is also easily implemented in hardware, can be integrated into a wide range of electronic systems requiring secure data transactions.

3.2.3. Asymmetric Encryption Algorithms

Asymmetric cryptography, or public-key cryptography, involves the use of two distinct keys, one public and one private. The private key is kept secret by its owner, while the public key can be freely shared with everyone. There are a number of different types of asymmetric cryptographic schemes, including encryption schemes and digital signatures.

While asymmetric cryptography does not, like symmetric cryptography, rely on the sender and receiver agreeing on a shared secret, the user of the public key must ensure that the correct key is used. Public Key Infrastructures (PKIs) are used for this purpose.

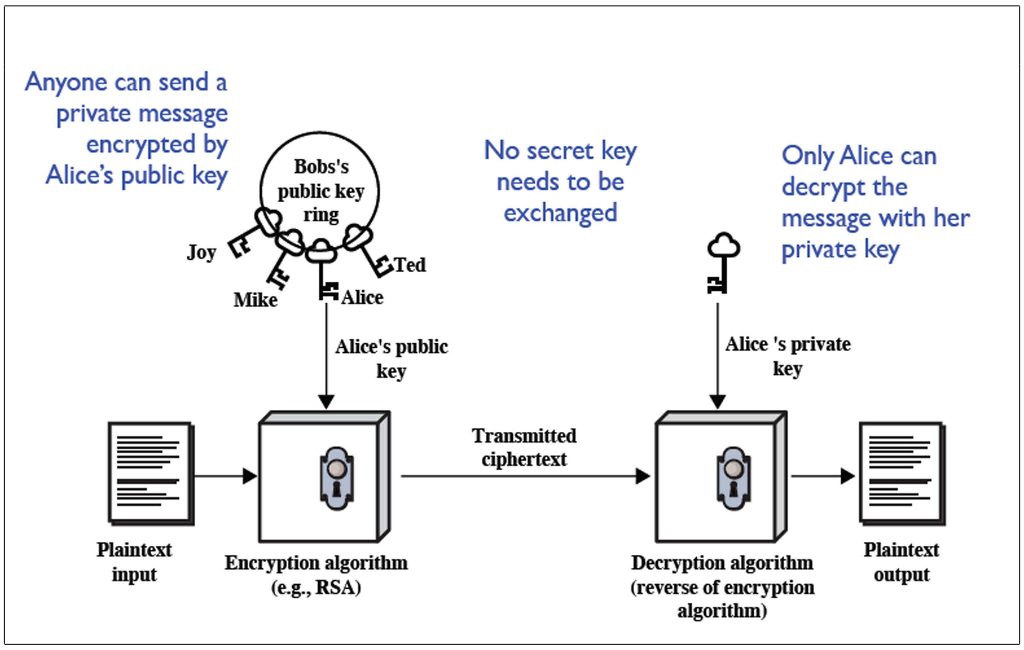

Anyone can send a private message encrypted by Alice's public key
Bobs's public key ring
Joy
Mike
Alice
Ted
No secret key needs to be exchanged
Only Alice can decrypt the message with her private key
Alice's public key
Alice 's private key
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 3.7. Public-Key Cryptography

Data encrypted using the public key can be decrypted using the private key. No other key will decrypt the data, and the private key will decrypt only data that was encrypted using the matching public key. In some cases, the reverse of the process also works; data encrypted with the private key can be decrypted with the public key.

Asymmetric ciphers are much slower than symmetric ciphers, so they are not usually used to encrypt long messages.

3.2.3.1. RSA

RSA is by far the most popular public-key cryptography algorithm. It supports both encryption and digital signatures. It is also the easiest one to describe and implement.

In RSA, a public key is based on the product of two large prime numbers. These two numbers must be kept secret as they are used to compute the private key. The product of the two prime numbers is referred to as modulus. The security of RSA lies in the difficulty of factoring large numbers.

RSA supports variable key lengths, especially 512 bits. Block size can be variable. Plaintext must be shorter than key length and ciphertext must be with the same size of key length.

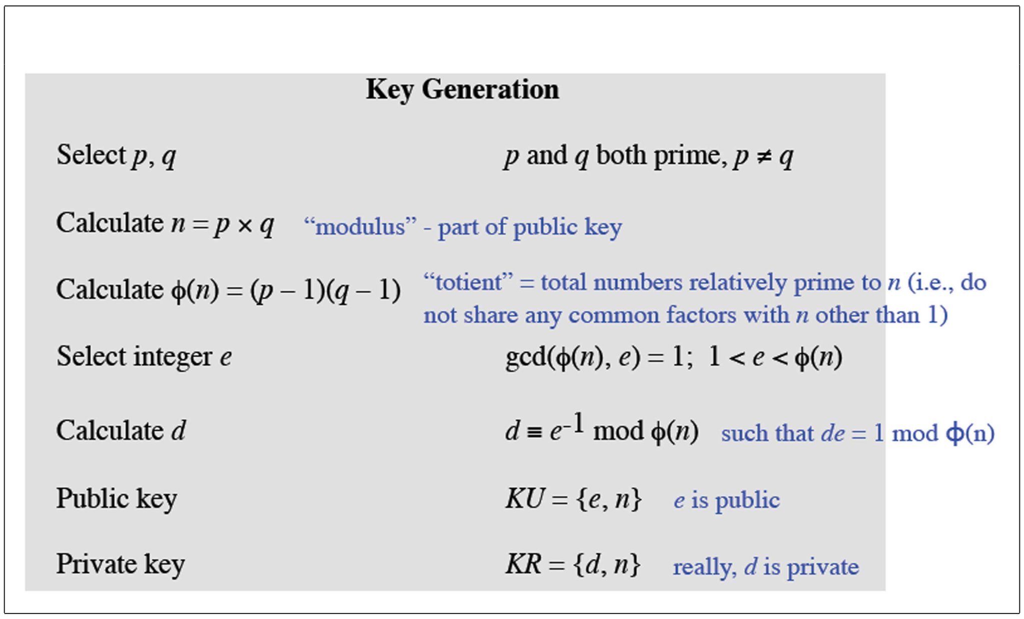
Encryption(Equation 3.3) and decryption(Equation 3.4) are of the following form, for some plaintext block M and ciphertext block "(Stallings 2003)":

$$C{=}M^{e}\ (mod\ n) \tag{3.3}$$

$$M{=}C^{d}(mod\ n){=}(M^{e})^{d}(mod\ n){=}M^{ed}(mod\ n) \tag{3.4}$$

Both sender and receiver must know the values of n and e, and only the receiver knows the value of d. This is a public-key encryption algorithm with a public key of $KU{=}\{e,n\}$ and a private key of $KR{=}\{d,\ n\}$.

Public-private key pair is generated as shown in Figure 3.8.



Key Generation

Select p, q	p and q both prime, $p \neq q$	
Calculate $n = p \times q$		“modulus” - part of public key
Calculate $\phi(n) = (p-1)(q-1)$		“totient” = total numbers relatively prime to n (i.e., do not share any common factors with n other than 1)
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$	
Calculate d	$d \equiv e^{-1} \bmod \phi(n)$	such that $de = 1 \bmod \phi(n)$
Public key	$KU = \{e, n\}$	e is public
Private key	$KR = \{d, n\}$	really, d is private

Figure 3.8. RSA Key Generation

If p and q is known, encryption/decryption is easy (modular exponentiation). If p and q is not known, finding the private key is equivalent to trying to factor n=pq into its prime factors. RSA is an effective algorithm because of difficulty for factoring large numbers.

3.2.4. Hybrid Systems

Hybrid systems combine symmetric and asymmetric ciphers. The beginning of a conversation involves some negotiation, carried out using an asymmetric cipher, where the participants agree on a private key, or session key. The session key is used with a symmetric cipher to encrypt the remainder of the conversation. The session key's life is over when the two participants finish their conversation. If they have a new conversation, they'll generate a new session key, which makes the cryptanalyst's job harder.

Seagent security uses hybrid systems to share session keys between agents.

3.2.5. Integrity

Data integrity is a service which addresses the unauthorized alteration of data. It does this by detecting data manipulation by unauthorized entities.

A message digest can be used to verify data integrity. It is used to turn input of arbitrary length into an output of fixed length. This output can then be used in place of the original input. This has many advantages. The output always has the same length, so this can be taken into account when processing or storing the message digest. Also, the output is much shorter than the input, so that processing and storing can be done much quicker.

To make this work, a message digest function should have two properties:

1. Given a particular message digest, it should be very difficult to find an input that has the same message digest.
2. It should be very difficult to find two inputs that have the same message digest.

The first property prevents taking a particular digest and using it in connection with another message. Message can't be known because of this first property.

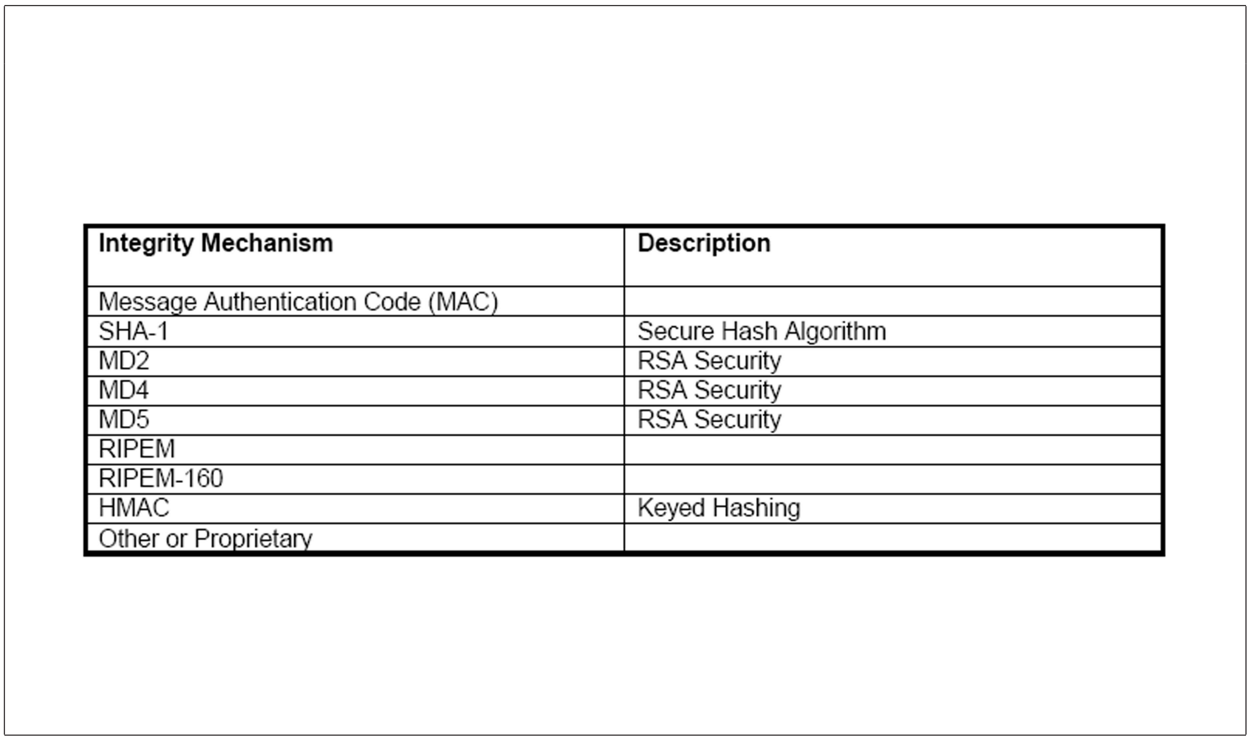
The second property means that if two inputs produce the same message digest, they must be the same input as well. Often the second requirement is taken even further: if a message is changed slightly, the message digest of the changed message should have changed a great deal.

To verify whether the received message is identical to the original, the recipient computes the hash of the received file. This hash is then compared to the hash of the original file. If these two are identical, integrity is verified.

3.2.5.1. Integrity Mechanisms

Integrity mechanisms that could be used by agent security applications are shown in Figure 3.9. Alternatively, an agent can request a specific integrity mechanism to ensure integrity.

Although other integrity mechanisms could be easily adapted to Seagent, MD5 is used to provide integrity. So only MD5 is explained in this thesis.



Integrity Mechanism	Description
Message Authentication Code (MAC)	
SHA-1	Secure Hash Algorithm
MD2	RSA Security
MD4	RSA Security
MD5	RSA Security
RIPEM	
RIPEM-160	
HMAC	Keyed Hashing
Other or Proprietary	

Figure 3.9. Integrity Mechanisms (Source: FIPA 1998)

3.2.5.2. Message Digest Algorithm(MD5)

MD5 is a widely-used cryptographic hash function with a 128-bit hash value. MD5 processes a variable length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks; the message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512. The remaining bits are filled up with a 64-bit integer representing the length of the original message.

The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words. These are initialized to certain fixed constants. The main algorithm then operates on each 512-bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed rounds; each round is composed of 16 similar operations based on a non-linear function, modular addition, and left rotation. There are four possible functions that is used in each round.

