University of Khartoum

Post Graduate Faculty

Faculty of Mathematical Sciences

SMAG System

A secure Mobile Agent Generator

By

Tarig Mohamed Hassan

Supervisors:

Dr. Mohamed Ahmed Al-Affendi **Dr. Izzeldin Kamel Amin**

A thesis submitted for fulfillment of Ph. D. degree in computer sciences

2004

Dedication

I would like to dedicate this work to my family, for loving, raising, and guiding me in every single step of my life, without them I might not be here writing these words.

Acknowledgement

I would like to thank my supervisor Dr. Mohamed AL-Affendi for guiding and supporting me during my PhD. Without his valuable advices and discussions I would not be able to see an end to this long journey. I am also very grateful to my supervisor Dr. Izzeldin Kamel Amin for offering his guidance and constructive criticism and ensuring that what I did was worthwhile. Special thanks go to Mr. Syed Khalil who offered an unlimited support.

Abstract

Mobile Agent Systems are not fully utilized because many security problems need to be solved. SMAG system is a new Mobile Agent System that is designed and implemented in this thesis. The system greatly reduces the security threats of mobile agents. It also incorporate a "mobile agent generator" that accepts general functional specification for an agent, and then it generates the executable. SMAG system contains new security mechanisms. These mechanisms protect all system components, such as View Security Mechanism and Simulator Mechanism to protect mobile agents, Generation Mechanism and Agent Virtual Machine to protect Service Providers and Controller mechanism to protect mobile agents' mobility and integrity. By using C# and .NET framework, SMAG system has been implemented. To illustrate the SMAG system, a Bookshop system has been designed and implemented by using the system infrastructure.

خلاصة

إن أنظمة ال Mobile Agent لم تستخدم بشكل واسع و ذلك لضعف الناحية الأمنية فيها. SMAG هو عبارة عن نظام Mobile Agent تم تصميمه و تنفيذه في هذه الرسالة. يولد النظام ال agent بصور آمنة وفقا لمتطلبات مستخدم النظام وذلك عن طريق المولد الخاص بالنظام. يحتوي SMAG على عدد من طرق الحماية التي تقوم بحماية كل مكونات النظام. باستخدام لغة C# و NET Framework. تم تنفيذ نظام SMAG. لعرض مكونات نظام SMAG تم تصميم و تنفيذ نظام للمكتبات باستخدام البنية التحتية لنظام SMAG .

Table of Contents

Table of Figures

Table of Abbreviations

SMAG	Secure Mobile Agent Generator	
MASIF	Mobile Agent System Interoperability Facility	
RPC	Remote Procedure Call	
C#	C Sharp Language	
ASP	Active Server Page	
SOAP	Simple Object Access Protocol	
DDL	Data Link Library	
CA	Certificate Authority	
CFG	Context Free Grammar	
JVM	Java Virtual Machine	
ICG	Intermediate Code Generation	
SDS	Source Data Space	
DDS	Destination Data Space	
AVM	Agent Virtual Machine	
VSM	View Security Mechanism	
GM	Generation Mechanism	
SA	SubAgent	
DD	Dynamic Data	
CSC	Code Statement Class	
SCM	Statement Code Member	
ESC	Encrypted Statement Class	
ACC	Agent Container Class	
ADTC	Agent Data Transfer Class	1

Introduction

2. Introduction

A mobile agent is an autonomous piece of code that can migrate under self-control from a machine to another in a heterogeneous network. It is a new methodology for computers to communicate. By using a mobile agent system, users can achieve many and different tasks in different places. Through a journey, the mobile agent interacts with different machines to accomplish tasks. So, these machines receive and serve visitors' mobile agents. The mobile agent starts the journey by using an itinerary table. This itinerary table can be fixed and the user specifies it. Also, the itinerary table can be partially fixed. The mobile agent can decide during the journey to visit interesting new places [1]. A movement of the mobile agent is done through a mobility mechanism. In addition, the mobile agent can communicate with different parts of the mobile agent system through communication messages. Also, the user can monitor his/her mobile agents during their journeys.

Traditional models in distributed systems use client-server paradigm which clients and servers communicate either through message passing or remote procedure call (RPC) [2, 3]. This communication method is usually synchronous. A mobile agent model uses a remote programming concept (asynchronous). The remote programming concept allows the mobile agent to execute its task in hosts (remote machines). So, the mobile agent system components need a certain protection. Mobile agent systems are not fully utilized because many security problems need to be solved. The security of the mobile agent has received significant attention. Gary [4] defines four points in security area for mobile agent systems which are: (1) Protection of the host against the malicious mobile agents. (2) Protection of the mobile agent against the malicious host. (3) Protection of other mobile agents. (4) Protection of the underlying network. This thesis focuses on the mobile agent security area. A secure model of the mobile agent

is designed and implemented. The model provides all parts of the system with protection requirements. By using the C# and the .Net framework, the SMAG system has been developed. Some experiments are conducted by using this system.

2. Basic History Concepts

By using different types of paradigms, the applications of distributed systems have been built. Through client-server model, the applications are structured. The client server model is widely used. In this model of communication, a server offers a set of services. There are three main models that allow computers to communicate, as follows:

- Remote Procedure Call (RPC):

  A remote procedure call (RPC) allows a client to invoke a procedure that is located on a server. The invocation is done by using the standard procedure call mechanism [5]. This communication model uses a synchronous or an asynchronous. Figure 1 shows this model.

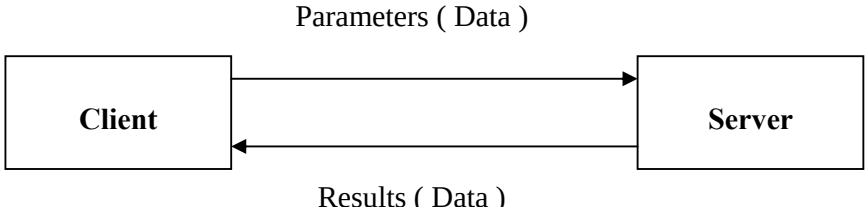


Figure 1 RPC Model

- **Message Passing**

  In this model, a client sends a request message to the server. The server receives the message and sends back a response. This communication model uses a synchronous or an asynchronous. Figure 2 shows this model.

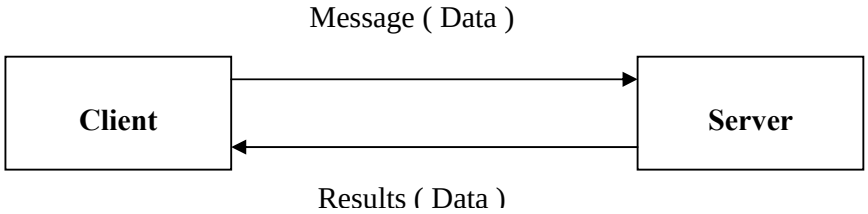


Figure 2 Message Passing Model

- **Remote Evaluation**

  By using a remote evaluation model, a client sends a subprogram to the server. The server executes the subprogram and returns its result to the client. This model was proposed by Stamos and Gifford [6]. Figure 3 shows this model.

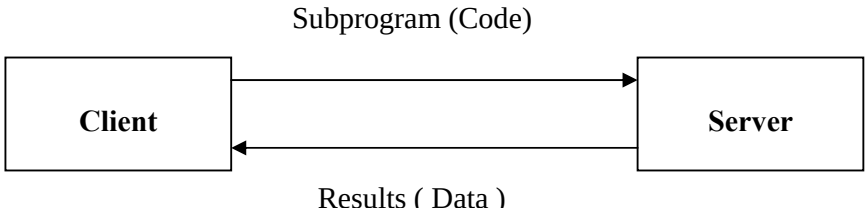


Figure 3 Remote Evaluation Model

- **Active Message & Mobile Agent**

This model was introduced by earlier systems like R2D2 and Chorus [7, 8]**.** The active message could migrate from a machine to another**.** It refer to its an executed code that is executed in each machine. A generic concept is a mobile agent that carries a code and data.

Figure 4 shows this model.

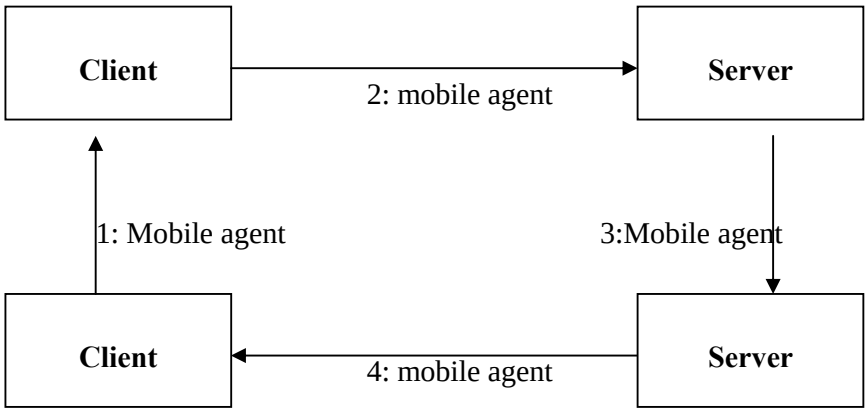


Figure 4 Mobile Agent Model

3. Motivation

Mobile agent systems introduce several advantages in distributed system application areas, for example [9]:

3.1 Mobile Computing

A mobile agent paradigm is suitable for mobile computing [10]. Computers can be disconnected from the network whereas the mobile agent achieves its work. For example, a user can send out the mobile agent through mobile devices (Laptop) into the network. The user can disconnect the device and the mobile agent will start its execution in remote hosts. The mobile agent will be ready with a result when the user reconnects the device.

3.2 Bandwidth

In traditional client-server model, a client can make a series of cross-network call to lower level operations. These calls bring intermediate data across the network on every call. So, this will waste an amount of network bandwidth [11]. Mobile agents move to hosts (servers). They do not waste bandwidth because the mobile agent performs a desired processing before it returns to the client.

3.3 Latency

By using mobility feature, a mobile agent migrates to a host and interacts with the host resource. This mechanism is faster than across the network. Latency between the user and the client server application (Web-based application) is much lower [9].

3.4 Dynamic Deployment

In a short time, a mobile agent can move through different parts of a distributed application in the network sites. It allows development, testing and installation during its journey [12].

4. Mobile Agent Application Areas

By using mobile agent systems, many application areas can be developed. These applications provide useful features. In the following paragraphs, some of these applications can be described briefly as follows [13].

4.3 E-Commerce Application

An e-commerce application allows users to perform business transactions. The e-commerce applications use the network to achieve their tasks [14]. The mobile agent can help in this field by locating appropriate offering, negotiating and executing business transactions on behalf of its owner.

4.4 Distributed Information Retrieval

A mobile agent can query many places during its journey. Using the mobile agent in information retrieval applications is more efficient than moving all the data through the network. In addition, mobile agents can perform extended searches to collect additional information [15].

4.3 Personal Assistant

Mobile agent systems can help their owners in different personal works by executing some tasks on remote hosts. For example, inviting people to a meeting, the mobile agents negotiate and establish a meeting time, confirm the time of the flight, and so on [16].

4.6 Telecommunication Network Service

A Mobile agent with intelligent network environments can support the dynamic deployment of intelligent network services. The mobile agent can provide customized services on demand [17].

4.7 Network Management

Mobile agents can contribute in this type of application. The mobile agent can be used to monitor a system. It can be used also to bring possible trouble to the attention of the system maintainer [14]. The future research will use the mobile agent to achieve self-configurations, self-diagnostics and self-repair [18].

6. Thesis Structure

The thesis concentrates on the mobile agent security. It is organized as follows:

Chapter 1 Literature Review: This chapter presents most pervious works of mobile agent systems. It also presents Mobile Agent System Interoperability Facility (MASIF) as standard for mobile agent systems. The chapter reviews the basic concept of mobile agent systems. Moreover, the pervious efforts in the mobile agent security area and the security requirements are presented. In addition, it presents some interesting mobile agent systems issues. The presentation includes the architecture and a security model of each.

Chapter 2 Problem Definitions: Based on the information highlighted in chapter 2, the main problems in the security area of mobile agent systems are identified and listed. This chapter presents the research challenges and its contributions.

Chapter 3 SMAG System Architecture: This chapter explains the Mobile Agent Specification Language (MASL) as a new language to represent mobile agent behaviour. Also, it presents the syntax and the semantic of the MASL.

This chapter also describes the architecture of the system model and the contributions of the new model. It describes the role of each component. The model consists of a mobile agent, a mobile agent base, and a service provider (host), a controller and a system administrator. Moreover, it shows the mobility, communication and a service protocol.

Chapter 4 Security of the SMAG system: This chapter explains the thesis contributions in the security area of mobile agent systems. This chapter introduces:

- A Generation Mechanism and an Agent Virtual Machine to protect service providers.
- A View Security Mechanism to protect mobile agents.

- A Simulator Mechanism to protect dynamic data states.
- An Encryption Mechanism, a Controller and a Safe-Data-digest mechanism to protect transferring of mobile agents and communication messages.

Chapter 5 SMAG system Implementation: This chapter describes some information about the C# language and the .NET framework which are used in the SMAG system development. Moreover, it explains the classes in .NET framework used to implement the cryptography mechanisms.