3.2.6. Authentication

Authentication is provided to guarantee that entities are who they claim to be, or that information has not been manipulated by unauthorized parties "(Menezes et al. 1997)". Secrecy and authentication were truly separate and independent information security objectives.

It may at first not seem important to separate the two but there are situations where it is not only useful but essential. For example, if a two-party communication between Alice and Bob is to take place where Alice is in one country and Bob in another, the host countries might not permit secrecy on the channel; one or both countries might want the ability to monitor all communications. Alice and Bob, however, would like to be assured of the identity of each other, and of the integrity and origin of the information they send and receive.

An asymmetric cipher can be used for authentication. It is supposed that Alice encrypts her message using her private key. When Bob receives Alice's message, he decrypts it using her public key. He can be sure that the file is from Alice because only Alice's private key could have encrypted the file in the first place. This type of authentication is called as data origin authentication. Seagent security is especially interested in data origin authentication.

3.2.7. Digital Signature

Digital signature is data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery(e.g. by the recipient) "(Stallings 2003)".

The twin requirements to give assurance that a message is generated by a given entity (Data Origin Authentication), and to give assurance that it has not subsequently been altered (Data Integrity) cannot be separated. For if a message has been altered then this can be viewed as a change of origin. And if an origin cannot be determined then there is no basis from which to discuss alteration "(Gordon 1998)".

A signature scheme consists of two components: a signing algorithm and a verification algorithm. Bob can sign a message x using a (secret) signing algorithm *sig*. The resulting signature *sig(x)* can subsequently be verified using a public verification algorithm *ver*. Given a pair *(x, y)*, the verification algorithm returns an answer true or false depending on whether the signature is authentic. RSA public-key cryptosystem to provide digital signatures is shown below "(Stinson 1995)":

Definition 3.2..1. Let $n = pq$, where p and q are primes. Let $\mathtt{P} = \mathtt{A} = \mathbb{Z}_n$, and define

$$\mathtt{K} = \{(n, p, q, a, b) : n = pq, p, q \text{ prime}, ab \equiv 1(mod\phi(n))\}$$

The values n and b are public, and the values p, q, a are secret.
For $K = (n, p, q, a, b)$, define

$$sig_K(x) = x^a \ (mod\ n) \tag{3.5}$$

and

$$ver_K(x, y) = true \Leftrightarrow x \equiv y^b \ (mod\ n) \tag{3.6}$$

$(x, y \epsilon \mathbb{Z}_n)$

Seagent security implements above digital signature scheme by using one of the Java's implemented algorithms as RSAwithMD5.

3.2.8. Non-repudiation

Non-repudiation means to ensure that a transferred message has been sent and received by the parties claiming to have sent and received the message. Non-repudiation is a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

Non-repudiation can be obtained through the use of:

- **Digital signatures** function as a unique identifier for an individual.
- **Confirmation services** can create digital receipts to indicated that messages were sent and/or received.
- **Timestamps** contain the date and time a document was composed and proves that a document existed at a certain time.

CHAPTER 4

AGENT PLATFORM SECURITY

4.1. Security Attacks

Security attacks are divided into two categories, passive and active attacks. Passive attacks relates to attacks on confidentiality to assets, while active attacks relates to attacks on integrity and availability to assets "(Stallings 2003)".

Security attacks related to confidentiality "(Houmb 2002)":

- Eavesdropping
- Traffic analysis

Eavesdropping is an attack where the attacker exposes some or all of the content in a message. This is achieved by tapping data sent on the network either through special equipment (hardware), software, or by an attacker acting as a man in the middle. When an outsider (or insider) acts as a man in the middle, all information is sent through the attacker on its way to its destination. Traffic analysis does not reveal any of the content in the message but monitors and probably also logs where and to whom the information is sent. An attacker could then be able to predict the nature of the information sent, based on the size of the messages and on the sender and the receiver.

Security attacks related to integrity "(Houmb 2002)":

- Modification of message contents
- Masquerade
- Fabrication
- Replay

Modification of message contents is an attack where the attacker alters some or all of the content in a message when being transported over the network, or stored on a storage device. Masquerade is an attack where the attacker acts on behalf of the original sender or receiver. Fabrication is an attack where the attacker forges a message.

Security attacks related to availability "(Houmb 2002)":

- Denial of service
- Masquerade
- Fabrication
- Replay

Denial of service is an attack where services or resources are blocked for authorized use or when a client or a server is exposed to such exhausting load that they are not capable of performing normal activities. Denial of service also relates to all types of network problems, such as unstable network communication, other problems with the network connection or simply because someone has removed or destroyed the communication medium. In this case the attack could be intended or unintended. Fabrication, masquerade and replay in connection with availability relate to situations where the purpose is to prevent legal access "(Houmb 2002)".

4.1.1. Security Attacks Through Agent Communication

When security attacks are thought about agents, firstly three properties of agents have to be concerned "(Poslad et al. 2001)":

1. Agents communicate with other agents using syntactically and semantically rich complex messages.
2. Agents make heavy use of facilitators, match-makers and brokers to discover and hence interoperate and cooperate with unknown agents.
3. Agents within one domain or agency interact with agents in other autonomous domains or agencies.

A common way to break into a distributed system is to cause the message handler to fail leaving the system in a state where remote access rights are increased. Due to the complexity of agent interaction (Figure 4.1), many different kinds of syntactic and semantic attack can occur, causing the system to fail.

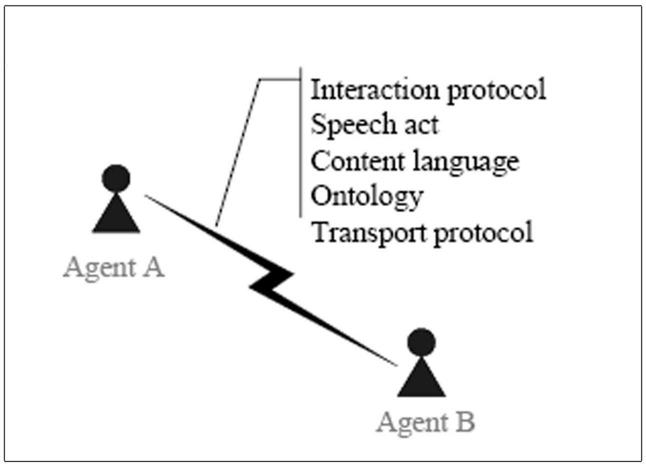


Figure 4.1. Comm. of agents using rich complex messages (Source: Poslad et al. 2001)

Agents frequently interoperate with other agents in order to jointly solve tasks they cannot solve themselves. This provides agents with fault-tolerance during dynamic service provision. But in order to find a suitable provider agent, agents frequently make use of other intermediary or mediator agents (Figure 4.2). A service agent registers with a directory agent, that directory agent registers with another directory agent. A user agent searches a remote directory agent via a local directory agent in order to locate a suitable remote service. The mediator services become an obvious focus for Denial of Service (DoS) attacks, masquerades of one agent by a different agent and the introduction of faulty services that individually or in combination can disrupt the whole agency "(Poslad et al. 2001)".

Agents within one agency can often need to interact with agents in another agency, for example, because particular services do no reside in the local agency (Figure 4.3). Whereas it is not that difficult to develop a trust model within individual agencies, it is far more complex to develop trust models that enable co-operation between multiple autonomous agencies that perhaps have their own policies for interoperation "(Poslad et al. 2001)".

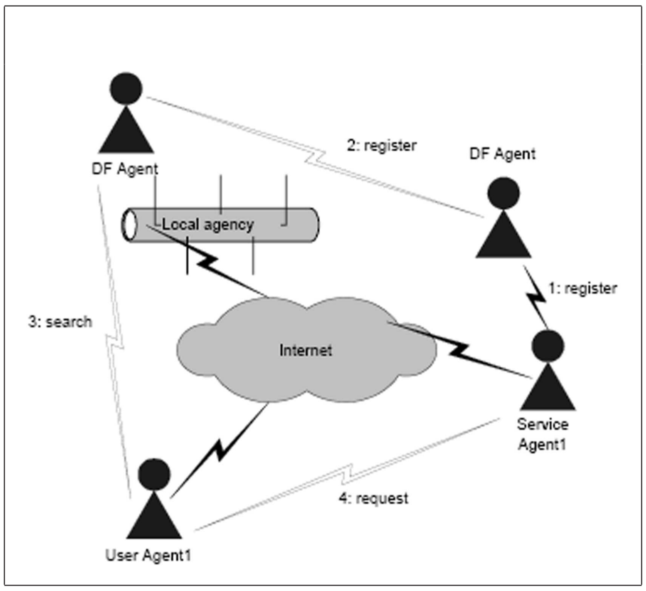


Figure 4.2. Mediators to interoperate with unknown agents (Source: Poslad et al. 2001)

When the use of the mediator agents is considered, for example authenticate write access can be given to the mediator agents - this would help guard against one agent masquerading as another. However, it could be easily tie up the mediator with frequent multiple reads and cause a DoS. It could be prevented by introducing authentication for read access but this would interfere with bootstrapping the system and hinder unknown agents having the option of browsing an agencys information before it joins it. If the mediator behaves as an intermediary between a first-party and second-party agent, different encryption schemes and privacy schemes could be improved to share information with the second party but protect it from the third party and vice versa "(Poslad et al. 2001)".

When multi agent system interaction is considered, each agency is easily protected by using firewalls, secure channels, access control and authentication against agents outside the agency. However, if multiple MAS need to cooperate, it is needed to reveal some aspects of the system security to other agencies that are autonomous to first agency, and first agency may be currently uncertain how to trust them "(Poslad et al. 2001)".

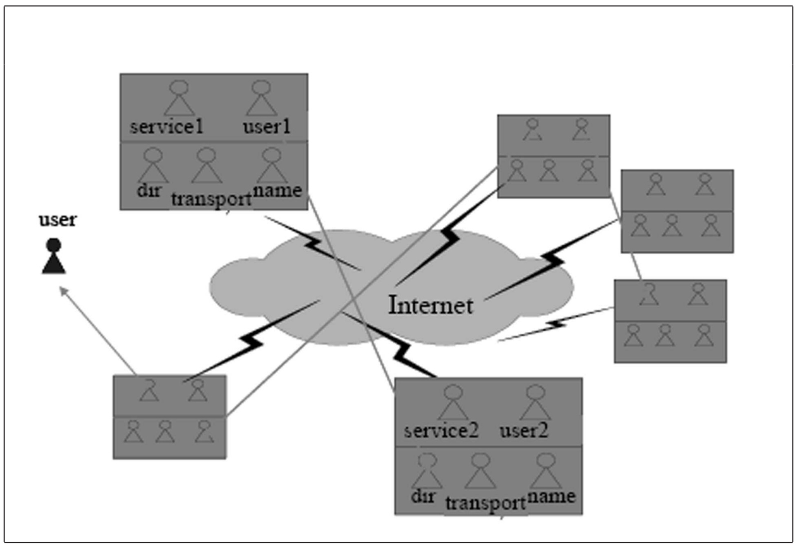


Figure 4.3. Multi agent system interactions (Source: Poslad et al. 2001)

For communication between agents executing on separate hosts, encryption is typically required for confidentiality protection. An agent can include the required functionality for encryption/decryption and its associated key management functionality. This might be a desirable solution for certain applications. Depending on the device, this might be implemented in software only, but may also be assisted by special hardware. Similarly, integrity functionality can be implemented within the agent "(Poslad et al. 2001)".

For communication between agents executing on the same host, confidentiality and integrity of communication are assumed to be provided by the platform. The platform needs to separate agents in such a way that information leakage and manipulation are prohibited. To further protect the communication between agents on the same host would not make sense, since the party in physical control of the host (who would be the only other potential threat) will have direct access to agents anyway. If further protection is required physical measures must be used "(Poslad et al. 2001)".

Seagent security considers to provide security of messages and authentication of data origin. So message alteration, message corruption and message reading by unauthorized parties is discouraged by Seagent end-to-end encryption mechanisms. In the test environment of Seagent, usage of the agents on the same host makes the security concept easier for the programmer for discarding the physical control.

4.1.2. Important Issues Against Security Attacks Through Agent Communication

As application communication becomes more semantic and as standard interaction becomes richer to support the increasing number of open service spaces to support dynamic service providers, much more finely grained asset model of communication is needed "(Poslad et al. 2003)".

A more finely grained model of agent communication can be viewed as a set of four layers that the issues should be considered with respect to providing security at each of these levels "(Poslad et al. 2003)":

- **Transport level issues:** There is already much existing work in the area of message transport between processes, especially in the context of client-server models. Therefore, sending messages between agents is not necessarily relegated entirely to some existing controllable transport, so existing transport-level security may not necessarily cover agent message-passing. For instance, agents may use forward messages through proxy agents. So it is not clear that relying entirely on existing transport-level security is desirable.

- **Communicative act issues:** The addition of new communicative acts to access the security service has the advantage of simplicity.For example, adding new speech acts as apply-certificate, issue-certificate, renew-certificate, update-certificate and revoke-certificate for certification process. This approach could have been adopted for agent management in the agent management specification. The disadvantage is that FIPA has resisted adding service or application specific speech acts, for example for security, in order to keep the core set of speech acts generic and to a minimum. Rather than introduce new speech acts, an ontological approach is introduced as a powerful alternative approach.