Also, this chapter describes the SMAG system implementation. The implementation covers the most contributed concepts in this thesis that are detailed in chapter 6. The descriptions are from a programmers' perspective. By using C# and .NET framework, the SMAG system has been developed.

Chapter 6 Illustrating SMAG Usability: A Distributed Bookshop System Using Mobile Agents: In this chapter, the system is used as a platform to implement a Bookshop application. Samples of scenarios are presented. They show how the system manipulates these scenarios.

Conclusion and Future Works.

Appendix: It contains the software resources of the SMAG system.

Chapter 1

Literature Review

1.1 Introduction

This chapter covers the main concepts used in mobile agent systems as surveying. It covers the basic concepts of a mobile agent, a mobile agent base, a service provider (Host), mobile agent mobility, communications and Security. These items constitute a mobile agent system model. It also covers the Mobile Agent System Interoperability Facility (MASIF) as standard for mobile agent systems. Moreover, the chapter presents some interesting mobile agent systems issue.

1.2 Basic Concepts of Mobile Agents

This section describes the basic concepts that the mobile agent systems are built on them. The description is general over all models issued. It also covers several mechanisms supported by these models.

1.2.1 Mobile Agent

A mobile agent is an entity which on behalf of its owner, performs different tasks at different places. It can move from one place to another under its control. From a programmer perspective, the mobile agent is an executed code that can carry a list of computation operations to perform them on different machines. It can also request its current location to dispatch it to another machine according to its itinerary. These operations represent owner's requirements. A journey of the mobile agent is described in itinerary table that contains addresses of the target destinations that the mobile agent will visit. A mobility mechanism is a key feature of the mobile agent systems. This feature allows the mobile agents to move from a place to another. By using the itinerary table, the mobile agent moves to a first machine. When it arrives at that place, it performs some tasks. These tasks represent mobile agent behaviour at that place. The mobile agent knows what to do in each place. After it completes its works

at that location, it requests the current location to send it to the next place. This scenario will be repeated in each host that the mobile agent will visit. By using identification information, the mobile agent identifies itself to each place. According to the mobile agent identification, the place can accept or reject it. After completing all the visits of the journey, the mobile agent returns home with a collection of results. The mobile agent owner can extract information from the mobile agent. During the journey, the mobile agent can communicate with some parts of the system through communication message protocol under some security conditions. The owners of mobile agents also can send messages to their mobile agents in order to terminate or notify them by some information. By using language environments, most of the current mobile agent systems use these environments to execute their mobile agents. There is no generic model for a mobile agent frame. So, the mobile agent has several forms.

1.2.2 Mobile Agent Base

A Mobile Agent Base is a home of mobile agents. From this component, the mobile agents start their journey. When the mobile agent completes its duties, it returns back home. Moreover, the mobile agent base monitors and controls the mobile agents when they are in the network space and until their return. In addition, it provides the system with some basic services such as creating, dispatching, managing, storing, communicating, generating identification information for each mobile agent and so on. The mobile agent base creates the mobile agent according to the owners' requirements. When the owner decides to dispatch the mobile agent, it will achieve that goal by using some information about a target destination address. It can also manage the mobile agents such as, terminating, suspending, resuming, calling them to return and sending messages …etc. The mobile agent base allows the mobile agents to

use parts of its storage media. Through communication protocols, the mobile agent base can either communicate with other parts of the system or self-initiate or communication according to the mobile agent owner's requests. The mechanisms of communication messages are different from a system to another; they depend on the protocols which are used and the mode of communication messages such as synchronous or asynchronous. Through unique identification information, the mobile agent base can deal with the mobile agent before, during and after it starts its journey. This information is generated by the mobile agent base in the most current systems. The users can access and perform all the above-mentioned operations. They must have authorities and access permissions for security reasons. Chess et al [12] suggests an agent passport for its basic attributes for traveling. In the most current systems, the mobile agent bases have the same objectives but there are some differences in their architectures. The following sections describe some of these architectures presented by some systems issued.

1.2.3 Service provider (Host)

A service provider represents a machine that the mobile agents visit. It provides the mobile agents with specific services. The mobile agent can visit one or more than one service provider during its journey. The service provider consists of multi places. Each place can host one mobile agent at a time. Thus, the service provider can serve many mobile agents at one time. Through the place, the mobile agent can access resources. Each service provider must have a unique name and address in the system. Before the host starts to deal with the visitor mobile agent, it authenticates its identification information. If the mobile agent passes all security requirements, the host will provide them with several services according to their level of access permission. Some service providers store the visitor's mobile agent in their storage

media in order to protect the mobile agents from system crashes [19]. The service provider can receive messages through defined protocols and forward them to target destinations (hosted mobile agents). It also allows the mobile agents to interact with each other through communication protocols. After the mobile agent completes its duties in the host, it requests the current service provider to transfer it to the next station or return its home.

1.2.4 Mobility

Through mobility feature, mobile agents can move from one place to the other in a network media. When the mobile agent completes its work in a service provider, it requests its current location to dispatch it to the next place. According to the request of the mobile agent, the service provider captures mobile agent states. By using mobility mechanism, it transfers mobile agent data to the target destination. Vigna and Fuggetta et al.[1,20,21] defined the components that are hosted by the service provider as execution units and resources. The execution unit represents an algorithm of the computation. The resource represents the entities that provide the algorithm by information. It can be shared among multi execution units, for example, data files or objects in the object oriented systems. The execution unit consists of two parts: part one, a code segment that describes the behaviour of the computation. Part two, a state that consists of data space and execution state. The data space is reference to resources that can be accessed by the execution unit. The execution state is private data that cannot be shared. It also represents the execution control information. Most mobile agent systems offer two forms of the mobility characterized by the execution unit constituent that can be moved: (1) Strong mobility allows a code and an execution state of the execution unit to migrate to different places. (2) Weak mobility allows a code with some initial value to migrate without execution state. There are

some issues in implementations of code mobility [2]. One possibility is to allow the mobile agent to carry all its codes. So, the mobile agent can be executed in every location. Some systems prevent the mobile agent to carry any code but it contains references to its code resources. If the mobile agent needs any code to use and that code is not available in a current location, the current location will contact the code resources and download the mobile agent execution requirements. Other systems prevent the mobile agent from carrying any code and the required code will be pre-installed on the service provider. This approach is good in the security because no foreign code is executed. But it suffers from poor flexibility. Section 3 presents many types of mobility mechanisms introduced by some interesting mobile agent systems issued.

1.2.5 Communications

Communications concept allows mobile agents to communicate with other entities in the mobile agent system. These entities may be mobile agents, service providers, mobile agent bases and owners. A communication model can be a synchronous communication or an asynchronous communication. Synchronous communication means a sender blocks until the reply of the message arrives. Asynchronous allows the sender to continue performing its task and the result of the message may come later on. The communication model can be based on abstract information which contains an item key, an access control list and an item value [14]. There are many mechanisms in this area [2], such as:

- Message Passing: This mechanism allows a mobile agent to send a datagram style message in order to interact with the other mobile agents.
- Methods Invocation: This mechanism is used in object based systems. The mobile agent can interact with each other when they are co-located in one

place. They can be provided with references to each other; through these references they invoke operations.

- Collective Communication: This mechanism is useful in applications that use a group of the mobile agents which are collaborative tasks. It can be used to communicate with or within a mobile agent group.
- Data Shared: This mechanism allows the mobile agents to communicate asynchronous. The mobile agent can put its information in shared area of memory and other mobile agents can access this information. So, the shared area is for exchange data between authorized mobile agents.
- Event message: This mechanism is used when some events occur. The mobile agent may request the system to notify it when some actions happen. This mechanism can be referred to as publish-subscribe of the event delivery. Another model is to provide broadcast of events.
- Session Communication: This method introduced by Baumann et al [22]. It can be used to synchronize the mobile agents that want to communicate with each other, even if they stay in different places.
- KQML Communication: This mechanism uses Knowledge Query and Manipulation language that is a high level protocol and language for agent's interactions and communication. This language is based on speech act theory in linguistics [1, 23, 24].

1.2.6 Security of Mobile Agent

Many proposals are suggested by different mobile agent systems. Gary [25, 4] defines four points in security area for the mobile agent systems:

- Protection of the host against the malicious mobile agents.
- Protection of the mobile agent against the malicious host.

- Protections of other mobile agents.
- Protections of the underlying network.

1.2.6.2 Security Requirements

In the computer network System, users require five security requirements [26]. These requirements also are required by users of the mobile agent system:

- **Confidentiality**: a mobile agent base has private data that is stored in the system or carried by the mobile agent. These data must remain confidential. A system framework must be able to ensure that confidentiality.
- **Integrity**: A mobile agent base must protect mobile agents against unauthorized modification.
- **Accountability**: Each mobile agent must be held accountable for its actions. It must be uniquely identified, authenticated and audited.
- **Availability**: A mobile agent base must be able to ensure the availability of both services and data to the mobile agents. Also, it must support concurrent execution, access control, deadlock management …etc.
- **Anonymity**: a mobile agent base may be able to keep mobile agent identification secret from other mobile agents.

1.2.6.2 Host Protection

The host may face different types of risks from a malicious mobile agent that are detailed in [27]:

- Pilfering of sensitive information.
- Damage to host resources.
- Denial of services to other mobile agents.
- Annoyance attack.

Some technical solutions are proposed in this area are following [26, 28]:

- **Software-Based Fault Isolation:** By using software, this method isolates application modules into clear different fault domains [29]. It also allows a programme that is written by unsafe language to be executed safely within a single virtual address space of an application. The access to a system resource can be through a unique identifier domain. This technique referred to Sandbox Model: Limits the privileges, and each mobile agent must have authority to access resources [30, 31].
- **Code signing**: The mobile agent must carry an insurance that host is trusted [30, 32, 33]. A digital signature is used to confirm the authenticity of an object. By using asymmetric encryption mechanism, the digital signature is generated.
- **Firewalling**: Examine the mobile agent as they enter a trusted domain and decide whether to execute it or not. If the answer is 'yes', what are the properties for running the host in this situation [32, 34].
- **Safe Code Interpretation:** Some systems use various mechanisms to protect the host, such as by depending on the programming language, e.g. safe-Tcl interpreter [35, 36]. The idea of this method is that commands considered harmful can be either made safe or denied to the mobile agent. For example, to deny execute of commands, the command can be executed as arbitrary string of data. Java programming language is one of the most widely used [21]. Java uses Sandbox model, because it isolates memory and method accesses. The security in java is enforced through a variety of mechanisms.
- **Proof Carrying Code:** In this approach, a mobile agent producer proves that a mobile agent code has safety properties [37]. Also, it is a prevention method

and not detection like Code Signing. The code of mobile agent and proof are sent to the service provider. The service provider can verify the proof code. This technique dose not use cryptography technique or external assistances. If verification succeeds, a safe code is transferred**.** Moreover, this approach is based on logic, type theory and formal verification. But, there are difficult problems to be solved before it can be considered practical. Also, it is tied to the hardware and operating environment of the service providers.

- **Path Histories:** The Idea of this method is to save an authenticable record of the previous service providers that a mobile agent visits [12, 38]. When the mobile agent arrives to the next service provider, the service provider can discover whether to process the mobile agent and what resource constraints to apply. Each service provider adds and signs entry to the path, indicating its identification and identity of the next service provider to be visited. To prevent an attack, the signature mechanism is used.
- **State Appraisal:** this method is to ensure that mobile agent state information has not altered [39]. Appraisal Functions are used to discover privileges that are granted to the mobile agent. The mobile agent that its state information violates, it may grant a restricted set of privileges. Otherwise no privileges will be granted to the mobile agent. The appraisal functions are produced by mobile agent owner. They become part of the mobile agent code.

1.2.6.6 Mobile Agent Protection

A mobile agent represents the user's tasks and may contain sensitive data. It is important to protect the mobile agent from malicious hosts which may read any information in the mobile agent without cooperation. The following points define some risks around the mobile agent described in [40]:

- Eavesdropping: Try to know the behaviour and the objectives of the mobile agent.
- Intercepting and Altering: Try to intercept the duties of the mobile agents or change the data by deleting or substituting.
- Reply: Send a copy of the mobile agent for illegitimate purpose.
- Masquerade: An entity pretends to be a different entity.

The main idea that is selected with this problem is a detection and not prevention, such as [41, 26].

- **Execution Tracing:** Gary [42] proposes to detect attack when the mobile agent returns. Again Vigna [43, 41, 26] proposes a mechanism to detect misbehaviour by using cryptographic traces and looking to the mobile agent history file (log) where the owner of the mobile agent will know if the mobile agent has achieved its duties correctly or not [44]. But by using this mechanism, the mobile agent system must maintain a large log file. Also, a secure protocol is required for transferring cryptography hashes for external entities.
- **Obfuscated Code:** This mechanism is to create a black-box out of an original mobile agent [45]. This mechanism uses a black-box to perform the same work of the mobile agent as an original mobile agent, but by different structure. This mechanism has disadvantages, for example, no black-box algorithms exist that work for arbitrary data. Sander and Tschudin [46] used cryptography in their approach in special cases by having the mobile agent programme compute not the original, but an encrypted version of it. The result of this function is decrypted by the

mobile agent’s owner. But, cryptography theory has not a schema that computes arbitrary function in a non-interactive manner.