- **Ontology level issues:** Making use of the existing FIPA speech acts and interaction protocols but referencing one or more security ontologies would minimize the changes to the existing ACL specifications to support security. It may be beneficial if FIPA seeks to reuse existing security schema from the mainstream computer network community. However, as most security specifications are quite narrow, it is unlikely that a single security ontology could be specified.

- **Interaction protocol level issues:** Firstly, it has to be said that providing security at the level at the interaction protocol level is that conversations naturally provide session keys between agents. One natural paradigm is that an agent, wishing to interact with another agent in the context of some task, can authenticate itself to that agent; the agents can then share public keys that are valid for the duration of the interaction.

  A given security implementation may have the potential to influence the interaction protocols themselves. For instance, if authentication becomes a part of every interaction among FIPA agents, this could either become embedded in the interaction protocols themselves.

In Seagent design considerations, the interaction protocol level is important. According to developed interaction protocol, the security concerns of Seagent are realized successfully.

4.2. Challenges in Multi Agent System Security

There are many challenges that the agent community needs to overcome in order to develop security as shown below "(Poslad et al. 2001)":

1. Security is very complex, agent systems are just specialized distributed systems. Secure distributed systems can only be developed by, or have already been solved by, security experts - delegate security issues to them.
2. Security is part of the software infrastructure in which the agent platform is embedded and it is outside the scope of an agent architecture.
3. There is no benefit for security to be either monitored or controlled at the level of agents (at the application layer).
4. Some agents' systems do not need security. The early focus on the MAS community was on collaborative, rational, agent services within closed Intranets.

5. Security is domain and platform (implementation) specific - there is no general agent security architecture that is suitable for all applications and implementations. Hence, there is no reason for a general standard.

6. Complete specifications, and models of agent systems and agent security need to be completed before designing and building secure agent systems.

"These hypotheses are interrelated. At one level, the first four hypotheses all boil down the belief that security can be handled properly in the supporting infrastructure for agents rather than at the agent layer. To refuse or support these four hypotheses the similarities and differences between agent-systems and the application domains need to be understood in which they are being deployed and, conventional distributed systems and their associated domains.

The fifth and sixth hypotheses suggest an approach coupled with a top-down approach to develop secure agent systems. This may be useful but there are many agent systems already in operation. These could benefit from bottom-up approaches coupled to inductive development of agent security models "(Poslad et al. 2001)"."

In Seagent, the security concern is tried to be developed as a supporting infrastructure. The service layer and the mechanisms that are related to the software architecture has been decoupled from the agent layer. However, the Seagent security is developed after the agent infrastructure of Seagent. So the bottom-up approach is applied to the Seagent platform. This has been an advantage that the security mechanism has been tried on the deployed communication layer to be able to see the results of the security approach.

4.3. Security System Requirements For Agents

Security mechanisms, described in the only published FIPA security specification "(FIPA 1998)" aren't enough for the requirements at all. Then some additional requirements and dissimilarities from mentioned FIPA specification and outline of solution are discussed. The mostly discussed issues are shown below "(Novák et al. 2003)":

- It is possible to secure not only the whole ACL Message but only its part is required as well. Signing certain part of text or data for storing in database along with its signature and guarantee its authentication for later use (record of transactions) or encrypt only password required for access to particular resource, to allow subsequent detection of kind of requested data (e.g. from log file). This can be reached using the structure (class) containing not only carried text (signed or encrypted) but also additional information concerning security (type of security action, created signature, identification of used key). On the receiving side, this structure (class) is processed and original text obtained.

In Seagent, payload is encrypted and signed as well. There is an asymmetric encryption for only the session key is used, not the whole payload. But payload encryption is an important concern to be explained.

Asymmetric cryptography (also known as public key cryptography) is well suited for a mobile agent that needs to send back results to its owner or which collects information along its route before returning with its encrypted payload to its owner. This is due to the fact that the encryption key does not need to be kept secret. However, to encrypt very small messages is either very insecure or results in a large overhead compared with the original message. A solution called sliding encryption has been proposed that allows small amounts of data to be encrypted, and consequently added to the cryptogram, such that the length of the resulting ciphertext is minimized. Due to the nature of asymmetric cryptography the agent is not able to access its own encrypted payload until arriving at a trusted host where the corresponding decryption key is available.

- It is preferably not to bind security support tightly into the agent platform because agents can run on different platforms (without this type of security) or even without platform. For example agent collects data from appropriate server, running on different operating system, and provides data to MAS (in secure way).

- Security algorithms have to be concerned by security module automatically in secure communication. Negotiation about security algorithms can be very time-consuming on occasional connection. If agent sends over such connection message with security algorithm which recipient does not understand, recipient cannot inform the sender about it immediately.

- All private keys and other security related data have to be available to their owner only. Data have not to be accessible to anyone else (even the agent platform). Platform can be distributed across many computers and hence it is impossible to ensure security within the whole platform, if the private data are managed by platform. Every agent has to keep its private data secured, even during its migration on other platform.

4.4. FIPA Specified Security Concepts

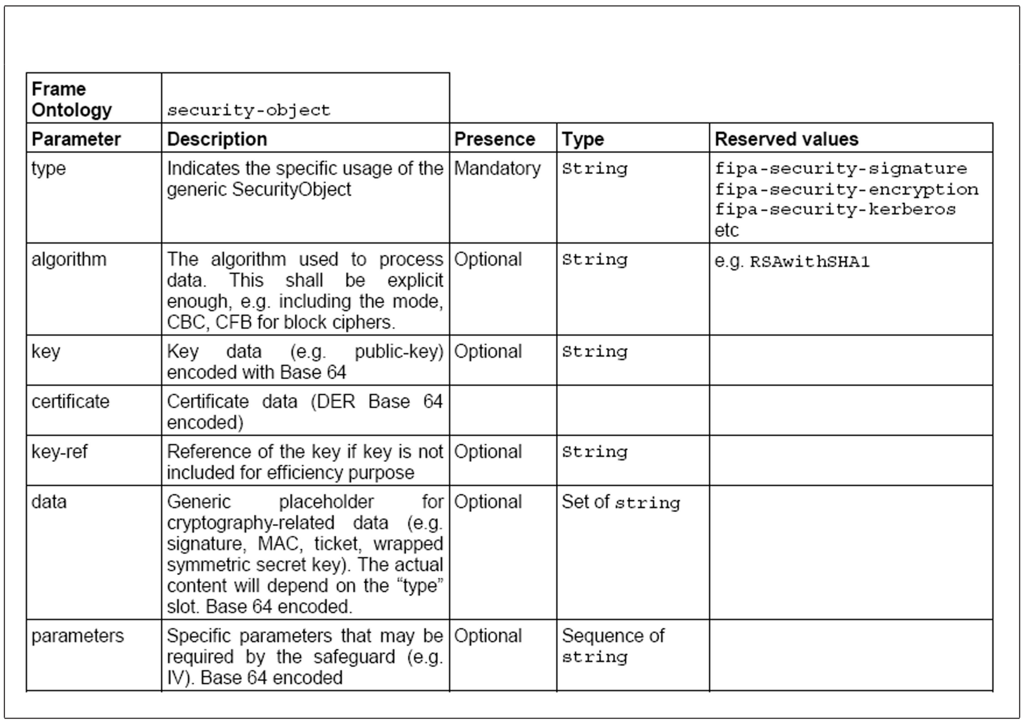
According to FIPA specification "(Vitaglione et al. 2003)", per message security means that each individual ACL message contains the security information required to process the embedded security safeguards. Agents are intended to process the security mechanisms themselves where appropriate so as to provided end-to-end (or peer-to-peer) security.

The security information required to process security safeguards needs to be transmitted within the FIPA TransportMessage. The SecurityObject introduces the generic placeholder for such information, just like the ReceivedObject already represents stamps placed by the MTS. For instance, if the message is signed, the SecurityObject contains the signature of the message.

In order to ensure integrity or confidentiality of an entire ACL Message, most safeguards need to apply only to the message payload, therefore SecurityObject is attached to the message Envelope. The SecurityObject can be included as user-defined slot into the envelope (e.g. X-Security), or, if standardized by FIPA, as an optional slot (e.g. Security). Furthermore, the slot containing the SecurityObject can contain a set of SecurityObjects, according to the different safeguards applied to a message, just as the envelope can already contain several ReceivedObjects "(Vitaglione et al. 2003)". Seagent security uses this strategy as adding Security slot to the message Envelope.

SecurityObject must include all the information required by the message receiver to perform message authentication and decrypt the payload. To be able to manage these properties, it has a FIPA specification that includes all necessary information as shown in Figure 4.4.

Signatures can be used in order to provide data integrity and data origin authentication as mentioned before. When communication relies on message exchange, each message can be signed by the sender in such a way that the receiver can verify the validity of the signature. The verification results allow the receiver of the message to securely evaluate the information integrity and the identity of the sender. The payload has not been modified between signature calculation and verification, therefore signatures can be applied to a FIPA ACL message by calculating the signature over the encoded payload. This allows protecting the information included into all ACL slots. The envelope's Security slot is used for the sending and receiving of this signature as shown in Figure 4.5.



Frame Ontology	`security-object`			
Parameter	**Description**	**Presence**	**Type**	**Reserved values**
type	Indicates the specific usage of the generic SecurityObject	Mandatory	`String`	`fipa-security-signature` `fipa-security-encryption` `fipa-security-kerberos` etc
algorithm	The algorithm used to process data. This shall be explicit enough, e.g. including the mode, CBC, CFB for block ciphers.	Optional	`String`	e.g. `RSAwithSHA1`
key	Key data (e.g. public-key) encoded with Base 64	Optional	`String`	
certificate	Certificate data (DER Base 64 encoded)			
key-ref	Reference of the key if key is not included for efficiency purpose	Optional	`String`	
data	Generic placeholder for cryptography-related data (e.g. signature, MAC, ticket, wrapped symmetric secret key). The actual content will depend on the "type" slot. Base 64 encoded.	Optional	Set of `string`	
parameters	Specific parameters that may be required by the safeguard (e.g. IV). Base 64 encoded	Optional	Sequence of `string`	

Figure 4.4. FIPA Security Object Specification (Source: Vitaglione et al. 2003)

The sender owns a cryptographic public/private key pair. He calculates the *signature = f(payload)*. *f* is a non invertible function, calculated by the asymmetrically encrypted(RSA) hash function(MD5) calculated over the payload. The encryption is processed by the sender's private key. The verification process consists in asymmetrically decrypt (using the senders public key) the message signature to have the hash of the payload at the source. Then the hash is calculated over the received payload. If the two hashes are equal, the integrity of payload is ensured as well as the origin of the message.

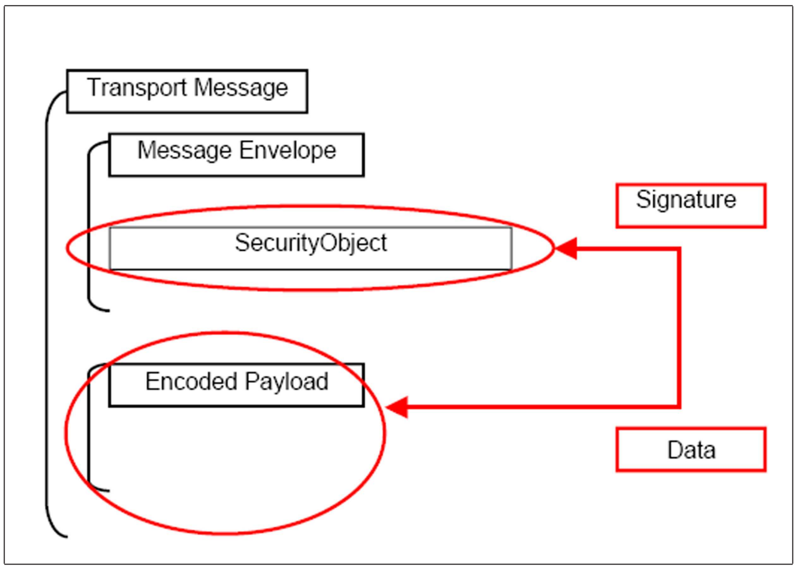


Figure 4.5. SecurityObject for the signature of a message (Source: Vitaglione et al. 2003)

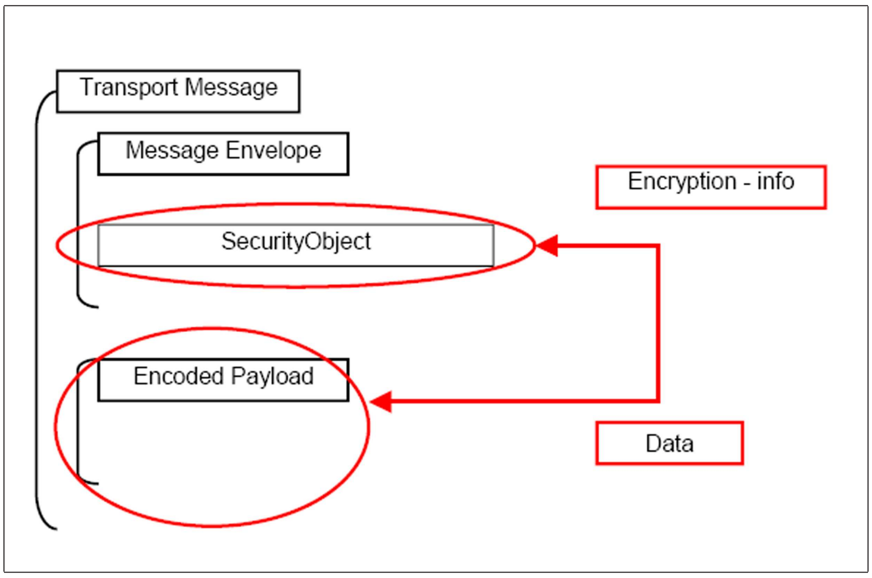


Figure 4.6. SecurityObject for the encryption information (Source: Vitaglione et al. 2003)

In the encryption scenario, the SecurityObject contains the information required by the receiver to decrypt the payload. There are different sub-scenarios, depending if the payload has been encrypted with a symmetric or asymmetric algorithm, if the symmetric secret-key is known by the receiver, etc.

4.5. Overview of Security Related Scenarios

It has been presented a series of simple security-related scenarios to illustrate some of the security issues "(FIPA 2001)":

- Publisher/directory scenario
- Courier/broker scenario
- Task allocation scenario

4.5.1. Publisher/Directory Scenario

Assume that there are 5 agents:

- User Agent wants to make a reservation for a hotel.
- Hotel Agent1 gives reservation service.
- Hotel Agent2 gives reservation service.
- Agent X is a malicious agent.
- FIPA DF agent acts as a directory/publisher for services by different vendors "(FIPA 2004)". All directory entries are in DF agent directories are public.

Hotel Agent1 and Hotel Agent2 register their services with a Directory Service, e.g., DF. Given the current FIPA specifications, Agent X can:

1. Pretend to be Hotel Agent2 and change service description of Hotel Agent2 at the DF, declaring that Hotel Agent2 now gives another service, not reservation service. In this way agent Hotel Agent2 will not be able to make any profit since it will be asked to provide a service that it is not able to support and nobody will ask it for the real service it is able to provide.

2. Pretend to be User Agent and places and reserves 10 rooms from Hotel Agent1. Therefore, Hotel Agent1 books 10 rooms. But there is not any customer that can use these rooms. Hotel1 will cost these rooms because of not using these rooms on that day.

3. Pretend to be Hotel Agent1 and book rooms although these rooms can be booked before. This situation may confuse the hotel's work plan.

4. Pretend to be a multitude of customers and overload Hotel Agent1 and Hotel Agent2 with fake offers to be processed resulting in valid service offers becoming delayed and perhaps causing them to be subsequently withdrawn.

5. Intercept and replay any message between Hotel Agent1, Hotel Agent2 and User Agent.

In this scenario, main security concerns are:

- Lack of authentication: an agent can masquerade as another agent(list items 1-3).
- Authorisation problems/lack of access of control:
  - a masquerading agent is authorized to change another agents published service description(list item 1).
  - any agent can also read any service provider agents entry in the DF(list item 1).
- Service Disruption: a masquerading agent can disrupt the normal service provision of that agent by making untrustable requests(list item 1).
- Denial of Service Attack(list items 2-4).
- Replay Attack(list item 5).

4.5.2. Courier/Broker Scenario

A broker is an agent that offers a set of communication facilitation services to other agents using some knowledge about the requirements and capabilities of those agents. A typical example of brokering is one in which an agent can request a broker to find one or more agents who can answer a query. The broker then determines a set of appropriate agents to which to forward the query, sends the query to those agents and relays their answers back to the original requestor. The use of brokerage agents can significantly simplify the task of interaction with agents in a multi-agent system. Additionally, brokering agents also enable a system to be adaptable and robust in dynamic situations, supporting scalability and security control at the brokering agent "(FIPA 2002c)".

According to above definitions,Seagent agent mechanisms are used below for this scenario. The main security concerns that are explained in Section 3.2. are shown in a life time scenario.

When it is assumed that there are three agents:

- User Agent
- Hotel Agent
- A courier agent (broker) agent acts as an intermediary for all messages between Hotel Agent and User Agent. This type of condition is also called as man in the middle attack in security concept.

If all discourse between User Agent and Hotel Agent is via the courier agent, the following security related problems could occur:

1. The courier agent can open up messages between User Agent and Hotel Agent and observe their contents.
2. The courier agent can modify the messages between User Agent and Hotel Agent.
3. The courier agent can insert messages from other parties to masquerade as User Agent or Hotel Agent.
4. User Agent can deny having sent a message, the courier can deny having been called to deliver it, Hotel Agent can deny having received a sent message.

In this scenario, main security concerns are:

- Lack of privacy: The courier agent can open up and see the contents of messages between User Agent and Hotel Agent(list item 1).

- Lack of integrity: The courier agent can misrepresent or corrupt messages between User Agent and Hotel Agent(list item 2).
- Lack of authentication: An agent can masquerade as another agent(list item 3).
- Repudiation: An agent can deny that an event such as a message transmission or message delivery has occurred(list item 4).

4.5.3. Task Allocation Scenario

In this scenario "(FIPA 2002d)", the manager agent wishes to have some task performed by one or more other agents, and further wishes to optimize a function that characterizes the task. This characteristic is commonly expressed as the price, in some domain specific way, but could also be soonest time to completion, fair distribution of tasks, etc.

The manager solicits proposals from other agents that specifies the task and any conditions the manager is placing upon the execution of the task. Agents receiving the call for proposals are viewed as potential contractors, and are able to generate proposals to perform the task. They may also be able to sub-contract the task to another contractors, behaving as a manager in this level of interaction. The contractors proposal includes the preconditions that the contractor is setting out for the task, which may be the price, time when the task will be done, etc. Alternatively, the contractor may refuse to propose. Once the manager receives back replies from all of the contractors, it evaluates the proposals and makes its choice of which agents will perform the task. One, several, or no agents may be chosen. The agents of the selected proposal(s) will be sent an acceptance message, the others will receive a notice of rejection. The proposals are assumed to be binding on the contractor, so that once the manager accepts the proposal the contractor acquires a commitment to perform the task. Once the contractor has completed the task, it sends a completion message to the manager.

This scenario is widely used in Seagent. By using the list of agents as given in Section 4.5.1. and Section 4.5.2., User Agent sends prices as call for proposals and Hotel Agents reply to these messages as agree or refuse communicative act messages according to their decision. The most suitable price is chosen between the agents and the interaction of messages end when User Agent obtains the necessary information for it. With the agree communicative act, User and Hotel Agent interaction goes on although rejection of an offer ends the conversation between these two agents.

When attacks through the interaction is determined between the user and the service agents, the following security related problems could occur "(FIPA 2001)". Although this list is not exhaustive it is not assured too much, they can be recognized as main concerns:

1. The rejected Hotel Agent can deny having received a rejection instead of an acceptance.
2. User Agent could divulge information to Hotel Agents that it does not want in turn divulged to sub-contractors.
3. User Agent cannot prevent a contractor from modifying any details of the contract when it is passed on to a subcontractor.

Although the second and third conditions aren't concerned in Seagent for now, the first one is an important issue. The main security concerns for all of these are:

- Non-repudiation: User Agent or Hotel Agent can deny that an event such as a message transmission or message delivery has occurred(line item 1).
- Lack of authority delegation: an agent such as a manager cannot control how second parties protect confidential information on to third parties (line item 2-3).

4.6. Security Interaction Protocols

The communication security mechanisms have to be considered further where in the FIPA architecture to be able to solve the security problem. So architectural issues and interaction protocols have to be covered by comparing the existing structures "(Borselius and Mitchell 2003)" and Seagent security security structure.

As shown in Figure 4.7, one way is to let the agent receive the message as usual, and when the agent has determined that it is an encrypted or signed message, it forwards the message to the appropriate service to decrypt it or to verify the signature. The agent would then be returned the decrypted message or an indication of the outcome of the signature verification. Similarly when the agent wants to send an encrypted message or a signed message it would send the message to the appropriate service and be returned the processed message, which now can be sent as usual.

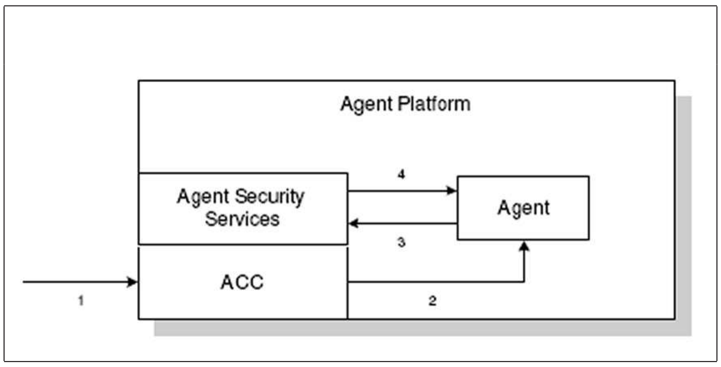


Figure 4.7. Security services separated from ACC (Source: Borselius and Mitchell 2003)

A second option, depicted in Figure 4.8, would be to let the ACC intercept the communication and other the security services in a more transparent way to the agent. This would require the ACC to be able to determine if incoming communication is encrypted or signed, as well as getting an indication from the agent whether encryption or signing should be done before sending the message.

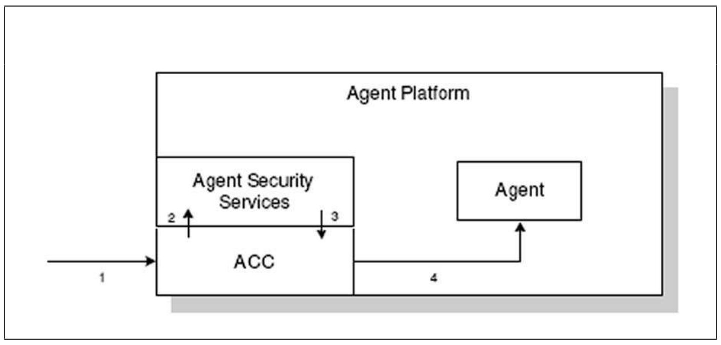


Figure 4.8. Security services transparent to agents (Source: Borselius and Mitchell 2003)

In Seagent, the message is sent to Seagent security service layer and the proper operations as encryption and digital signature are operated. Then the message with appropriate envelope properties are returned to the agent. If an agent firstly sends a message to other agent, envelope includes ”fipa-security-publickey”, a specialized property by Seagent, named SecurityObject. If the agents own each of the others public key, one of them generate the shared key and sends to the other. In this situation, the message envelope includes encrypted payload ”fipa-security-encryption” and ”fipa-security-signature” SecurityObject. If all of these operations have been made, the request only includes Envelope with ”fipa-security-signature” SecurityObject. This is basically the communication of agent and security infrastructure for Seagent.

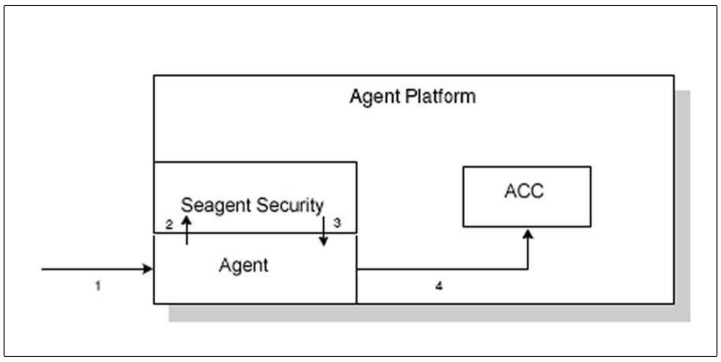


Figure 4.9. Secure communication for Seagent

4.7. Previous Work on Agent Communication Security

Through the way of working around agent security approaches, different strategies have been constructed by using basic security concepts and their interactions between agents and agent systems. With the extended security paradigms, the more secure multi-agent systems have been tried to be developed. Some of them are told below.

One approach proposes a security based agent for building a safeguard for a multi-agent system to protect its information, and to reasonably distribute tasks to cooperative objects to solve the issue of concurrent events ”(Huang et al. 2004)”. The main objectives for this kind of system approach are to overcome time and space constraints and to keep the balance between agents’ workload. In this approach, workload is especially tried to be balanced by resource management by counting the same task requests. If the number exceeds five, the system will add a record about the task information into the knowledge base. The next time the same task request happens, the knowledge base management module can directly search the result from the knowledge base. This approach fastens the knowledge base after security requests and responses between user and member of the MAS. These interactions for security employ strong identity authentication methodology, which applies cryptographic technology and a digital certificate to a system with role strategy, in order to overcome the problems of a login name/password and protect the system resources.