- **The Ajanta system**: This system uses an approach for protecting the mobile agent [47]. The approach consists of three mechanisms, the first is to allow the programmer to declare parts of the mobile agent state as Read-Only and if any modification occurs to these parts, the mobile agents' owners can detect it by using the digital signature mechanism. The second mechanism is let the mobile agent create append-only data states container where the data stored in this container can not be deleted or altered without detection by the mobile agent's owner. The third mechanism is to let programmers to define data states to specific hosts and no other hosts can deal with these data states. These mechanisms use the encryption, the decryption and the digital signature.
- **Partial Result Encapsulation**: This mechanism detects tempering by using encapsulating the results of the mobile agent actions at each place, for subsequent verification or when it returns to the base place [48, 44, 64]. The disadvantage of this mechanism, for example, does nothing to ensure mobile agent privacy. Also, the results to encapsulate may not be immediately clear.
- **Environment Key Generation:** When some environment condition is true, this mechanism allows the mobile agent to take an action. A key is generated which is used to unlock some executable code that was encrypted [49]. This approach has weakness such as: the control of the mobile agent could simply modify. The host limits the capability to execute a mobile agent code that is created dynamically, sense it is

considered an unsafe operation. The mechanism is connected with other protection mechanism.

- **KeyLets**: This mechanism partitions a mobile agent as components according to the task type [50]. By using secret keys, it encrypts each component to protect them. The distribution of keys to different hosts is done through the execution of specific type of mobile agent that is termed a Keylet. The disadvantages of this approach: Propagation requires a third party code producer that can supply the mobile agent by a template the mobile agent owner. Also, a large number of transactions related to the keylet and a host may not be willing to support the increased of computation. Moreover, key revocation is not good in quality. In addition, it requires a complicated mechanism to categorize tasks of the mobile agent. Also, this mechanism does not protect the mobile agent code completely.
- **Mutual Itinerary Recording:** This approach allows a mobile agent's itinerary to be recorded and tracked by another cooperating agent [51, 52, 53]. When the mobile agent moves among hosts, it provides the cooperative peer with the last, the current and the next location. The peer maintains a record of the itinerary and takes appropriate action when some defects occur. But, the weakness of this approach is the cost of the setting up of authenticated channel and inability of the peer to specify which of the two hosts is responsible if the mobile agent is killed.
- **Itinerary Recording with Replication and Voting:** By using replication and voting technique, a mobile agent arrives to its destination safely can be ensured [54]. The idea uses multi copies of mobile agents and the

malicious host may corrupt a few of them. Enough replicates complete computation successfully. This technique is suitable for specialized application where mobile agents can be duplicated without problem and the additional resources consumed by replicate mobile agents.

1.2.6.7 Protections of other Mobile Agents

The security of other mobile agents when they are located in one place means the mobile agent is prevented to interfere with other mobile agents. This problem can be viewed as sub problem of the protection of host against mobile agent because the visiting mobile agents are viewed as host resources to the mobile agent that try to interfere with them [25]. When the host is protected, all mobile agents located in the host are automatically protected.

1.2.6.8 Protections of Underlying Network

A mobile agent uses the network to travel from one host to another. It may consume the network resources excessively by clones unlimited. Annoyance attack will occur to the network. Shadow protocol (proposed by Bamann and Rothermel [27]) can find and terminate all clones of the mobile agent in network.

1.2.6.6 Secure transfer mobile agent and communications

A mobile agent needs a secure way when it decides to leave its current location to another. The communication message to / from the mobile agent must deliver in secure ways. The mobility and communication messages can be dealt with as the same in the world network, because the mobile agent is defined as an object and it is converted to binary data before it transmits to its target destination and this also happens to the communication messages. The transmission data may face different types of attack detailed in [40]:

- Attempts to steal useful information from it.
- Eavesdropping which can leakage sensitive information.
- Intercept and modify the transmission data.

Various approaches suggested in this area, in order to protect the data transmission. Some of them, for example, are used cryptography mechanisms, by encrypt the data of the mobile agent or message before transfer it and decrypt it when the destination receives the data.

1.3 Mobile Agent System Interoperability Facility (MASIF)

In 1998, the Object Management Group (OMG) introduced the MASIF as standard for mobile agent systems [55, 56]. MASIF contains a set of definitions and interfaces. The interfaces are for interoperable between mobile agent systems. There are two interface definitions: (1) A MAFAgentSystem, for the transferring a mobile agent and management. (2) A MAFFinder, for naming and finding a mobile agent. MASIF standardizes the following operations:

- Mobile Agent Transfer: This operation describes the methods for receiving mobile agents and search for classes' requirements.
- Mobile Agent Management: This operation describes the standard to create, suspend, resume and terminate a mobile agent.
- Mobile Agent System Type and Location Syntax: This operation describes the supported of the mobile agent by the system type and through standard location syntax; the mobile agent system can locate one another.
- Mobile Agent and Mobile Agent System Names: This operation describes the syntax and the Semantics of the mobile agent and mobile agent system names. It

supports the mobile agent tracking and how it locates the other mobile agents in different systems through their names.

1.4 Mobile Agent Systems Issue

In this section, some of the interesting mobile agent systems are presented. The presentation covers the architectures of these systems. Also, it focuses on the security mechanisms that are used in each. The main objective of this survey is to show the different models and specify the strong and weak points in each. The following paragraphs cover some mobile agent systems issue [57]:

1.4.1 Telescript System

Telescript system was a first commercial implementation of the mobile agent concept in 1990 by General Magic [58]. The system contains three components: (1) the language of development of the mobile agents and places: (2) The interpreter (Engine) of the language. (3) The communication protocol for exchanging the mobile agents. The mobile agent is derived from the agent class. This class contains several public methods used during the mobile agent execution. Each mobile agent has a name, an identification authority and some information about its destination (Ticket). The server in Telescript consists of many stationary agents that are interacted with visiting mobile agents (places). The server might contain a number of the places. Each place has a name. The place provides the visiting mobile agents with different services. The mobile agent in Telescripte system can move from a place to another under the control of itself by executing "Go" command and this command requires some information about the mobile agent and the address of the next station. The protocol of communication allows two servers to communicate in two levels:

(1) the high level specifies how Telescript converts the mobile agent into binary data. (2) The low level specifies the mutual authentication between two servers and it transfers the data of the mobile agent as binary data format. The mobile agent uses the meet instruction to communicate locally with execution environment. The co-located mobile agents can communicate with each other by method invocation mechanism and event signaling facility [16].

The security model in Telscripte is based on two mechanisms: The first is authority that lets a mobile agent and places to interact with each other. A place queries an incoming mobile agent's authority and according to the result, it accepts the mobile agent or rejects it. The second is the permit which limits access right, resource consumption quotas…etc. The server can terminate any illegal operation done by the mobile agent.

1.4.2 Aglets System Development Kit (Aglet SDK)

This system is developed by IBM Tokyo Research Laboratory [59]. The Aglet SDK is used in E-Commerce area. The system is written in Java Language. It consists of four components:

- Aglet Framework is a collection of classes that provides the system with different functions.
- Visual Agent Manager represents the Aglet base (local agent server).
- Agent Transfer Protocol is an application-level protocol which distributes the agent-based systems. Also, it is used to transfer the mobile agent from a node to another through a network connection.
- Agent Web Launchers is used to control the Aglet. It allows the system to create the aglet. It retracts aglet from/back to client's browser.

The aglet class represents the mobile agent. It contains the behaviour of the mobile agent. Also, it controls the Aglet movement and execution. The server of the system can host and serve multi Aglets concurrently. It has interface call AgletContext. Aglet can use AgletContext to get information about the environment in order to send message and interaction with other Aglets in the same server. The system uses Agent Transfer and Communication Interface Protocol (ATCI) to send and receive messages. The Aglet system defines four methods in Agent Transfer Protocol (ATP) for mobility: (1) The dispatch method for sending the mobile agent to a destination and starting serving the aglet. (2) The Retraction method for requesting the destination to return the specified aglet. (3) The Message method for passing a message and receiving a reply by using the Aglet identification name in response. (4) The fetch method for exchanging identified information.

The Aglet system defines three types of communication messages for remote location: (1) A now-type message: the sender waits until the receiver has achieved the handling of the message. (2) A one-way-type message: This is a synchronous mode and no acknowledgement will be back. (3) A future-type message: This is an asynchronous mode and the reply will be in the future. The Aglet has a queue for incoming massages.

The security model defines several policies in ASDK [40]. The policies define set of rules by one administrative authority. The rules contain:

- The conditions under which the aglets can access the resources.
- Specify the authority according to authenticate information required from the users or any entity work in the system.
- Security requirement for a communication between Aglets and other parts of the system.

- Specify the degree of security in any activity.

The AgletProxy protects the Aglet from malicious Aglets. It consults the SecurityManager when the Aglet manipulate with other Aglets or context.

1.4.4 Tacoma System

Tacoma system is a joint project developed by the University of Torms (Norway) and Cornell University in 1993. The current version of the system supports various operating systems, Java, ML , Perl, Python, Schema, C, C++ , and Tcl/TK. [36,42] [60] . The mobile agent components (Code and Data) are stored in folders. The folders can move from a host to another and meet stationary agents in a target place. Each mobile agent has various folders. A Code folder contains the source code of the mobile agent. A Data folder contains the data related to code in the code folder. The folders related to the mobile agent are grouped into a briefcase. The mobile agent uses the frame of the briefcase to travel from a host to another. The cabinet file is stationary folder work as permanent data in the host which the mobile agent uses. The connection point in the server that the visiting mobile agents enter through it, is called Tac_Firewall. Also, it controls the income briefcases. Depending on the programming language, the stationary agent called Tac_Exec extracts the mobile agent from the briefcase and runs it. Tacoma mobile agent can move from a host to another by using the meet primitive after specifying all information for that. Also, the meeting command is used for communicating with co-locate. The system supports the synchronous and asynchronous communication mechanism. Moreover, the cabinets are used as shared memory for communication.

The Tacoma system has a weak security system. For example, the system parts are executed in a sub-tree of the host file system. Also, the owner of the server specifies the hosts that can accept the mobile agent [20].

1.4.4 Agent Tcl System

Agent Tcl which was developed at Dartmouth Collage. It focuses on a five- research areas [61]: (1) Performance, (2) Supporting for multi languages, (3) Cryptographic authentication and restricted execution environments to protect a machine from malicious mobile agent, (4) Economic base models to limit the mobile agent's total resource consumption across multi machine and (5) Networking sensing, how mobile agent can determine the best path through the network according to its duties. The Tcl script represents the mobile agent. It can move from a machine to another that supports the mobile agent execution, communication, state queries and stable storage [4]. Agent Tcl consists of a server and an execution environment. In each machine, there is a server which accepts income mobile agents, authenticates the mobile agent identified information and selects the appropriate environment. Also, it keeps all information about the mobile agents under processing and can answer any query. Moreover, it controls the execution according to request of the authorized user by suspending, resuming, terminating the running of the mobile agent. Through the server, the mobile agents can achieve their communication messages. The execution environment has the interpreter which executes the mobile agent. It enforces the security mechanism and it captures the state of the mobile agent when it decides to travel to another station. Moreover, it provides the mobile agent with some functions that help to achieve the duties. Agent Tcl supports multi language such as Tcl, Java and schema. The mobility is achieved by using Jump instruction which captures the mobile agent state and dispatches the mobile agent to a new machine. Agent Tcl uses

two mechanisms in low level of communication. (1) Message passing. (2) Direct connections. These mechanisms work as local and remote locations. In high level Remote Procedure Call (RPC)[62] implemented at the mobile agent level on the top of low level mechanism.

The security model in Agent Tcl protects the hosts from the malicious mobile agent by enforcing various mechanisms such as: the mobile agents are digitally signed before migration so that the receiver can authenticate them. The resource manager can decide according to a mobile agent owner's to specify the resources and the resource mangers have access control lists to make their decisions and etc… [63].

1.4.9 Ajanta System

The Ajanta system has been developed on the Java programming language [2]. It has a good security model that contains protection mechanism for the mobile agent and another for a server (host). The system defines many classes which represent the main functions of the system. The mobile agent is one of these classes. It contains a set of methods. Programmers can inherit this class and add their methods. The server can provide the mobile agent with basic functions such as execution, migration, and a binding to a server environment and resources. The multi mobile agents can be hosted by a server at one time and each one has an isolated and protected domain. The server has a database called a domain registry which maintains information about the mobile agents currently executing on. Any server has environment component which acts as interface to the visiting mobile agents to the server. The server maintains a resource registry which is used to set up save binding between the mobile agent and the resource. Also, the server has an agent transfer component to dispatch the mobile agent to remote destination. This unit uses Ajanta agent Transfer Protocol (ATP). Moreover, the server has a component called Server Interface which works as

interface to perform a remote request. Through this component, for example, the mobile agent owners can control their mobile agents. Any entities in Ajanta system has global independent location name in the URN format.