Another approach outlines a design for a secure mobile agent architecture termed MATRIX (Mobile Agent Technology Robust Infrastructure Exploration) "(Ghanea-Hercock and Gifford 2001)". Static agents on each host communicate each other by using mobile agents via XML formatted agent communication language. So a mobile agent authentication strategy is developed for reaching static agents. Additionally the encryption of messages between agent servers during communication, role-based origin authentication and encrypting collected data by agents are the main security concerns for MATRIX approach. This approach has also an authorization solution used by agent servers that the tasklists requested by the mobile agents are processed to prevent malicious agents can access services or take information from the agent server's services.

Another approach is especially a message-based approach. It uses Open PGP message structure in order to be able to evaluate its appropriateness for FIPA messages "(Borselius and Mitchell 2003)". The Open PGP protocol defines standard formats for encrypted messages, signatures, and certificates for exchanging public keys. PGP messages are constructed from a number of records referred to as packets. A packet is a piece of data that has a tag specifying its meaning. A PGP message consists of a number of packets. Some of those packets may themselves contain other packets. Each packet consists of a packet header, followed by the packet body. If there are reasons for treating the elements within the ACL differently, for example to only encrypt or sign certain elements, additional information would need to be provided to the agent to indicate how the message should be processed. To make use of the Open PGP message structure for FIPA agent communication, there has to be additional information to be added in the message envelope. For full extensibility, the ACL message should also carry information describing the security mechanisms applied to the message.

"(Borselius and Mitchell 2003)" approach is similar to the Security Object approach of FIPA. Different tagged packet types for different goals as Public Key Packet or Secret Key Packet substitutes the giving names of packets in the envelope's "type" field. This kind of approach has been developed near the time of Security Object approach. So this is a good comparative paradigm for the agent security developers.

CHAPTER 5

SEAGENT SECURITY

Seagent security is based on end-to-end message encryption and digital signature mechanisms. When it is developed, security concepts have been used that are told in the previous chapter.

It should be noted that security usually comes at a cost. Additional computing resources as well as communication resources are required by most solutions to achieve the mentioned security requirements in the previous chapter. So the solutions for a secure system should be optimized to provide the maximum performance for the system.

In Seagent, security mechanisms need to be able to at the purpose and nature of the communications of various conditions with differing security requirements. So while designing Seagent security, flexibility and modularity of the security subsystem is especially considered.

5.1. Design Issues of Seagent Security

FIPA agents operate within the context of FIPA agencies (called FIPA agent platforms) which provide typical generic support services, also called middleware services. FIPA agencies manage their life-cycles, enable them to provide and access services and supports communication with other agents in the same and in different agencies. Current agent middleware services, except for the communication services, are provided by two core FIPA middle agents (also called facilitator agents) called the Agent Management System (AMS) and the Directory Facilitator (DF) respectively.

All service user agents and service provider agents must register themselves with the AMS. This registration process allows the definition of a contract between any agent and the AMS in order to enable the AMS to manage their life-cycles.

There is current debate within FIPA as to whether these middleware services ought to continue to be accessed solely through service provider agents.This has been criticized as being both inefficient, e.g., an agent must send a forward message to an ACC to get it to send a message to the other agent residing in another agency, and perhaps too unmanageable to integrate or embed within a nonagent service infrastructure. Thus, in the current agency model, the ACC still exists as a communication service but it is no longer an agent, it is invoked via some internal API "(Poslad and Calisti 2000)".

According to the explained communication protocol, Seagent security addresses the challenges especially insecure communication. Security model manages corrupted messages by using a PKI handshaking protocol between the agents to verify validity of both parties. All messages for user agents, service provider agents and middleware services are encrypted according to Public Key Infrastructure(PKI). Although there is not a certification mechanism to be able to validate their identities, dispatching mechanism is able to cope with message security, only with digital signatures without a third party. All agents are held accountable for their actions because they have to sign all service queries and requests with their own private key. As there is a unique private key public key pair, once an agent signs a request, the agent can be held accountable. These concepts are explained in Section 5.3. with diagrams in detail.

Seagent security design is based on below Figure 5.1 primitively. The security concerns are implemented as cross-cutting concerns for communication and planning layers. The encrypted message is created in Planner(or Dispatcher) and sent to the other agent via communication channel(ACC). While these operations are occurred,only the service layer for the security scenario interacts with the agent or Planner. So the low-coupled and highly-cohesive software infrastructure is preserved.

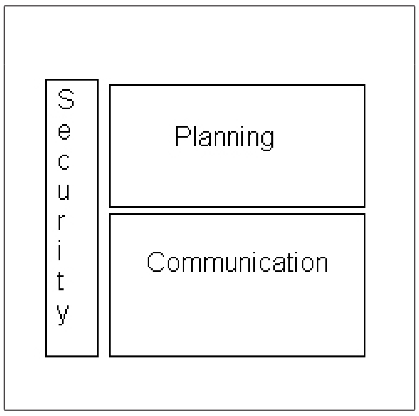


Figure 5.1. Seagent Security Model

5.2. Security Modules for Seagent

Seagent security mechanism is basically used on the FIPA's request-agree-inform communicative acts. So the modules for Seagent is told according to this scenario in this section.

To provide message security in Seagent, each agent has an AgentSecurityManager. AgentSecurityManager provides the agents to hold their private/public key pair, session keys used with the other agents, public keys of the agents that are communicated together and the states of these public keys that the agent sends its public key to other agents or receives public key from the other agent. These informations are used in secure communication between agents. SeagentSecurityService is the service layer of the Seagent security structure. It is an encapsulation for encryption, signing, decryption, verifying and public key operations. Agents and Dispatcher use the SeagentSecurityService to make the appropriate operations to be able to provide a secure message.

As shown in Figure 5.2, firstly each agent generates their public/private key pair to provide the secure conversation. In the beginning of the conversation between agents, sender agent checks if it has sent its public key to other agent. If it has not sent before, it adds the public key to the transport message with the help of SeagentSecurityService. Then sender agent updates its public key map as it has sent the public key to the receiver agent. Then sender agent sends its ACL message with public key to receiver through Dispatcher. In this conversation, receiver agent makes the similar operations by using the transport message. It updates its public key map and public key state map to remember that the receiving message from each of these agents to each other is encrypted in the future conversation steps and the message structure is processed according to this information.

As shown in Figure 5.3, for the second message between agents or agree message, sender agent asks if it has the pair public key of the communicated agent and it sends its public key to the receiver agent. If pairPublicKeySendState is not true, it sets the pairPublicKeySendState to true for understanding sending public key to other agent. However, SeagentSecurityManager determines the agent to add its public key to transport message. Then the sender agent looks up if there is a shared key generated before. If there is not, it generates and adds it to its AgentSecurityManager. Then sender agent encrypts the session key asymmetrically to be able share with the receiver agent. Lastly, the ACL message is encrypted symmetrically and signed with this session key. The encrypted ACL message, envelope with public key, encrypted session key and signature object is sent to the pair agent. While the session key is encrypted with asymmetric encryption, the ACL message uses symmetric encryption because of the slowness of the asymmetrical encryption. Encrypting whole message with asymmetric encryption is a huge amount of time. However, encrypting only the session key is a one time operation. The session key is used along the conversation to be able to symmetrically encrypt the ACL message and this improves the communication performance.

As shown in Figure 5.4, the receiver side takes this encrypted message and firstly decrypts the session key with its private key obtained from its AgentSecurityManager. Then it adds session key to its sharedKeyMap and updates sharedKeyStateMap according to setting receiveState of the sender agent as true. Then the agent decrypts the ACL message with this session key and decrypts the digital signature with the pair public key to be able to own hash. Then it hashes the decrypted ACL message to compare two hashes. If two of them are equal, it understands that this ACL message from the expected sender and the message has not been tampered during conversation. So decryption and verification steps are processed successfully.

As shown in Figure 5.5, sender agent sends the transport message through Dispatcher to receiver agent. Dispatcher generates the encrypted and signed message. Then it obtains the session key of the conversation and the public key of the receiver agent by using AgentSecurityManager of the sender. Then it sends the message to the receiver agent. Receiver agent takes the message and processes the reverse of the above operations. Using receiver agent's own private key and session key from its AgentSecurityManager, decryption and verification of the message is processed and the agent achieves to make operations of the decrypted and verified ACL message.

After each of the agents have their public keys and session keys, they can easily encrypt and sign messages. So in the third and the latter messages or the inform message as told in the previous paragraph, messages are encrypted and signed properly. They are sent with encrypted ACL message and the Signature SecurityObject in the Envelope and the decryption process is made as told before.

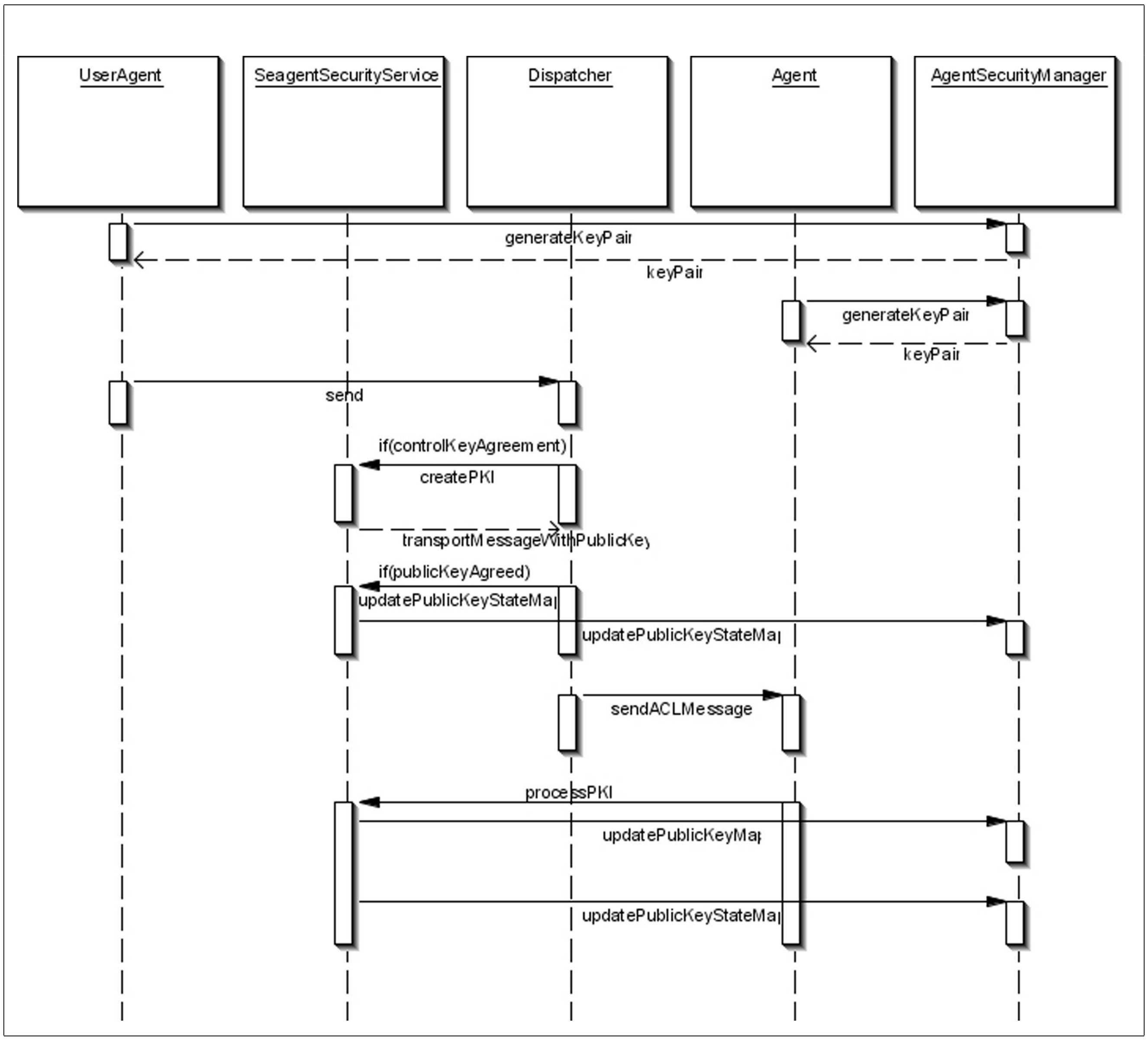


Figure 5.2. Sending public key in Seagent(Request)

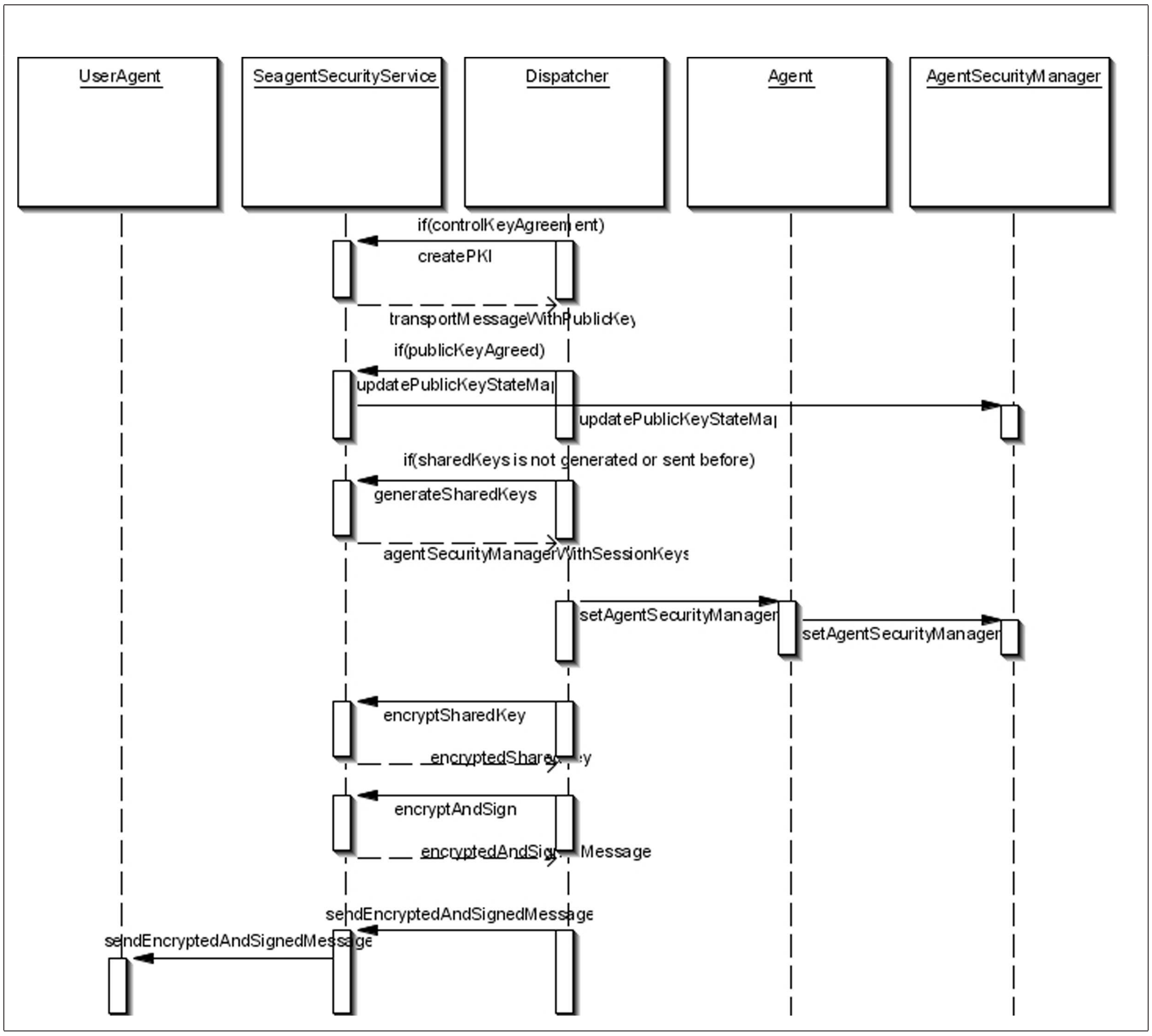


Figure 5.3. Sending public and session key with encrypting ACL message (Agree)

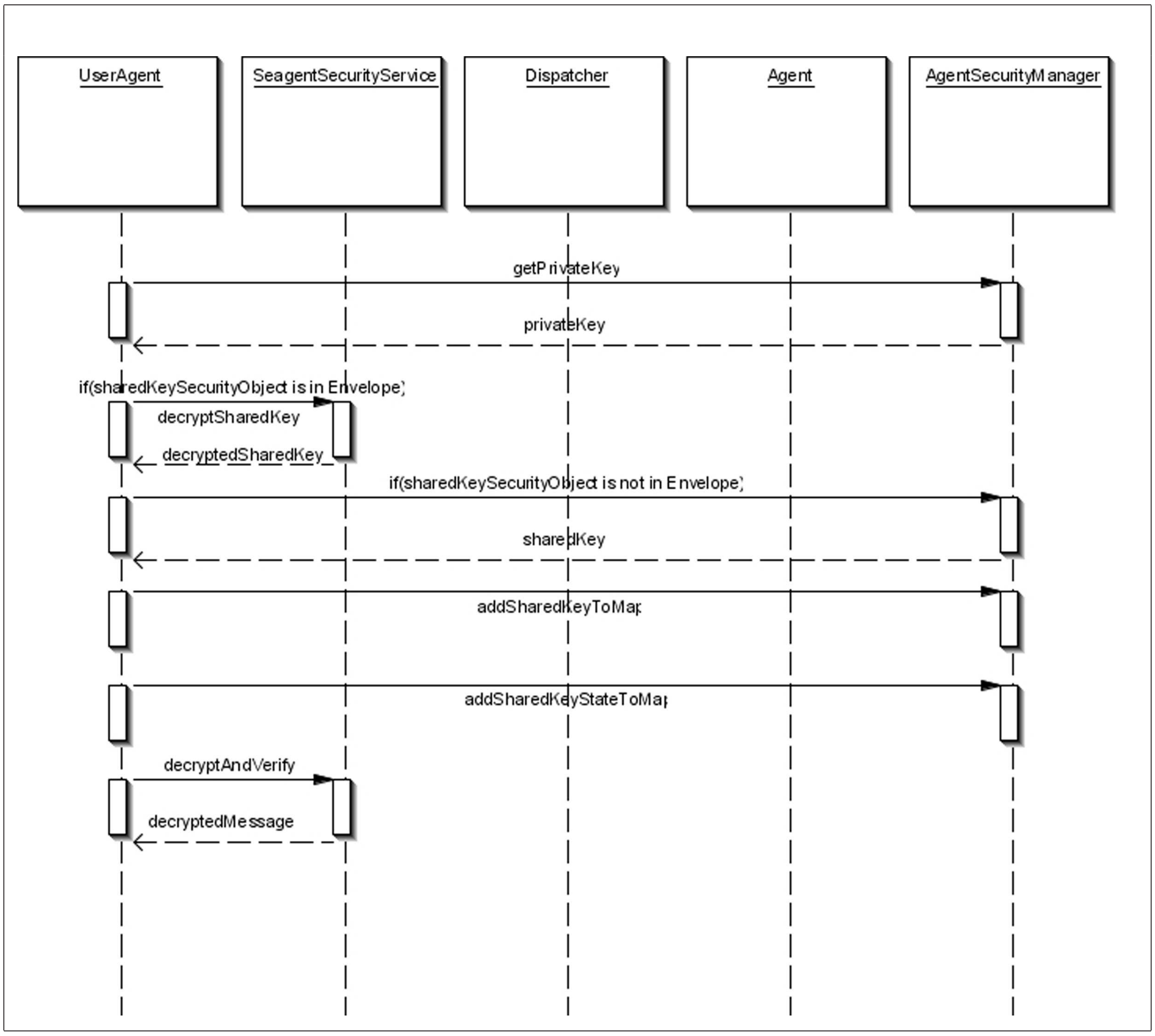


Figure 5.4. Decryption of session key and message in Seagent

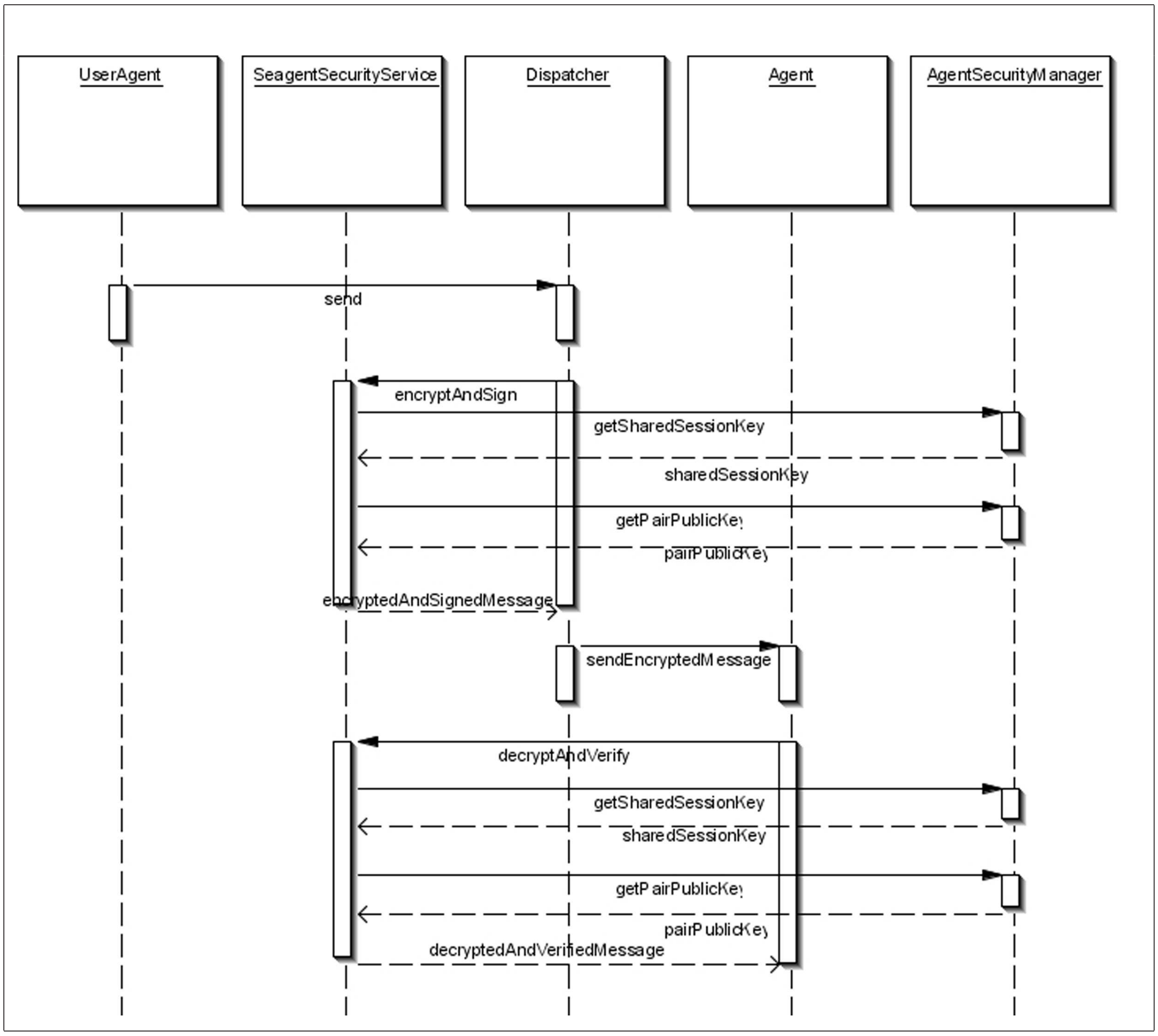


Figure 5.5. Steps of conversation after sharing session and public keys

5.3. Security Design and Implementation for Seagent

Seagent security is basically called by SeagentSecurityService class that implements an interface as shown in Figure 5.6. SeagentSecurityService is used as the communicator of the agents, planner and Seagent security classes. The methods that are called by Seagent security service is processed in SecurityFacade. SecurityFacade is used to simplify access of SeagentSecurityService and SecurityMechanisms to decouple subsystems and provide the granularity of security issues. SecurityFacade can access AgentSecurity-Manager to reach agent's security properties as public key, private key or session key. By taking these necessary security components, SecurityFacade execute like a bridge to encrypt and decrypt session key, encrypt and decrypt payload, sign message etc. This kind of approach provides the SeagentSecurityService only what to execute, not how to execute. Method infrastructure that will be processed by mechanisms are called by SecurityFacade and the result is used by SeagentSecurityService.