The security model in Ajanta system covers different parts of the system. Once the agent arrives to its destination, a protection domain is created to execute it. The other mobile agents can not temper with their code. The system also uses a proxy based mechanism to control access of resources. The mobile agent is given a reference to a proxy instance which contains a secure path to the resources. Moreover, the security model protects the mobile agent state from malicious operations. Three mechanisms are designed in this system [47]. (1) A ReadOnlyContainer mechanism allows the mobile agents to declare parts of their state to be read only, and detect any modification occurred to these parts. (2) An AppendOnlyCotainer mechanism allows the mobile agents to store the data collection during their travels and any tempering with it can be detected. (3) A TargetedState mechanism allows the mobile agents owner to encrypt parts of the mobile agent in order to let specific server access these parts and prevent other servers.

1.4.10 Concordia system

Concordia system developed by the Mitsubishi Electric Information Technology Center America in 1997[65]: The system is written in Java language. The mobile agent in the system can move from a place to another. It is represented in Java Class. The mobile agents in the application can work as groups and they achieve complex results. They perform their tasks concurrently. Concordia consists of the set components that provide the system with different functions such as administration, security, communication, and so on. The Concordia server consists of several components. The agent manager can create, terminate and execute the mobile agents.

A queue manger recognizes the mobile agent transportation over the network. The service bridges allow the visiting mobile agents to interact with services provided by the server. A persistent store manager protects the mobile agent from the system crashes. Security manager enforces the local security policies. A directory manager allows the mobile agent to search and locate the application it looks for. The mobility mechanism in the system has a queue manger which the server uses to store the mobile agents before they move to another station until they have been received by their destinations. The mobility is accomplished by using Java Remote Method Invocation (RMI) and Secure Socket Layer (SSL). A mobile agent journey is described in itinerary table which contains multi destinations and the work that will be done in each. The communication mechanism in the system allows the mobile agents to interact with each other by two mechanisms: (1) Distributed events: the mobile agent must first register with component called the event manger to be able to receive the event messages and also to send messages to other mobile agents. (2) Agent collaboration, this mechanism allows mobile agents to be in a special group that let them interact and exchange the information with each other and introduce some results.

Three security mechanisms are supported by Concordia. (1) Transmission protection mechanism which uses digital certificates for authentication, encryption the mobile agent before dispatching to another node and a symmetric key exchanges to authenticate a sender and a receiver. (2) Mobile agent storage protection mechanism which encrypts the mobile agent when it stores in the servers. (3) The server resource protection mechanism is controlled by security manager component. The security manger authenticates the mobile agent before it accesses resources.

1.4.11 Voyager System

The Voyager developed by the ObjectSpace and it has been written in Java agent-enhanced Object Request Broker (ORB) [66]. The goals of the system are to provide a universal solution for distributed object interaction techniques. Each mobile agent consists of two entities: (1) A virtual stationary agent that works as a reference to the second entity and stays in the server which creates the mobile agent. (2) The mobile agent can move over the network. The system can contact the mobile agent through the virtual agent. The virtual agent forwards and receives the messages from a real mobile agent. Each mobile agent is identified by a unique name. Also, it has a limit life time. Servers in Voyager provide the mobile agents with runtime environments and several services. Also, each server has a ClassLoader for new resources and can act as code server through built-in HTTP support. The mobile agent can move from a place to another through Move-Message. This message contains the destination address and the task will be done after the mobility is completed. Before the mobile agent moves, it creates special agent called secretary, which forwards all the messages that are sent to location after the mobile agent leaves. The Voyager supports different types of communication messages: synchronous messages, one way messages and future messages. Also, it supports multicast and publishes or subscribes model.

The security model in the system depends on Java security and distinguishes between native and foreign objects. The native objects whose class in the server and the foreign objects whose class must be downloaded through the network. The native object can perform any operation but the foreign object must obey some security policies.

1.4.12 Mole System

In 1995, the University of the Stuttgart, Germany developed Mole system. Mole system is the first mobile agent system implemented in Java [67, 68]. The mobile agents in the system consist of the group objects. These objects have reference to their execution environment. They can move from place to another in order to meet other mobile agents or interact with services. Also they are identified by two ways (1) unique identifiers generated by the system. (2) Badge is an application generated identifier and it represents the role of the mobile agent at a given time. The mobile agent can wear many badges at the same time. A server in Mole system consists of multi places. Each place can receive and execute the mobile agent. Also, it provides an available service to the visiting mobile agent. A server can control the total CPU time of the visiting mobile agent through Master Control Process (MCP) which schedules all threads. The mobility in the system uses the Java RMI. When the mobile agent decides to travel, it calls MigrateTo method with the next destination address in DNS format. The mobile agent is serialized by using serialization class. The serialized data is sent to a target destination. From this data, the mobile agent is created and started. If some classes are not found in a location, the target location will request the classes from a server code or a source location. The mobile agent can interact with each other by RMI or passing messages. Before the communication starts, they must create a session. The mobile agent can participate a session and if that is not required, they must be in the same place. The session automatically ends when the mobile agents leave their places.

The Mole system uses Sandbox security model [30]. In this model, the mobile agent has no access to underlying system. There is a stationary service where the agents can access resources and secure abstraction of the resources inside the server.

Chapter 2

Problem definition

2.1. Introduction

The mobile agent concept has good contributions in distributed system application areas. It provides many applications in these areas with new features such as reducing network overload, mobile computing, interacting with the resources much faster than across the network, dynamic deployment …etc. But unfortunately, the use of the mobile agents' technology is still young and a number of issues need to be solved. These include:

- No generic models for general purpose to support flexible and reusable components.
- No efficient mechanisms for controlling, monitoring and terminating the mobile agents.
- Most of the current mobile agent systems issued are built on a complex model and they lack transactions support.
- Also most current security mechanisms have failed to protect the mobile agents completely from malicious hosts.
- There is no approach to protect a mobile agent if the host kills the mobile agent itself or alters part of it.
- Some current models use cryptography mechanisms to protect the mobile agents, but these mechanisms are concerned with detection if a mobile agent faces any security problem. These mechanisms do not prevent many attacks.
- Moreover, some current mobile agent systems depend on authentication authorization and access permissions to protect hosts. These ideas ignore the misuses of these authorizations and privileges to perform illegal operations. So, hosts are still unprotected by these mechanisms.

2.2. Thesis challenges

As mentioned above, the mobile agent technology needs effort in different positions to be promising in distributed system applications. There are many challenges in solving all the previous problems which are as follows:

- Defining, designing, and implementing research model fundamentals components as mobile agent experimentation.
- The research model must be simple in usage and acceptable in various areas of applications. In order to be a generic model and to support mobility, the model must provide a secure mobile agent environment, control and monitor mobile agents…etc.
- Protection of a host against malicious mobile agents can be achieved by preventing a mobile agent from access to resources of hosts without permissions, destroying data, stealing sensitive information, altering data and doing any illegal operations. The protection must prevent any possible attacks.
- Protection of the mobile agent against malicious hosts: A mobile agent may encounter many types of attacks. The protection must cover mobile agents' code and data state altering. It must cover not only any information in mobile agents but also their algorithms. In addition, in case a host destroys or changes the mobile agent itself, the protection must cover this point also.
- Protection of the mobile agents' mobility from a place to another: The protection must prevent any leakage of the sensitive information, altering data and transferring of the mobile agents.
- Secure message communication: the communication messages must be sent by secure ways in order to avoid any illegal operations that may be done through them.

2.3. Thesis Contributions

As it is described above, the research has many challenges to deal with. This thesis provides many solutions that solve these problems. This thesis covers all the challenges. In this section, brief descriptions of the contributions provided by this research are given:

- A new Mobile Agent Specification Language (MASL) is designed and implemented. The user of the SMAG system can use this language to describe his/her requirements that will be achieved by a mobile agent. This language is very simple and flexible to use. The syntax of the language statements are appropriated to the mobile agent concepts. By using this language, users cannot be able to represent any malicious operations.
- The SMAG system is designed and implemented out to provide a generic model for a mobile agent system that can be worked in different distributed system application areas. The model includes architectures of several components. The SMAG system provides the users of the system with friendly interface screens. Through these interfaces, they describe and control their mobile agents. It also provides many functions such as mobility, communication, storage …etc of the mobile agents.
- The SMAG system generates a mobile agent according to their owner's specification through MASL. The generation of the mobile agents is done in places isolated from the users. So, the system will accept only the mobile agents that are generated only by the system. Exactly, the generation is done in the mobile agent base.
- To protect a mobile agent, the SMAG system uses a View Security Mechanism (VSM). The VSM is designed and implemented in this thesis. It

uses cryptography mechanisms to protect a mobile agent. The main idea of this mechanism is to encrypt different parts of the mobile agent by different secret keys, so the mobile agent will own different views during its journey. Each encrypted part can be only decrypted by a specific service provider that manipulates this part. Thus, the VSM will completely protect the mobile agent from any illegal operations.

- To protect a mobile agent in case a service provider kills a mobile agent or alters any other parts that are not specified for it, the SMAG system uses controllers and Safe date mechanism to deal with this problem. The controller saves a copy of the mobile agents and data digest before dispatching it to the service provider. Chapter 4 describes different cryptography mechanisms. When the mobile agent returns to the controller, the mechanism checks any effect that occurs. This mechanism is designed and implemented in this thesis.
- For protecting the information that a mobile agent collects through its journey, the Simulator Unit is designed. When the mobile agent returns home, this unit will check all the information that is collected by the mobile agent by executing again the algorithm of the mobile agent locally and making a comparison between two results.
- An Agent Virtual Machine (AVM) is one of the contributions that give the model power in the security area. The AVM is an interpreter that interprets mobile agent behaviour in each visited place. For security reasons and to minimize the mobile agent size, this unit is hosted only by the service providers and the mobile agent base. So, the mobile agent carries only code representations executed by the AVM. This approach allows the development in the MASL.

- Dynamic Behaviour Generation: The SMAG system supports dynamic behaviour generations. So, a mobile agent can visit new places that are not predefined. Through the controllers, the dynamic behaviour is generated. Each controller is provided with a garbage collection unit that frees some parts of the mobile agent in order to be used by the dynamic behaviour

Chapter 3

SMAG System Architecture

3.1 Mobile Agent Specification Language

The Mobile Agent Specification Language (MASL) is a new specification language that allows users to describe their requirements from mobile agents. It is designed and implemented in this thesis. The SMAG system supports the MASL and it accepts only the mobile agent that is generated through this language. By using this language, the user can express different tasks that are achieved by the mobile agent. After the user writes task representations through MASL, the SMAG system takes these representations as a source to the generator of the system. The generator will generate the behaviour of the mobile agent according to that representation. The generation code is represented as information. The information consists of task information and execution code information. The generation is done in an isolated of place from the user and that is for security reasons. The Agent Virtual Machine (AVM) is one of the system components. The AVM is hosted by in a service provider and it can execute the generation code. Like other languages, MASL has specific tokens and syntax. The generator will accept only a text that is written by using these tokens and syntax. This chapter explains the syntax and tokens of the language. The MASL is a flexible language because its syntax are very simple. It is so easy to use. The tokens of the language have a clear meaning, just like a natural language. The user must arrange the statements according to the order of the requirements because the system will generate the behaviour of the mobile agent by using the logic steps of the descriptions. The statements of the language can be divided into two main categories as follows:

1. System Statements: These statements are used to describe the operations that are related to the mobile agent mobility, communications among different parts of the SMAG system, mobile agent creation, etc.

2. Manipulated statements: These statements are used to describe the operations that are related to the users' process requirements such as arithmetic operations, logical operations, call services, defined variables, etc.

3.2 Tokens in MASL

The MASL defines many tokens that are accepted by the generator of the system. The meaning of these tokens are near to English language, for example, the word **Move to** lets the mobile agent move from a place to another. Also, the word **define** for defining variables, it is used to define something in the English language. The variable in MASL must start with alphabetic character and followed also by alphabetic characters. The arithmetic operation tokens are **+, -, *,** /. The logic operation tokens are **>, <, <=, <=, =, <>.** The assignment operation is =. The types of variable are number or string. The token of the call service is **call service**. There are other tokens in the MASL.

3.3 Syntax in MASL

This section describes different syntax in MASL. Before that there are some assumptions about services that the mobile agents can call during their processes:

- **Service Category:**

  All the services provided are based on standard taxonomies and each Service must use the protocol of the services that are defined in this chapter.

- **Service Description:**

  Each service must have a description of the objectives. The description specifies the name, parameters requirement with services, service location and data requirements.

- **Service Call:**

  Before a mobile agent calls service, it must be first in the place of the service that is provided by a service provider. So, the user must know the location of each service.

By using Context Free Grammar (CFG), this part represents the syntax of the MASL. The descriptions show the items:

<*Agent specification*>::= **create agent** < *agent name*> **as begin**
<*statement*>**;{**<*statement*>**;} end**

<*agent name*> ::= <*identifier*>

< *statement*> ::= <*Call Service statement*>|<*Mobility statement* >|
<*Communication statement* >|<*Control statement*>|
<*Assignment statement*>| <*tasks block*> |
<*Declare variable* >

<*Call Service statement*>::= **call service** <*service name*>
[**with value** < *parameters assignment* >
[**set result as** <out*put parameters assignment* >

< *parameters assignment* > ::= <*parameter name*> =<*value*>
[{**,**<*parameter name*> =<*value*>}]

<*output parameters assignment* > ::= <*identifier*> =< *parameter name*>
[{**,**<*identifier*> =< *parameter name*>}]

<*parameter name*> ::= <*identifier*>

<*value*> ::= <*identifier*> | <*number*> | <string>

<*Mobility statement* >::= **move to (** <*location*> **| any | all | home)**

[**through controller**]

<*Communication statement*>::= <*information message*> | <*notify message statement*>

<*information message*> ::= **send (**<*parameters assignment*> **to**

<*location*> **|home)** |

get ((<identifier>{,<identifier>} **from**

<*agent name*> **|<***location***>**)

<*notify message statement*> ::= **forward me by agent (Status | location)**

| return back < *agent name***>**

<*service name*> ::= <*identifier*>

<*Control statement*> ::= <*if Statement* >

<*if Statement* > ::= **if** <*expression*> **then** (<*statement*>| < *task block*>)

[**else** (<*statement*>| < *task block*>)]

<*assignment statement*> ::=	<*identifier*> = <*expression*>
<*tasks block*> ::=	**begin** < *statement*> { ; <*statement*> } **end**
<*expression*> ::=	<*simple expression* > (= |< | <= | > | >= | <>) <*simple expression*>
<*simple expression*> ::=	[<*sign*>] <*term*> { (+| **-** | **OR**) <*term*>
<*term*> ::=	<*factor*> {(***** | / | **AND**) <*factor*>}
<*factor* > ::=	<*identifier*> | (<*expression*>) | <*unsigned constant*> | (**NOT** (<*factor*>)
<*number*>	<*signed constant* > | <*unsigned constant*>
<*signed constant* > ::=	<*sign*> <*digit*>{<*digit*>}
<*unsigned constant*>::=	<*digit*>{<*digit*>}
<*sign*> ::=	+ | **-**
<*location*> ::=	<*identifier*>
<*identifier*> ::=	<*letter*>{<*letter*> }
<*digit*> ::=	**0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
<*Declare variable* > ::=	**define** <*identifier*>[{,<*identifier*>}] **as** **(number | string)**

3.4 Scenario

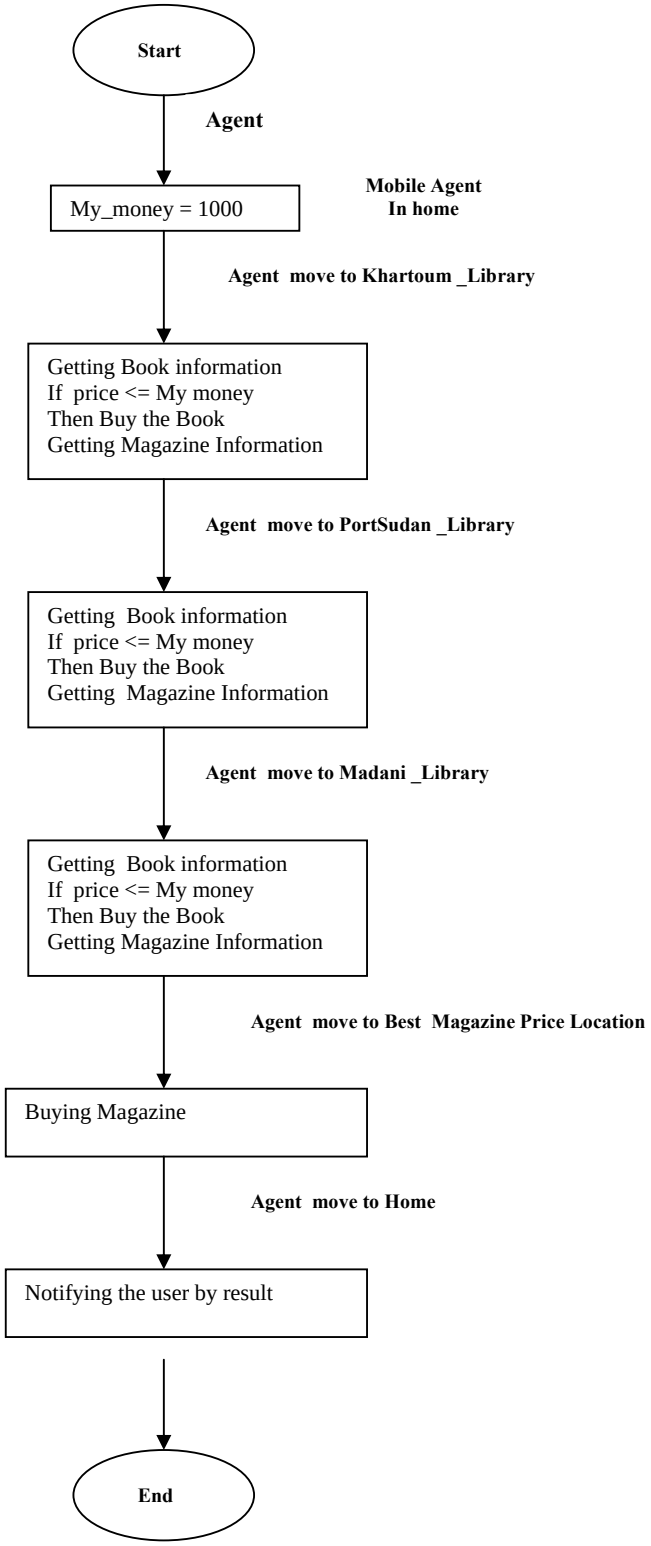
The following scenario contains many tasks which the user wants the mobile agent to achieve. According to the user's scenario, the tasks are represented into MASL syntax:

Ahamed studies computer science. He wants to buy three items by using the mobile agent and he has only $1000. The items that he wants to buy are Java book, Oracle book and Windows2000 magazine .The Java book is available in a services provider called Khartoum_library. The Khartoum_library has the following service: Books.Computer.Language.Buy() for buying the computer language books, this service needs parameters Book name, publishing year , author's name, credit-card number and user's address. The service will return success or failure. The Oracle book is available in services provider called Portsudan_library, Portsudan_library has the following service: Books.Computer.DataBase.Buy () for buying the Database books. This service needs parameters Book name , publish year , author name ,creditcard_number and user-address. The service will return success or failure. He wants to buy also Windows2000 magazine from any one of the service providers that introduces the minimum price. The services providers available are Khartoum_library, Portsudan_library, Juba_library and Madani_library.

The following table describes the services which are included in the service providers that the mobile agent will visit.

Services Providers	**Services**	**Input parameters**	**Output parameters**
Khartoum_library Portsudan_library Juba_library	Books.Computer.Language.Buy() Books.Computer.Database.Buy()	Book_name publish_year author_name creditcard_no user_address	Result {success or failure }
Madani_library	Books.Computer.info()	Book_name publish_year author_name	Price
	Magazine.Computer.info()	Magazine_name Magazine_date	Price
	Magazine.Computer.Buy()	Magazine_name Magazine_date creditcard_no	{success or failure }

Scenario Flowchart



According to the above scenario, the representation in MASL is as following:

```
create agent   Buy_Items  as

begin

        Define my_money, price_ver ,  magazine_price , min_price    as number ;

        Define  result_message, location_name   as string ;

        My_ money = 1000 ;


 move to  Khartoum_library    ;

   /* the following specification for buying  JAVA Book and get information

      About  windows2000 magazine price in Khartoum_library    */

        call service  Books.Computer.info() with values

                book_name  = 'JAVA' , publish_year = '2001' ,

                author_name = 'Jone Smith' in  Khartoum_library

                set result as  price_var =  price ;

if  price_var  <=  my_money  then

begin

                call service  Books.Computer.Language.Buy() with value

                        book_name  = 'JAVA' , publish_year = '2001' ,

                        creditcard_no =  '121323445' , author_name = 'Jone Smith'

                        user_address = ' sudan , kartoum  P. O. Box 3439 '

                        in  Khartoum_library

                        set result as  result_message  =  result ;

                if  result_message = 'success' then

                        my_money = my_money – price_var ;

end;
```

call service Magazine.Computer.info() **with values**

Magazine_name = ‘windows2000’ ,Magazine_date = ’01-01-2002’

In Khartoum_library **set result as** magazine_price = price ;

Location_name = Khartoum_library ;

Min_price = magazine_price ;

Move to Portsudan_library **;**

/* **the following specification for buying ORACLE Book and get information About windows2000 magazine price in Portsudan_library** */

call service Books.Computer.info() **with values**

book_name = ‘ORACLE’ , publish_year = ‘1999’ ,

author_name = ‘Peter Ward’ **in** NewYork_library

set result as price_var = price ;

if price_var <= my_money **then**

begin

call service Books.Computer.Database.Buy() with value

book_name = ‘ORACLE’ , publish_year = ‘1999’,

creditcard_no = ‘121323445’ , author_name = ‘Peter Ward’

user_address = ‘ sudan , kartoum P. O. Box 3439 ‘

in NewYork_library

set result as result_message = result ;

if result_message = ‘success’ **then**

my_money = my_money – price_var ;

end;

call service Magazine.Computer.info() with values

Magazine_name = ‘windows2000’ ,Magazine_date = ’01-01-2002’

In NewYork_library **set result as** magazine_price = price ;

If min_price > magazine_price **then**

begin

Location_name = NewYork_library;

Min_price = magazine_price ;

end ;

move to Madani_library**;**

/* **the following specification get information**

About windows2000 magazine price in Madani_library */

call service Magazine.Computer.info() with values

Magazine_name = ‘windows2000’ ,Magazine_date = ’01-01-2002’

In Dubai_library **set result as** magazine_price = price ;

If min_price > magazine_price **then**

begin

Location_name = Dubai_library ;

Min_price = magazine_price ;

end ;

move to Madani_library ;

/* **the following specification get information**

About windows2000 magazine price in Madani_library */

call service Magazine.Computer.info() with values

Magazine_name = ‘windows2000’ ,Magazine_date = ’01-01-2002’

In **Madani_library set result as** magazine_price = price ;

If min_price > magazine_price **then**

begin

Location_name = **Madani_library** ;

Min_price = magazine_price ;

end ;

/* **the following specification for buying windows2000 magazine from suitable place */**

if Location_name <> **Madani_library then**

Move to Location_name ;

Call service Magazine.Computer.Buy() with values

Magazine_name = ‘windows2000’ ,Magazine_date = ’01-01-2002’

In **Madani_library set result as** result_message = result;

if result_message = ‘success’ **then**

my_money = my_money – min_price ;

move to Home ;

end

3.5 SMAG Architecture

This chapter details concepts of the research model architecture. It describes the SMAG system components. It shows all components in integrity view and details the duties of each. The model consists of several components. Each one has specific roles in the paradigm. They exchange the services among themselves to achieve their tasks. The following components are included in the model:

- Mobile agent.
- Agent Virtual Machine
- Mobile agent Base.
- Client Node.
- Service Provider.
- Controller.
- System Administrator.

Figure 3-1 shows a general view of the SMAG components.

3.6 Mobile agent

A mobile agent is an entity which represents the user's requirements. It has an ability to move from a place to another under its self control. All mobile agents have the same structure but their behaviours are different and according to the user specification. The SMAG system generates the mobile agent in an isolated place (in the mobile agent base) from users for security reasons. The generation is done by the generator which a mobile agent base hosts. The generator unit accepts only the syntax of the MASL. This language prevents any malicious task so, the generator generates only safe code. This is a big achievement to the security area. The following paragraphs describe the mobile agent elements:

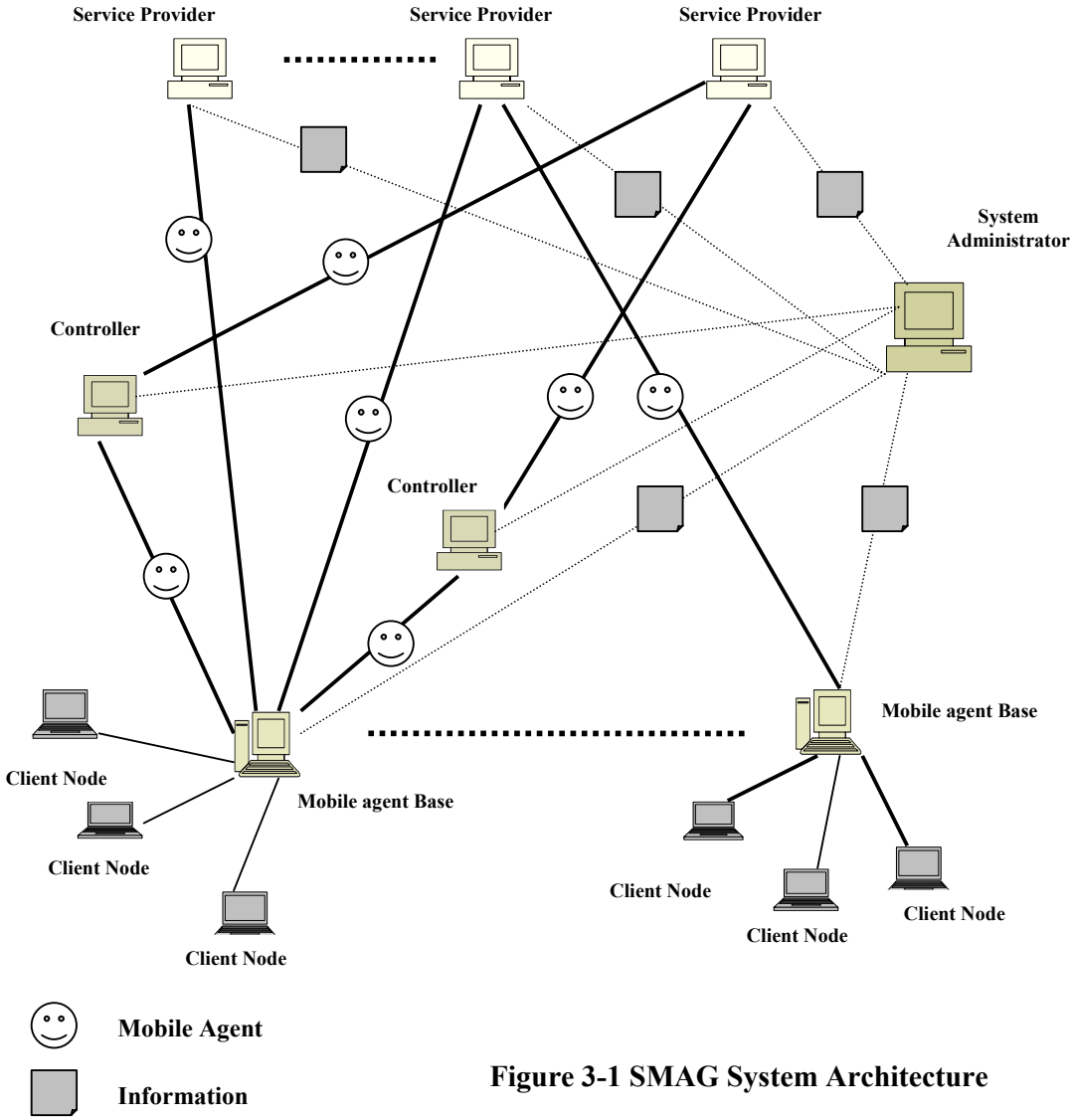


Figure 3-1 SMAG System Architecture

3.6.2 Intermediate Code Generation (ICG)

The SMAG system generates behaviour of the mobile agent according to a user specification. This element represents the behaviour of the mobile agent before it starts its journey. Each place that the mobile agent will visit has ICG.

3.6.2 Identification Information

This unit contains identification information about a mobile agent. The system provides the mobile agent with this information before it starts it's a journey. This information is about, for example, a mobile agent owner, a unique mobile agent name, a base name, an itinerary table …etc.

3.6.3 Source Data Space (SDS)

A mobile agent needs some data as initial values before it starts a journey and these data represent code representation inputs.

3.6.4 Destination Data Space (DDS)

Results of executing code representations are stored in this element.

3.6.6 SubAgent

Before a mobile agent starts its journey, the mobile agent base partitions the mobile agent behaviour into groups of code representations according to the place of the execution. These codes are ICG. Each group is called SubAgent. So, each mobile agent has a SubAgent in each place that it will visit. In addition, the SubAgent has its own ICG, SDS and DDS. This structure also helps the mobile agent protection area. According to the mobile agent tasks, Keylets approach partitions the mobile agent as components [50]. The disadvantages of this approach are the large number of transactions related to the keylet and a host may not be willing to support the increased computations.

As detailed above, the mobile agent has many elements. Figure 3-2 shows the architecture of the mobile agent.

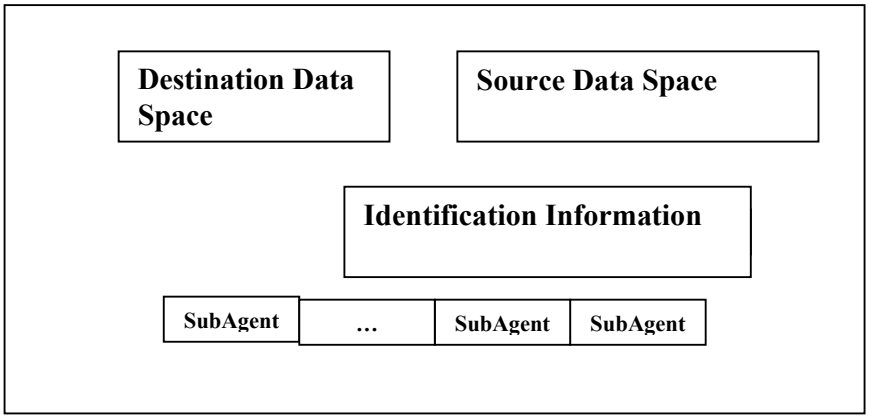


Figure 3-2 Mobile agent Architecture

3.7 Agent Virtual Machine (AVM)

This unit interprets intermediate code generations (ICGs) that are carried by a mobile agent. It takes an ICG in each place as inputs and executes it. This unit can execute, for example, expression statements, if statements, mobility statements, call service statement …etc. Each service provider and mobile agent base hosts the AVM for the following reasons:

- Size of the mobile agent will be minimized because the mobile agent will carry only ICGs and some information.
- Any development in the MASL will be accepted because the size and performance of the mobile agent will not be affected by that.
- From security perspective, it is better to allow the mobile agent to carry ICGs and service providers execute them. So, the mobile agent has not a full control of execution processes because the engine of execution is hosed by the service

provider. On the other hand, the service provider has not a full control of execution processes because the ICGs are carried by the mobile agent.

In addition, this idea is a new for the mobile agent systems. It is used by Java programming language [21]. The Java Virtual Machine (JVM) is hosed by a machine that can execute classes from the same machine or from a different one and that gives Java the power of the security. So, this approach is acceptable in a wide area of distributed system applications.

3.8 Mobile agent Base

A mobile agent base is one of the system components. The mobile agent starts and finishes its journey in this component. By using MASIF standardizations, this component is built [55]. For security reasons, it must be in an isolated place to be out of the users' control. The following points describe the duties of the mobile agent base:

- User Registration: Any user who wants to manipulate with the system must first make a registration to get ID services and authorization to have access to different parts of the system.
- Mobile Agent Generation: When a user submits his / her specification, the mobile agent base takes this specification and generates the mobile agent behaviour.
- Mobile Agents Management: It allows users to manage their mobile agents, for example, list the mobile agents, get information about their status, send and receive messages to and from the mobile agent.

- Host Services: The mobile agent base can host services and provide them to the visiting mobile agents. So, the mobile agent can work as a service provider (this feature is optional).
- Member of System Administrator: A mobile agent base must be registered in the system administrator to be a member of the system in order to communicate with other system parts.
- Security Requirement: It has a security unit which encrypts and decrypts SubAgents and the mobile agents in order to protect them.

As mentioned above, the mobile agent base has basic roles in the system. This structure lets the system to be scalable in a big geographical area by adding nodes of a mobile agent base under the control of the system administrator. To achieve its duties, the mobile agent base has the following units:

3.8.1 Generation Unit

A mobile agent base uses this unit to generate user specification requirements as mobile agent behaviour. It checks syntax of this description according to MASL syntax and it generates ICGs that are represented in SubAgents.

3.8.2 Communication Unit

Through this unit, a mobile agent base dispatches and receives the mobile agents and communication messages.

3.8.3 Security Unit

All security requirements for the mobile agents and the communication messages are done through this unit. It has an ability to encrypt and decrypt behaviour representations by different secret keys and a mobile agent by another secret key. The next chapter describes the security mechanisms.

3.8.7 Database Unit

All information about users, mobile agents, services, service providers and system information are stored in this unit. A database unit consists of several tables.

3.8.8 Services Unit

This unit is optional by adding to a mobile agent base. This unit provides services to visiting mobile agents.

3.8.9 Users and Services Registration Unit

This unit receives and stores users and services information in the database unit. The registration is done through forms. This unit prevents any user to use the system without first registration and getting a user's account. The services will not be added to the services unit without first registration.

As mentioned above, the mobile agent base has many items and each item has a specific role, figure 3-3 shows these items in one view. In addition, this component is provided by the AVM to manipulate the mobile agent.

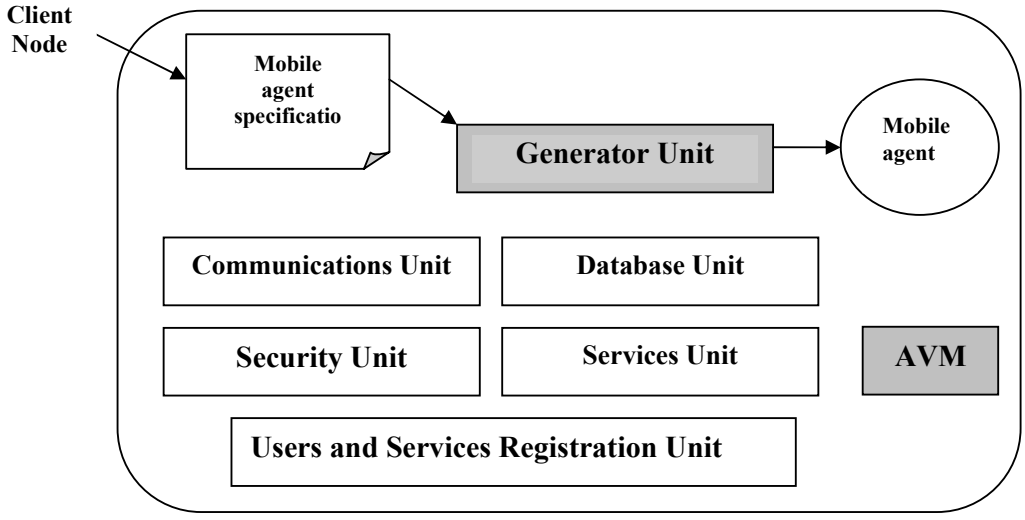


Figure 3-3 Mobile agent base Architecture

3.9 Client Nodes

This component allows users to interact with the system through the mobile agent base. The users can build and manage their mobile agents through this unit [59]. The Client Node is represented as internet site. The mobile agent base represents different operations as web forms. A user uses these forms to perform many operations such as:

- Registration: user gets ID subscription in order to own an account and to have an access to the system through an ID and a password.
- Describing a mobile agent: if the user has an account, he/she can represent mobile agent behaviour by writing a specification through MASL.
- Managing a mobile agent: a user can do many operations on the mobile agents such as:
    - Altering a mobile agent specification.
    - Deleting a mobile agent from the account.
    - Displaying a mobile agent context.
    - Getting status of mobile agents.
    - Sending and receiving communication messages to and from the mobile agents.

3.10 Service Provider

A service provider is one of the system components. It hosts the service resources which the mobile agents can request. The mobile agent can visit at least one service provider to accomplish its tasks. This is the main feature of the mobile agent systems which allows the mobile agents to travel through multi service providers. The main goal of the service provider in the model is to provide services to visiting mobile agents securely. Any organization can represent its businesses role as services. So, the

SMAG system is suitable for different types of applications. Multi mobile agents can concurrently visit one service provider and serves each of them without any defect in performance because the service provider provides services in isolated executing threads [2]. Each thread represents a place with a collection of services. The service provider supports many operations in order to achieve its goals such as:

- Receiving and Dispatching the mobile agents:

  A service provider has an ability to receive and store a visiting mobile agent on its storage unit. After a mobile agent completes its execution, the service provider dispatches the mobile agent to the next service provider according to an itinerary table or return home.

- Services Representation:

  A service provider is a container of services which can provide visiting mobile agents with them. Each service must follow a standard form of services representation which is described in services protocols in this chapter. Each service has code behaviour and a service interface which contains a service name, service input parameters and output parameters. The mobile agents can call these services through services interface.

- Receiving and Sending Communication Messages:

  A service provider has a unit that work post office. Any mobile agent that visits the service provider will get a mail account. This unit can receive messages from different parts of the system and delivers them to a target mobile agent. A message must be managed securely. Also, it can send messages from the mobile agent to the different parts of the system [2].

As detailed above, the service provider has many functions. The following section describes its elements and duties of each element:

3.10.1 Places Unit

Every service provider has a places unit. It represents a group of places. Each place can access the service provider resources. Each place can also host one mobile agent at a time. The place consults the security manger before it decides to serve the mobile agent. The place has an isolated execution thread. So, many places can work concurrently.

3.10.2 Communication Unit

This unit defines the connections protocol. Through this unit, the service provider is connected with several parts of the system. This unit has a listener using TCP protocol as platform. It defines an IP address and a port of the service provider machine. The listener must be active in order to be able to receive mobile agents and communication messages.

3.10.3 Mobile Agents and Messages Builder Unit

A service provider receives mobile agents and communication messages as data stream through communication unit. These data need to deserialize in order to rebuild the mobile agent or message object [67, 68]. This unit does such a task. Also, when the mobile agent decides to leave to another station or sends a message, this unit generates the serialization data from the mobile agent or the communication message object. After that, these data are sent through communication unit.

3.10.4 Database Unit

All information about a service provider is stored in this unit. Such as, system configuration, service information, communication information and security data. Every service provider has database unit. The database unit stores all information in local database. Only the service provider has authorization to access this unit.