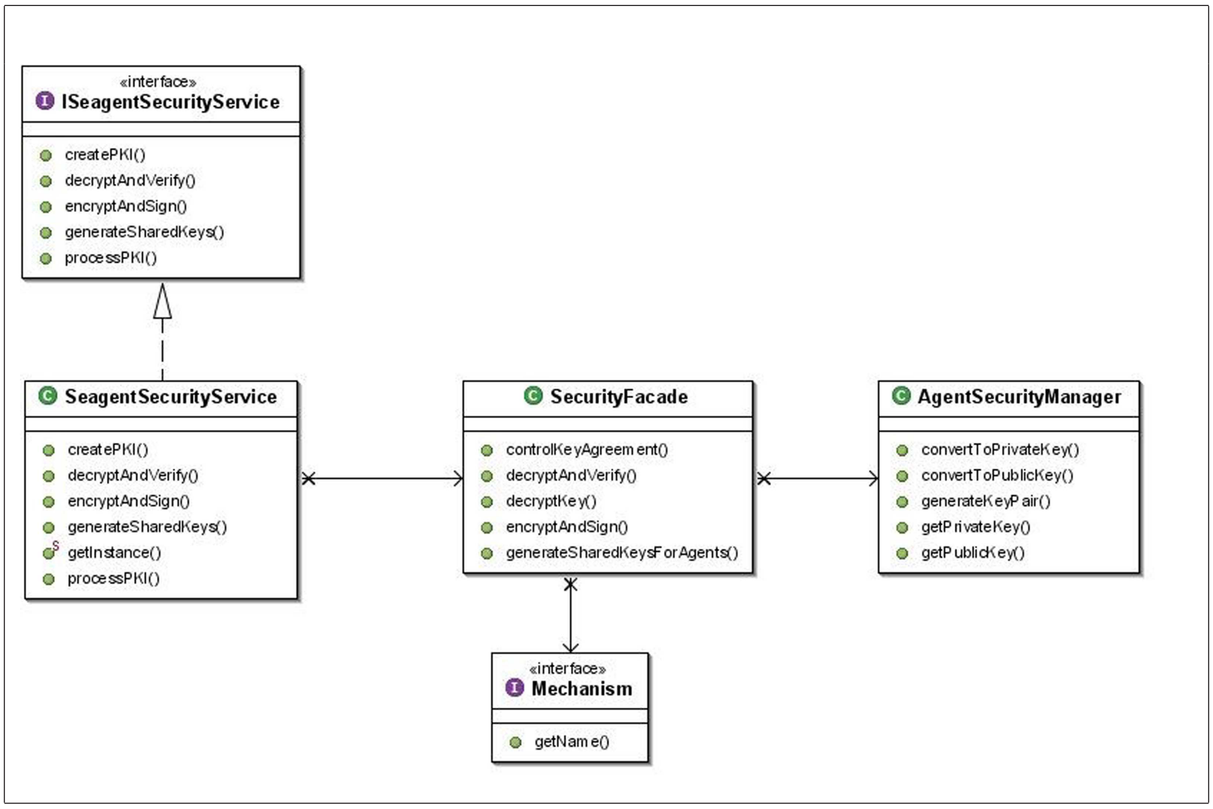


Figure 5.6. Seagent security class diagram

AgentSecurityManager properties taken from SecurityFacade is used for parameters passed to the Mechanisms(Confidentiality Mechanism, Key Agreement Mechanism and Signature Mechanism). According to the sequence of scenario, the methods of these classes are called and suitable operations are done for encryption, key agreement and signature as shown in Figure 5.7. ConfidentialityMechanism is used for encryption and decryption of payload and the key. KeyAgreementMechanism is used for controlling pair public key of each of the agents. KeyAgreementMechanism controls that the agent has sent the public key, has received the pair public key or generated and shared the session key. SignatureMechanism signs and verifies the payload of the transport message.

After asymmetrically encrypting session key, symmetrically encrypting payload or generating signature, they have to be hold in the FIPA SecurityObject. Three different types for three different mechanisms as "fipa-security-publickey", "fipa-security-encryption" and "fipa-security-signature" are generated with different parameters. Although "fipa-security-encryption" and "fipa-security-signature" are the standard FIPA reserved words, "fipa-security-publickey" is not a reserved word. Because there is not a known reserved word for public key agreement. According to design phase, instead of creating different constructors for each of these three objects, Builder design pattern has been used for different instances of SecurityObjects as shown in Figure 5.8. The main purpose of this design approach is to simplify complex object creation by defining a class whose purpose is to build instances of another class. So SecurityObjectManager is used to create a SecurityObject according to the type of the mechanism with different parameters of the SecurityObject.

Key generation is another important concern. For now, 3DES for symmetric key generation and RSA for asymmetric key generation is used in Seagent as shown in 5.9. However, the abstract SymmetricKeyGenerator and AsymmetricKeyGenerator classes are able to be extended for different algorithms by using a factory class for choosing the algorithm and this algorithm is agreed on the agents with the SecurityObject's "algorithm" field.

Seagent security also encodes session keys with Base64. Encoding of the keys in the SecurityObject is a FIPA specified SecurityObject property. Base64 encoding scheme is used for keys as specified in the spec. Base64 encoding scheme encodes the messages with 6 bit blocks and $2^6 = 64$ different characters from a-z, A-Z and 0-9 for the first 62 digits but the symbols chosen for the last two digits vary considerably between different systems, especially for padding. By encoding key, receiver side decodes the session key after decoding it and takes the usable session key. This transformation provides encapsulation of session key between agents.

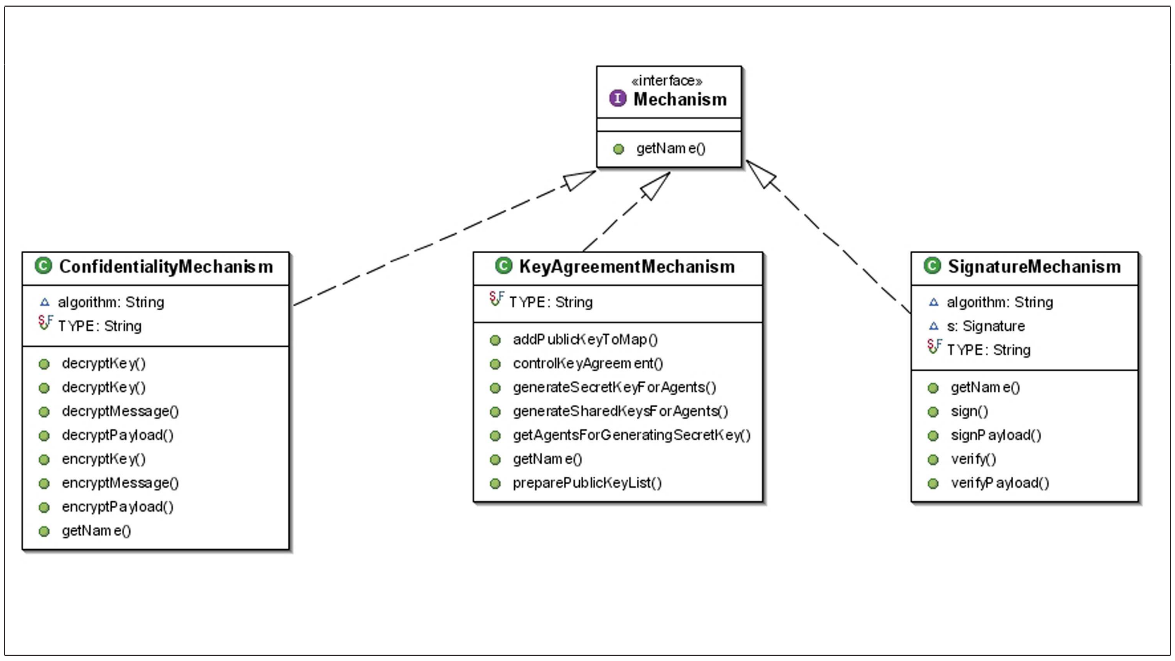


Figure 5.7. Seagent mechanisms class diagram

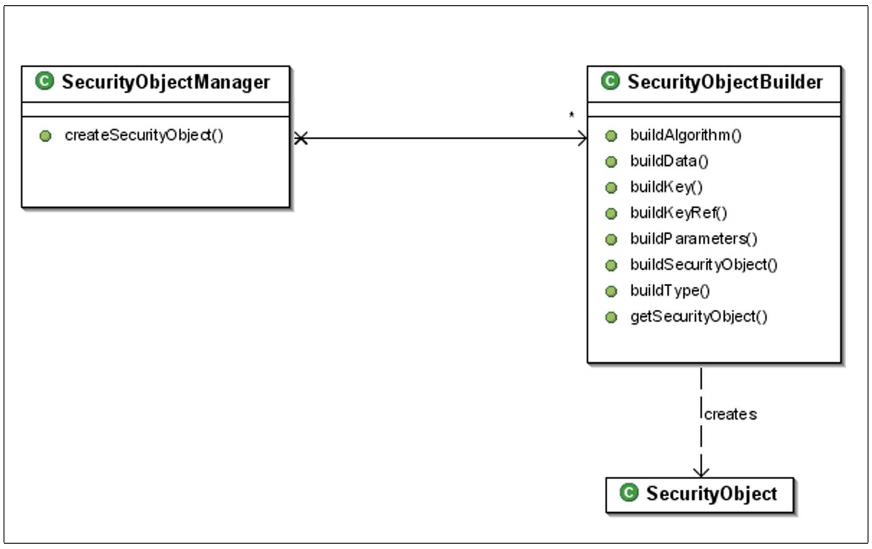


Figure 5.8. Seagent SecurityObject generation

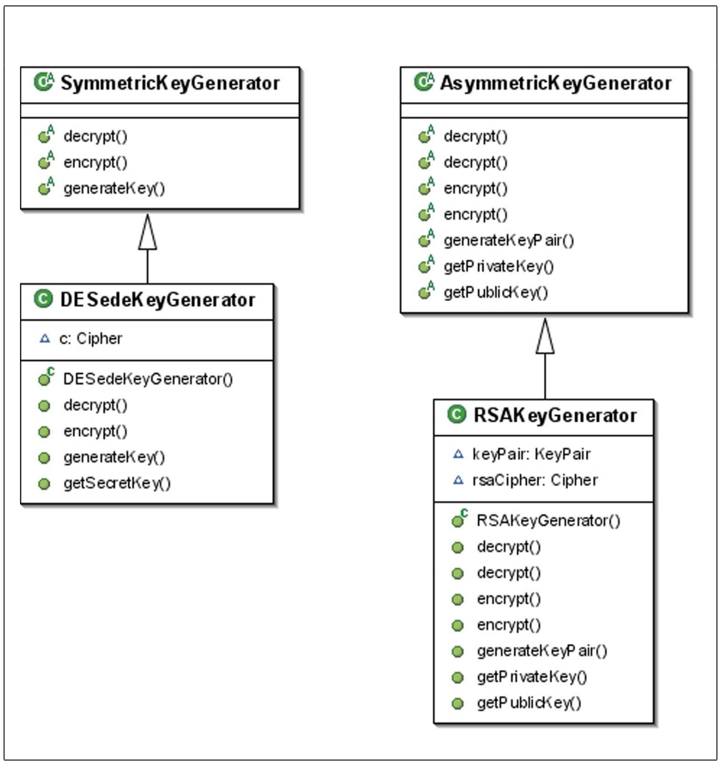


Figure 5.9. Seagent Key Generation

5.4. Experimental Results

In agent communication protocols for making a secure communication, the agents have to realize some protocol requirements according to communicative acts. The agent interaction is showed in Section 5.2. with detailed sequence diagrams.

In this implementation, the overheads for the communication is measured by testing in one host with Windows XP and JDK1.5.

In experimental analysis, firstly the unencrypted message communication is measured. According to this measure, this measurement is compared with the encryption scheme. But the encryption scheme has different steps. So each step time is individually calculated and the values have been obtained according to cost analysis. The values are shown in Table 5.1.

As shown in the Table 5.1, when the message content size increases, the communication time also increases. However, communication overhead is considerably limited in comparison to the overhead in security preparation.

Unencrypted message with public key in the envelope scheme in the Table 5.1 is only considered in the beginning of the conversation. Encrypted message with signature and session key in the envelope scheme is considered after one agent generates session key and wants to share it with the other agent. Especially the encrypted message with signature in the envelope and message without encryption lines of the table have to be compared because of the conversation's mostly used communication issues. The nearly 3 times more time for secure communication cost can be an acceptable or unacceptable result according to the goal of the system.

With the implementation of Seagent security with Java, the implementation mechanism scenarios are also tested and the results are shown in Table 5.2.

The table 5.2 shows that the key pair generation time slows down the system, but only in the beginning of the agent process time. Then the encryption and decryption process affects the system less than the key pair generation time.

Encryption and decryption processes are measured within the working time of the system by the subtraction of the end and start times of the process. These times are not equal for all messages. The experimental results are the average of the all message encryption and decryption processes.

These results can change with the communication traffic flow, host performance and algorithm preferences. In these results, 3DES as the symmetric encryption algorithm and RSA as the asymmetric encryption algorithm is used. The RSA implementation is provided by Bouncy Castle Provider because of the deficiency of Java Cryprography Extension(JCE). With the encryption algorithms' choice, the encrypted message size can also change and the experiments can give different results.

Table 5.1. Seagent communication experimental results(milliseconds).

Communication Experimental Results	**Time(ms)**
Message without encryption	47
Encrypted message with signature in the envelope	133
Unencrypted message with public key in the envelope	281
Encrypted message with signature and session key in the envelope	359

Table 5.2. Seagent security experimental results(milliseconds).

Security Experimental Results	**Time(ms)**
Key Pair Generation Time	12737
Session Key Generation Time	47
Encrypt and Sign Time	223
Decrypt and Verify Time	145

CHAPTER 6

CONCLUSION AND FUTURE WORK

Agents have many definitions and cover both software and artificial intelligence (AI) agents, being either mobile or intelligent agents. In this thesis, it is focused only on software agents and in particular the FIPA agents. The multi-agent system domain covers software agents, for which FIPA is a part of that is further based on the intelligent agent domain.

Demands by agents about security help giving agents a way to reason about their security-related capabilities and to adjust their own composition accordingly. So this demand causes developing agent based security solutions. Especially the interaction of agents and users increases this requirement in timely manner.

There are still ongoing projects that target the development, standardization and testing of the FIPA agent platform. There are several challenges related to these issues and the fact that building and fielding a MAS-based agent system. There are few systems available for use in the real world, which also contribute to the fact that little effort has been devoted to security issues in agent systems. The traditionally approach towards security and agents is to use encryption and to focus the research on adapting cryptography techniques into agent systems. Since FIPA agent systems platforms operate in an open environment, such as the Internet, communication links, information sources, services and agents can appear and disappear dynamically. This opens for attacks by unauthorised parties, such as hackers or malicious agents being able to spy on other agents (eavesdropping), steal or damage information or the infrastructure of components (manipulation), and preventing access for authorised parties (denial of service).

In this study, only the message security solutions for agents have been developed. The security issues have been told by using FIPA specified concepts. It is shown how conventional security protocols can be used to provide secure communication within an agent-based system. Data integrity and data origin authentication can all be provided for agent interactions.

Seagent security now only makes the message security using RSA, MD5 and 3DES algorithms. These mechanisms could be extended and an authentication protocol can be added to the system. These features can be provided by an agent and interaction schemes can be implemented.

Authorization of different agents could be made by agents role and competencies to a certification authority. So that in a complex AMS and DF, there could be an authorization solution. When a search operation is used to discover an agent by a particular entity, the agent services can be used either to constrain the search to an agent which supports Certificate Authority services. This facility, of course, requires that agents provide the search parameters of interest when they register with the DF. The DF should not be used to store sensitive information. For example, the DF can store certificates, but should not store private keys as all information in the DF is made public.

In conclusion, the first release of the Seagent security solution includes the all appropriate functionality for FIPA security concept. With the usage of system, the advantages and disadvantages of the security system can be seen better. However, new security modules can be added to the system.

REFERENCES

F. Bellifemine, A. Poggi, and G. Rimassa. *JADE: A FIPA2000 compliant agent development environment. Agents Fifth International Conference on Autonomous Agents*, pages 216–217, 2001.

A. H. Bond and L. Gasser. Readings in Distributed Artificial Intelligence. Morgan Kaufmann Publishers, San Mateo,CA, USA, 1988. ISBN 093461363X.

N. Borselius. *Security in Multi-Agent Systems.* In *Proceedings of the 2002 International Conference on Security and Management (SAM'02)*, pages 31–36. CSREA Press, 2002.

N. Borselius and C. J. Mitchell. *Securing FIPA Agent Communication.* In *Security and Management*, pages 135–141, 2003.

O. Dikenelli, R. C. Erdur, O. Gumus, E. E. Ekinci, Ö. Gürcan, G. Kardas, I. Seylan, and A. M. Tiryaki. *SEAGENT: A Platform for Developing Semantic Web Based Multi Agent Systems. Autonomous Agents and Multi-Agent Systems(AAMAS) 05*, pages 1271–1272, 2005.

R. Fikes, P. Hayes, and I. Horrocks. *OWL-QL: A language for deductive query answering on the semantic web.* Technical Report KSL 03-14, Stanford University, Stanford, CA, 2003.

FIPA. *FIPA Agent Management Specification.* Technical Report SC00023K, Geneva, Switzerland, March 2004.

FIPA. *FIPA Abstract Architecture Specification.* Technical Report SC00001L, Geneva, Switzerland, December 2002a.

FIPA. *FIPA ACL Message Structure Specification.* Technical Report SC00061G, Geneva, Switzerland, November 2002b.

FIPA. *FIPA Brokering Interaction Protocol Specification.* Technical Report SC00033H, Geneva, Switzerland, December 2002c.

FIPA. *FIPA Contract Net Interaction Protocol Specification.* Technical Report SC00029H, Geneva, Switzerland, December 2002d.

FIPA. *FIPA Security SIG Request For Information.* Technical Report f-out-00065, Geneva, Switzerland, February 2001.

FIPA. *FIPA Agent Security Management Specification.* Technical Report OC00020A, Geneva, Switzerland, October 1998.

FIPA. *FIPA Agent Message Transport Service Specification.* Technical Report SC00067F, Geneva, Switzerland, December 2002e.

S. Franklin and A. Graesser. *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents.* In *A Taxonomy for Autonomous Agents.* Springer-Verlag, 1996.

M. R. Genesereth and S. P. Ketchpel. Software agents. *Communications of the ACM*, 37 (7):48–53, 1994. ISSN 0001-0782.

R. Ghanea-Hercock and I. Gifford. *Solutions to security in mobile agent systems.* In *Mobile Agents - Where Are They Going? (Ref. No. 2001/150)*, pages 5/1–5/4. IEEE, 2001.

D. Gilbert. *Intelligent Agents White Paper.* Technical report, 1996.

J. Gordon. *Introduction to Cryptography.* Concept Laboratories Ltd, Hertfordshire SG3 6TL England, 1998.

J. R. Graham, K. S. Decker, and M. Mersic. *DECAF - A Flexible Multi Agent System Architecture. Autonomous Agents and Multi-Agent Systems(AAMAS) 03*, 7(1-2):7–27, 2003.

D. Greenwood, N. Lhuillier, S. Poslad, A. Reitbauer, S. Robles, J.J. Tan, G. Vitaglione, F. Bellifemine, G. Caire, and M. Calisti. *FIPA Agent Envelope Security Proposal.* Technical report, Geneva, Switzerland, December 2003.

S. H. Houmb. *Security Issues in FIPA Agents.* pages 4–6. The Norwegian University of Science and Technology (NTNU), 2002.

J. Huang, Z.D. Zhou, and Y.P. Chen. A security-based agent for a virtual enterprise. *The International Journal of Advanced Manufacturing Technology*, 23(7-8):620–625, 2004. ISSN 0268-3768.

N. R. Jennings, K. Sycara, and M. Wooldridge. *A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

J. Knudsen. *Java Cryptography.* O'Reilly, 1998. ISBN 1-56592402-9.

A. J. Menezes, P. C. V. Ooorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

P. Novák, M. Rollo, J. Hodík, and T. Vlcek. *Communication Security in Multi-Agent Systems.* In *CEEMAS*, volume 7, pages 454–463. Springer, 2003.

S. Poslad and M. Calisti. *Towards Improved Trust and Security in FIPA Agent Platforms. Autonomous Agents 2000 Workshop on Deception*, June 2000.

S. Poslad, P. Charlton, and M. Calisti. *Protecting What Your Agent Is Doing. AgentLink Agent Technology Conference RA5 Overview / Review Articles*, 7:7–11, 2001.

S. Poslad, P. Charlton, and M. Calisti. *Specifying Standard Security Mechanisms in Multi-Agent Systems. Trust, Reputation and Security: Theories and Practice*, 2003.

Y. Shoham. *Agent-Oriented Programming. Artificial Intelligence*, 60:51–92, 1993.

W. Stallings. *Network Security Essentials, Applications and Standards.* Prentice Hall, Upper Saddle River, New Jersey, 2003. ISBN 0-13-035128-8.

D. Stinson. *Cryptography: Theory and Practice.* CRC Press, 1995. ISBN 0849385210.

K. Sycara, M. Paolucci, M. V. Velsen, and J. A. Giampapa. *The RETSINA MAS Infrastructure. Autonomous Agents and Multi-Agent Systems*, 7(1/2):29–48, July 2003.

G. Vitaglione, N. Lhuillier, and D. Greenwood. *FIPA Agent Message Security Object Proposal.* Technical Report f-in-00095, Geneva, Switzerland, November 2003.

APPENDIX A

TERMS AND DEFINITIONS

3DES: Abbreviation for Tripple DES. Strengthened but even slower version of DESwith effectively its original key size trippled.

Action: A basic construct which represents some activity which an agent may perform. A special class of actions is the communicative acts.

Agent: An agent is the fundamental actor in a domain. It combines one or more service capabilities into a unified and integrated execution model which can include access to external software, human users and communication facilities.

Agent Communication Language (ACL): A language with precisely defined syntax, semantics and pragmatics that is the basis of communication between independently designed and developed software agents.

Agent Communication Channel (ACC): The Agent Communication Channel(ACC) is used for information provided by the Agent Management System to route messages between agents within the platform and to agents resident on other platforms.

Agent Management System (AMS): The Agent Management System is an agent which manages the creation, deletion, suspension, resumption, authentication and migration of agents on the agent platform and provides a "white pages" directory service for all agents resident on an agent platform. It stores the mapping between globally unique agent names and local transport addresses used by the platform.

Agent Platform: An agent platform provides an infrastructure in which agents can be deployed. An agent must be registered on a platform in order to interact with other agents on that platform or indeed other platforms. An AP consists of three capability sets ACC, AMS and default Directory Facilitator(DF).

Asymmetric Cipher: Encryption where the key to be used to decrypt the ciphertext is different from the encryption key. Both keys are correlated, but it is a "hard" problem to derive one from the other.

Attack: Exploiting of vulnerability.

Authentication: It is the process of ensuring, that in a communication protocol a presented identity is impersonated rightly by the presenting entity. Further, an authenticated communication includes ensuring integrity of the data exchanged.

Cipher: Class of functions for encrypting and decrypting. Ciphers deal with the characters of a message, at the syntactical instead of the semantical level as does a code.

Ciphertext: The result of applying a cipher function to a plaintext, the encrypted output.

Communicative Act: A special class of actions that correspond to the basic building blocks of dialogue between agents. A communicative act has a well-defined, declarative meaning independent of the content of any given act.

Confidentiality: The property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

Content: One part of a communicative act which represents the domain dependent component of the communication. Note that "the content of a message" does not refer to "everything within the message, including the delimiters", as it does in some languages, but rather specifically to the domain specific component. In the ACL semantic model, a content expression may be composed from propositions or actions.

Content Language: The content of a FIPA message refers to whatever the communicative act applies to. If, in general terms, the communicative act is considered as a sentence, the content is the grammatical object of the sentence.

Conversation: An ongoing sequence of communicative acts exchanged between two (or more) agents relating to some ongoing topic of discourse. A conversation may (perhaps implicitly) accumulate context that is used to determine the meaning of later messages in the conversation.

Cryptanalysis: Trying to break the security of a message.

Cryptanalyst: Person working on the field of cryptanalysis.

Cryptographer: Person working on the field of cryptography.

Cryptography: The science of keeping messages secure.

Cryptologist: Person working on the field of cryptology.

Cryptology: The sum of cryptanalysis and cryptography.

Deciphering: Synonymous but less common word for decryption.

Decryption: The opposite of encryption.

DES: Abbreviation for Data Encryption Standard. A specific block cipher.

Directory Facilitator (DF): The Directory Facilitator is an agent that provides a "yellow pages" directory service for the agents. It stores descriptions of the agents and the services they offer.

Encryption: Hiding the contents of a communicated message, so that unauthorized parties, that dont have access to a certain secret, i.e. the key, are not able to semantically understand the contents. The reverse operation of restoring the original contents is called decryption.

FIPA: Abbreviation for Foundation for Intelligent Physical Agents.

Hash: Cryptographic checksum of fixed length, that is easily calculated but where it is a "hard" problem to find either the original value or another value with the same checksum.

Integrity: Property of a communication, ensuring that no one is able to modify the communicated data without this change being noticed.

JCA: Abbreviation for Java Cryptography Architecture. Classes of Java related to cryptography.

Key: Input data for encryption and decryption functions. In symmetric algorithms the key has to be a pre-established shared secret between the communicating partners. In an asymmetric protocol, the secret key is held by only one party, the public key is, as the name suggests, public.

Man-in-the-Middle: An attacker, located between two communicating parties in the data path of a network.

Message: An individual unit of communication between two or more agents. A message corresponds to a communicative act, in the sense that a message encodes the communicative act for reliable transmission between agents.

Message Transport Service (MTS): The message transport service is an abstract service provided by the agent management platform to which the agent is currently attached. The message transport service provides for the reliable and timely delivery of messages to their destination agents, and also provides a mapping from agent logical names to physical transport addresses.

Mobile Agent: An agent that is not reliant upon the agent platform where it began executing and can subsequently transport itself between agent platforms.

Non-repudiation: The impossibility for the signer of a message to later deny, that he signed the document.

Ontology: An ontology is an explicit specification of the structure of a certain domain (e.g. e-commerce, sport). For the practical goals of FIPA (that is enabling development and deployment of inter-operable agent-based applications), this includes a vocabulary (i.e. a list of logical constants and predicate symbols) for referring to the subject area, and a set of logical statements expressing the constraints existing in the domain and restricting the interpretation of the vocabulary. Ontologies therefore provide a vocabulary for representing and communicating knowledge about some topic and a set of relationships and properties that hold for the entities denoted by that vocabulary.

Plaintext: The message to be hidden by applying a cipher function to it. Also the result of correctly applying a deciphering function to a ciphertext.

RSA Cipher: Abbreviation for Rivest-Shamir-Adleman Cipher. Probably the best known and analyzed asymmetric cipher , named after its inventors, based on the "hard" problem of factoring large numbers. Widely in use, but incumbered by patent issues until recently.

Signature: A hash over a document, performed with a secret key, that can be verified with the corresponding public key.

Stream Cipher: Encryption function, that works on only one bit of data at a time.

Symmetric Cipher: Encryption, where the key to be used to decrypt the ciphertext is the same as was used for encryption.

Threat: Circumstances that have the potential to cause loss or harm to the computing system.

User Agent: An agent which interacts with a human user.

Vulnerability: A weakness in the security system that might be exploited to cause loss or harm.