3.10.7 Services Controller

This unit works as services proxies. Each place has a copy of this unit. The visiting mobile agents request the services from this unit. This unit can execute multi services to increase the performance. This unit consults the security unit before answering the request of the mobile agent [59].

3.10.8 Security Units

Each service provider has a security unit. This unit represents all security requirements for a service provider such as, cryptography services. Figure 3-4 displays all items in the service provider.

The service provider hosts the AVM to deal with visiting mobile agents.

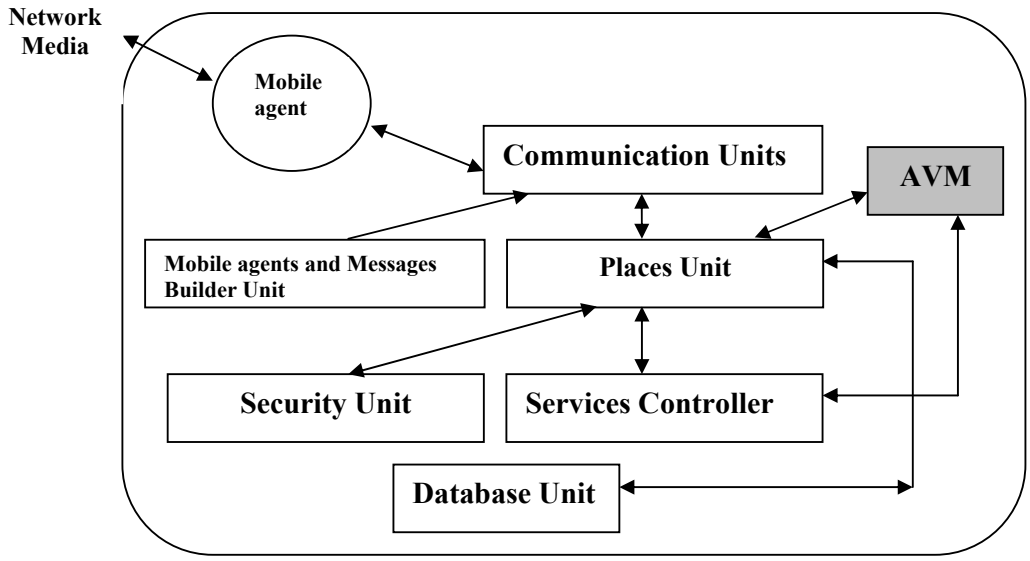


Figure 3-4 Service Provider Architecture

3.11 Controller

A controller is one of the system components. It is a trust place and provides the SMAG system with a big support in the security area. Each controller has a unique name in the SMAG system domain. Any service provider must be controlled by a controller. The controllers in the system play important roles as the following:

3.11.1 Protecting a Mobile Agent

Before a mobile agent travels to any service provider, it may first visit a controller of a target service provider, if a user requests that. The controller keeps a copy of the mobile agent and some security information about it. After that, it dispatches the mobile agent to the target service provider. In case the target service provider is out of service, the controller can help the mobile agent to skip it in order to continue a journey or return home. After the mobile agent completes its duties in the service provider, it returns to the controller again. By using the security information of the mobile agent, the controller verifies any attack occurred to the mobile agent while it is hosted by the service provider or not. In case any attack occurs or the service provider refuses to return the mobile agent to the controller, it will replace the copy version by the current version and allow the mobile agent to continue its journey. Service providers register themselves by using their names and their controller names and provide the system administrator by this information. The security service of this point will be detailed in the next chapter.

3.11.2 Generating Dynamic Behaviour

A mobile agent can detect interesting new places during its journey. To visit these places, it must have behaviour to execute on them. So, the controller can achieve this task. The generation task will be performed safely because the controller is a trusted place.

3.11.3 Garbage Collection Service

When a controller generates a dynamic behaviour, a mobile agent size will be large. So, this is a big problem. To solve this problem, the controller can be provided with a garbage collection unit that can free parts of the mobile agent that are not used in the future journey. After this unit specifies the free parts, the controller can use them to store the dynamic behaviour. Some programming language like Java uses this mechanism to manage its programmes [30].

To achieve the above services, each controller has the following components:

3.11.4 Mobile Agents Storage Unit

This unit can store mobile agents that want to visit a service provider.

3.11.5 Security Unit

This unit authenticates visiting mobile agents. Also, it keeps a copy of each mobile agent and some security information. It deletes this information after the mobile agent leaves it.

3.11.6 Dynamic Generation Unit

This unit provides a mobile agent with dynamic behaviour that allows the mobile agent to visit new interesting places. The generation of this behaviour will be done according to the user's request.

3.11.7 Garbage Collection Unit

A mobile agent visits many places during its journey. In each place, it executes some tasks. After it completes its duties in the place, parts of its behaviour will be unused in a future journey. So, this unit specifies these parts to be reused by the dynamic generation unit.

3.12 System Administrator

A system administrator controls all system components. A mobile agent base or service providers must make a first registration before becoming members of the system and they must get first subscription IDs. The mobile agent base or the service provider could not perform any operation or deals with other parts in the system without getting this ID from the system administrator. The ID is unique. So, the mobile agent base or the service provider has a unique ID. The ID consists of two parts. Part one, classification code represented to define the type of component. The mobile agent base classification code is "MB". The service provider classification code is "SP". In case the mobile agent base works as a service provider besides its job, the classification code is "MP". Part two is the sequence number that is followed by the classification code. The combination of the two parts must be unique. The system administrator keeps the tracks information about these components through ID's. The following points describe the main jobs of this component:

3.12.1 IDs Generation and Registration

As mentioned above, it generates a unique ID for a new member. The new member can use the ID for transaction operation over the SMAG system. The mobile agent base uses the ID to generate other unique names for their mobile agents. So, any component can distinguish the mobile agent's base from its name. Also, a message has a unique ID which is generated from its owner ID either the service providers or the mobile agent base. The IDs help to authenticate the objects according to the IDs whether the receivers can decide to accept them or not. The mobile agent base and service providers send some registration information such as, IP addresses, port

numbers, etc, in order to get the ID. The system administrator stores this information in a database.

3.12.2 Log Information Unit

This unit is very important because the information about the system is stored through this unit. It works as reference to all components. It also works as a system monitor. A mobile agent base or a service provider has a special and a secure section. This section stores the information related to a member, such as information about its communication messages, security information, mobile agents and … etc. So, the mobile agent base could get any information about their mobile agents' statuses or communication messages through this unit. When the mobile agent arrives to a new destination the receiver notifies the system administrator to update the information record about the mobile agent. Also, this is done to the communication messages.

3.12.3 Certification Authority Unit

This unit issues and publishes digital certificates for different parts of the SMAG system.

The digital certificate consists of the following fields:

- Owner Name.
- Issue Date
- Expire Date.
- Public Key (PK) of the mobile agent base or the service providers.

Figure 3-5 list these items in one picture.

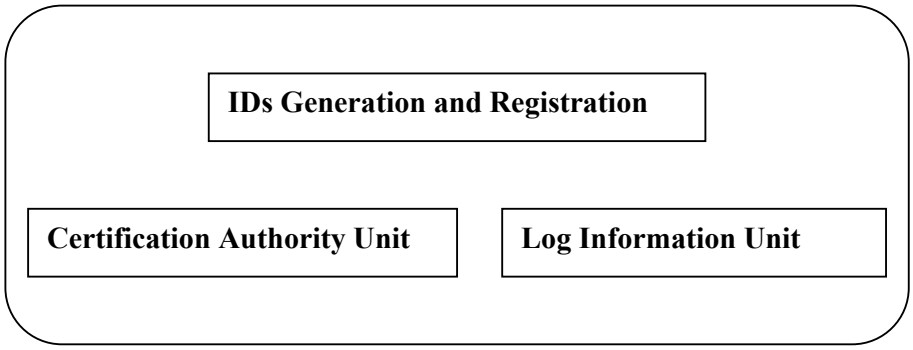


Figure 3-5 System Administrator Architecture

3.13 Service

Service providers provide visiting mobile agents with several services. The service represents a specific function that the mobile agent wants. So, each service provider contains a collection of services. When the mobile agent arrives to the service provider, it requests some services. The service provider serves the mobile agent through its services. The execution of the services is done under full control of the service provider for security reason. In the SMAG system, the services are divided into two types that are called system services and application services. The system services represent the system function requirements such as dispatching, receiving, storing mobile agents, manipulating with communication messages and so on. So, the basic functions of the system in the service provider are achieved through the system services. The system services are designed and implemented in the SMAG system. Copies of these services are distributed to all service providers that are connected to the system and to the mobile agent base. The application services represent business roles that service providers introduce. They are different from a service provider to another. They depend on application type that is introduced by the service provider. For example, some services for selling goods, distributing information, controlled devices and so on. So, a designing and implementing of the application services

depend on their objectives of the applications. But, their structure in each service provider is the same. Also, these services represent the gates to resources of the service provider. Through this type of the services, visiting mobile agents can access resources. The service provider supports different operations on application services such as updating, deleting and inserting services. Through a services controller, the service provider introduces its application services. The service provider must register its application services in system administrator and provides it with full description about them before they can be used.

The structure of the service consists of different parts. But, the behaviour may be different from a service to another. The following sections describe these parts:

3.13.1 Service Name

Each application service has a unique name on its services provider. The name consists of alphabet characters. Through the service name, the mobile agent can request that service. The service provider can execute the service by using its name. Also, service providers distribute their application services to other part of the SMAG system by using their names through the system administrator. The name of the service doesn't depend on the location. If two or more of the service providers use the same name of the service, the system can distinguish between them through the Identification name of the service providers that distributes the service. The name of the service is generated by the service provider owner.

On the other hand, the system services have names. The name of the specific service is unique in each part that uses the service. For example, the dispatching service has the same name in each service provider or a mobile agent base.

3.13.2 Service Parameter

A service needs two types of parameters, input parameters and output parameters. The Input parameters represent inputs data required by the service. According to these data, the service executes its code. The output parameters represent container units that are used to carry out the results of the service executed. Through the parameters, the service communicates with the external world. The content of the input and output parameters may differ from call to another. So, the behaviour of the service is constant. But, the results of the service are variables. From a programming view, each parameter has specific type. For example, character, integer, object, and so on. Some services have not input parameters, output parameters or both.

3.13.3 Service Body

Any service has specific function. The function of the service is represented by a service body. The service body also is called service behaviour. It consists of a list of representation processes. These processes are different from a service to another. They depend on the service that is achieved through it. The service body may use input parameters as data input, after that it executes the processes and puts the results in output parameters.

3.13.4 Service Protocol

Only services owners' can execute their services. For example, if the mobile agent requests a service from the service provider, it can not execute this service directly. But, it can execute it by using a call representation. The call representation contains a service name, input parameters and output parameters. The service provider takes the call representation to execute the service through service controller.

3.14 Communication Message

A communication message is used in the SMAG system to allow the system components to communicate with each other. The mobile agent owners can send communication messages to their mobile agents. Also, the mobile agents can send communication messages to their owners'. Moreover, the mobile agent can communicate with each other. The SMAG system uses asynchronous communication mode [2, 59]. So, a sender of a message does not wait until the reply return. A message is represented as object which contains message-id, a sender's and receiver's addresses type of the message and a message context. Each communication message has message-id that is used as reference to a message sender. An address of the sender and the receiver represents for example, a mobile agent id and a mobile agent owner id, in case the message is sent from a mobile agent to its owner. The type can be one of the following:

- Get_Information type: This type is used to get information from entity to another. It allows mobile agent owners to communicate with their mobile agents at runtime. It also allows the mobile agents to communicate with each other. Moreover, it allows the mobile agents to send some information to their owners during their journeys.
- Action type: This type allows only mobile agent owners to communicate with their mobile agents. Through this type, they can terminate, notify and manipulate with their mobile agents.

A message context represents information of the message and it is different from a message to another. Each message has a method that uses for authenticating the receiver before providing it with message context.

3.14.1 Communication message protocols

This section describes some roles for how and when the communication messages achieve their works in the SMAG system:

3.14.1.1 Communication between Mobile Agent and Owner

Through the mobile agent base, an owner can communicate with his/her mobile agents. Also, a mobile agent sends messages to its owner. There are two types of messages when the owner is the sender. The first, static message which is created before the mobile agent starts its journey. This type is built according to an owner's specification of the mobile agent by using notifying statement in the MASL. The time of sending this type of the messages is done when the mobile executes its processes. The second type is allowed to the owners and the mobile agents. This type is built after the mobile agent starts its journey. The communication message is sent to the mobile agent through controllers or service providers when the sender is the owner. Each controller or service provider has a message box. Before the mobile agent leaves, it checks the mail box if it has a message or not.

3.14.1.2 Communication between Mobile agents

By using Set_information and Get_information statements in the mobile agent description, a mobile agent owner lets his/her mobile agent to exchange its information with other mobile agents. Each service provider has specific share area of memory[60, 2]. Through this area the mobile agent can put or get information. The information is represented as object. Each object contains mobile agent identification information and information text.

3.15 Mobility Protocol

A mobile agent in the SMAG system can visit multi destinations in one journey by using mobility mechanism. The mobile agent requests its current location to dispatch it to the next place. This action is done when executing the mobility statement "**move to destination-name**". The mobility mechanism uses TCP protocol as platform of its private protocol. The mobile agent visits service providers through their controllers if the user requests that. So, if the mobile agent wants to visit a service provider A, it visits first its controller and it goes to A. After that, it returns to its controller before it travels to another service provider. The controller keeps a copy of the mobile agent before it sends it to one of its service providers. If any attack occurs by the service provider, the controller drops that affected object replaces it by the version copy. Also, it can visit directly the service provider. By using an itinerary table and a mobility service, the mobile agent starts its journey. Before the mobile agent moves to first place, it is converted to binary data by using serialize class in .NET framework and after that, it takes encryption from. The next chapter details the encryption mechanisms. A sender opens TCP connection line by using an IP address and a port number of the receiver. Through this line, the binary data of the mobile agent is sent to the destination. The receiver creates the listener to receive the binary data. By using the Deserialize class in .NET framework, the receiver recreates the mobile agent from the binary data. This list of operations is repeated when the mobile agent moves from a place to another. There are additional security tasks added to the list. The next chapter details the security tasks.

3.16 Scenario

This section presents a brief scenario that shows a driving of the several actions in the SMAG system. The purpose of this section is to give the reader a general view of the model processes.

1. **In case a user has no account in the system, the user requests a mobile agent base to get a new access account.**
2. **The user uses Client Node (Browser) in order to input his/her specification through the MASL.**
3. **The mobile agent base will authenticate the user before it gives permission to access the system.**
4. **The user enters a specification of the mobile agent.**
5. **The user specification will be submitted to the mobile agent base.**
6. **The mobile agent base uses its generator to generate the mobile agent behaviour.**
7. **The generator generates SubAgents.**
8. **The mobile agent base encrypts each SubAgent by using different secret keys.**
9. **It converts the mobile agent into binary data.**
10. **Also, it generates a signature data from the binary data.**
11. **The user gives the order to the mobile agent base to dispatch the mobile agent.**
12. **The mobile agent base encrypts the mobile agent by a receiver secret key.**
13. **The mobile agent base notifies the system administrator to create a log file for the mobile agent in order to keep track of all the actions of the mobile agent during the journey**

14. **Moreover, it sends the binary data and the signature data to the first host according to the user specification**.

15. In case the next host station is a service provider:

15.1When the mobile agent arrives to a service provider, the service provider provides the system administrator with that to update the information in the mobile agent log records.

15.2 The service provider decrypts the binary data by using its secret key.

15.3 Authenticating the binary data by using the signature data.

15.4 Deserializing the binary data to reconstruct the mobile agent.

15.5 It specifies a SubAgent that will be execute on.

15.6 Decrypting the SubAgent.

15.7 Executing the SubAgent by using the agent virtual machine (AVM).

15.8 Encrypting again the SubAgent.

15.9 Converting the mobile agent again into binary data.

15.10 Encrypting the binary data by using secret key of the next station.

15.11 Sending the binary data to the next station.

16. In case the next station is a controller

16.1When the mobile agent arrives to a controller, the controller notifies the system administrator with that to update the information in the mobile agent log records.

16.2 The controller decrypts the binary data by using its secret key.

16.3 Deserializing the binary data to reconstruct the mobile agent.

16.4 Authenticating the binary data by using the signature data.

16.5 Saving the mobile agent and security information in a queue.

16.6 Converting the mobile agent again into binary data.

16.7Encrypting the binary data by using a secret key of the service provider.

16.8 Also, it generates a signature data from the binary data.

16.9 Sending the binary data to the service provider.

16.10 In case the mobile agent returns from the service provider:

16.10.1When the mobile agent arrives to the controller, it notifies the system administrator with that to update the information in the mobile agent log records.

16.10.2 The service provider decrypts the binary data by using its secret key.

16.10.3 Deserializing the binary data to reconstruct the mobile agent.

16.10.4 Authenticating the binary data by using the signature data.

16.10.5 By using the mobile agent, the mobile agent in the queue and security information, the controller checks any attack that may occur to the mobile agent.

16.10.6 Selecting one of the mobile agents according to the result of the pervious step.

16.10.7 Converting the mobile agent again into binary data.

16.10.8 Encrypting the binary data by using the secret key of the next station.

16.10.9 Sending the binary data to a next station.

16.11 In case the service provider prevents the mobile agent from returning to the controller:

16.11.1 If the mobile agent stays more than a specific time in the queue:

16.11.1.1 The controller converts the mobile agent again into binary data.

16.11.1.2 It generates a data signature from the mobile agent.

16.11.1.3 Encrypting the binary data by using the secret key of the next station.

16.11.1.4 Sending the binary data to the next station.

16.11.1.5 Saving the mobile agent name in a dead list to kill the old copy if it comes after that from the service provider.

17. In case the next station is a mobile agent base:

17.1 When the mobile agent arrives to the mobile agent base, it notifies the system administrator with that to update the information in the mobile agent log records.

17.2 The mobile agent base decrypts the binary data by using its secret key.

17.3 Deserializing the binary data to reconstruct the mobile agent.

17.4 Authenticating the binary data by using the signature data.

17.5 Decrypting all SubAgents by using its secret keys.

17.6 Extracting the results from the SubAgents and other variables.

17.7Creating report about the mobile agent journey.

17.8 Notifying the user by that.

18. End.

Chapter 4

Security of SMAG System

4.1 Introduction

This chapter explains the security requirements for the SMAG system. The mobile agent is an executable code that is executed in different places. The places need to be trust on all visiting mobile agents. Also, the mobile agent owners need a secure journey to their mobile agents. This gives the importance to the security area in developing the mobile agent systems.

4.2 Cryptographic Mechanisms

Cryptography is used in security fields. The concept of using has a long history. One of the earliest cryptographic systems, Julius Caesar sent military messages to his generals [70]. The cryptography mechanisms are used to make secure communication among different parts. It has many concepts:

- Encryption: This process converts a message from readable to be unreadable by using a key. The key is a numerical value used by the encryption process to change the information.
- Decryption: This is opposite to the encryption process. It converts the encrypted message to its origin by using the same key or another key (depend on the mechanism).
- Algorithm: The well defined set of roles that are used to encrypt and decrypt the message. It is represented in mathematical function.

In cryptography world, the message that needs to be secured is called plaintext. After the message is encrypted it is called cipher text. The cryptography mechanisms are used in wide areas of the data communication fields. The researcher continually improves the cryptography mechanisms to be more trustful and powerful. There are many cryptography mechanisms as follows [70]:

4.2.2 Symmetric Encryption

A symmetric encryption is one of the cryptography mechanisms. It uses the same key for encrypting and decrypting. The key is called a secret key. The users who exchange data and use this mechanism must keep this key securely. The algorithm that is used in this mechanism is called a Secret-Key Algorithm. It uses the key to encrypt the message and the same key to decrypt the message. There are some popular secret-key algorithms and key size such as [69]:

- RC2 – 64 bits.
- DES – 64 bits.
- 3DES – 192 bits.
- AES - 256 bits.
- IDEA – 128 bits.
- CAST -128 bits (CAST256 uses 256 bits key)

Figure 4-1 shows the encryption and decryption processes.

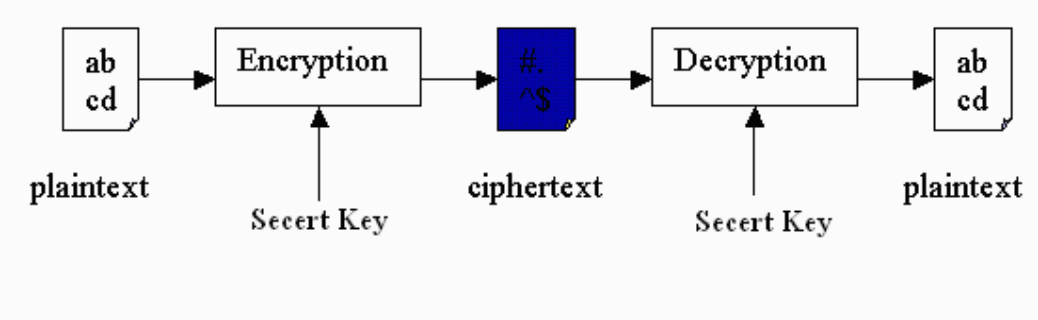


Figure 4-1 Symmetric Key Encryption

4.2.2 Asymmetric Encryption

Asymmetric Encryption is also another mechanism in cryptography. It uses two keys that are mathematically related and impossible to distinguish one from the other. The first key is called a public key and the second is called a private key. This mechanism uses one key to encrypt the message and the other key to decrypt the message. The receiver of the message generates the two keys (Private Key and Public Key) and he/she distributes the public key only to all senders who want to send secure data to the receiver. Also, he/she keeps the private key. The sender uses the public key of the receiver and the algorithm of the mechanism to encrypt the message. By using the private key, the receiver decrypts the message. So, no one can decrypt the message without the private key. The most universal algorithm of this mechanism is called RAS. The Asymmetric Encryption has a problem in the performance because the processing requires intensive CPU and this is a problem when many simultaneous sessions are required. So, the symmetric encryption is better than the asymmetric encryption in some areas for performing reason. Figure 4-2 shows the encryption processes in this mechanism.

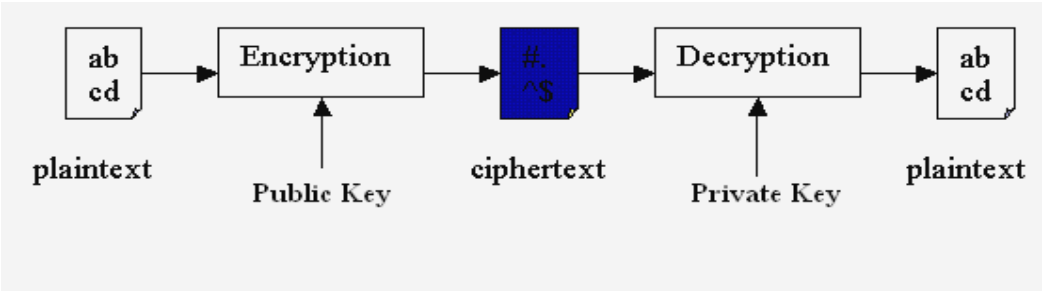


Figure 4-2 Asymmetric Encryption

4.2.3 Digital Signature Encryption

This mechanism uses the asymmetric encryption in a different way. The sender uses his/her private key to encrypt the message. Everyone can decrypt this message by using only the sender's public key. The message signed by the sender

is called digital signature. The receiver ensures that message comes from the expected sender [70, 69]. Figure 4-3 shows this process.

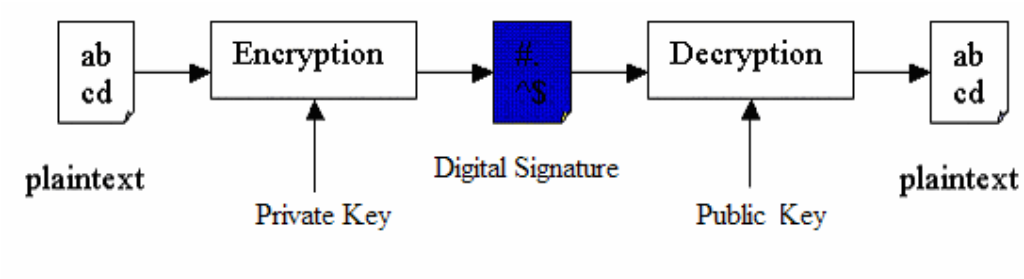


Figure 4-3 Digital Signature Mechanism

4.2.4 Hashing Algorithm

A hash function is mathematical function that is used to generate message-digest from a message [69]. It uses the original message as an input and the output is a message-digest. The message-digest is a unique for the message (fingerprint of the message). Also, this function is called one way hash function. The asymmetric encryption can use this function to let the cryptography mechanism to be more powerful and reliable by following these steps:

1. The sender uses the hash function to generate a message-digest from the message that is to be sent securely.
2. The sender uses his/her private key to sign (encrypt) the message-digest (digital signature).
3. The sender uses the receiver public key to encrypt the message.
4. The sender sends the encrypted message and the digital signature of the message-digest to the receiver.
5. By using the sender public key, the receiver decrypts the digital signature to get the message-digest.
6. The receiver decrypts the encrypted message by using his/her private key.

7. By using the hash function, the receiver calculates the message-digest of the decrypted message.
8. By making a comparison between the two message-digests, the receiver will know if they match or not.

All these steps guarantee that message will be sent correctly and securely to the receiver. The receiver also guarantees that the message is from a well-known sender. Figure 4-4 shows the above steps.

4.2.6 Digital Certificates and Certificate Authorities

A digital certificate binds the details of the individual or organization to their public key. The more acceptable and widely used format for digital certificate is the X.509 standard. The communication between the sender and the receiver is achieved by accessing the certificate of each other. So, each one gets the public keys of the other from the digital certificates. The Certificate Authorities (CAs) issue the digital Certificates and they are known as trusted third party. The CA idea prevents any misuse of the digital certificate, for example, a false creation of the certificate. With the increases in communications, the number of CAs is also increases.

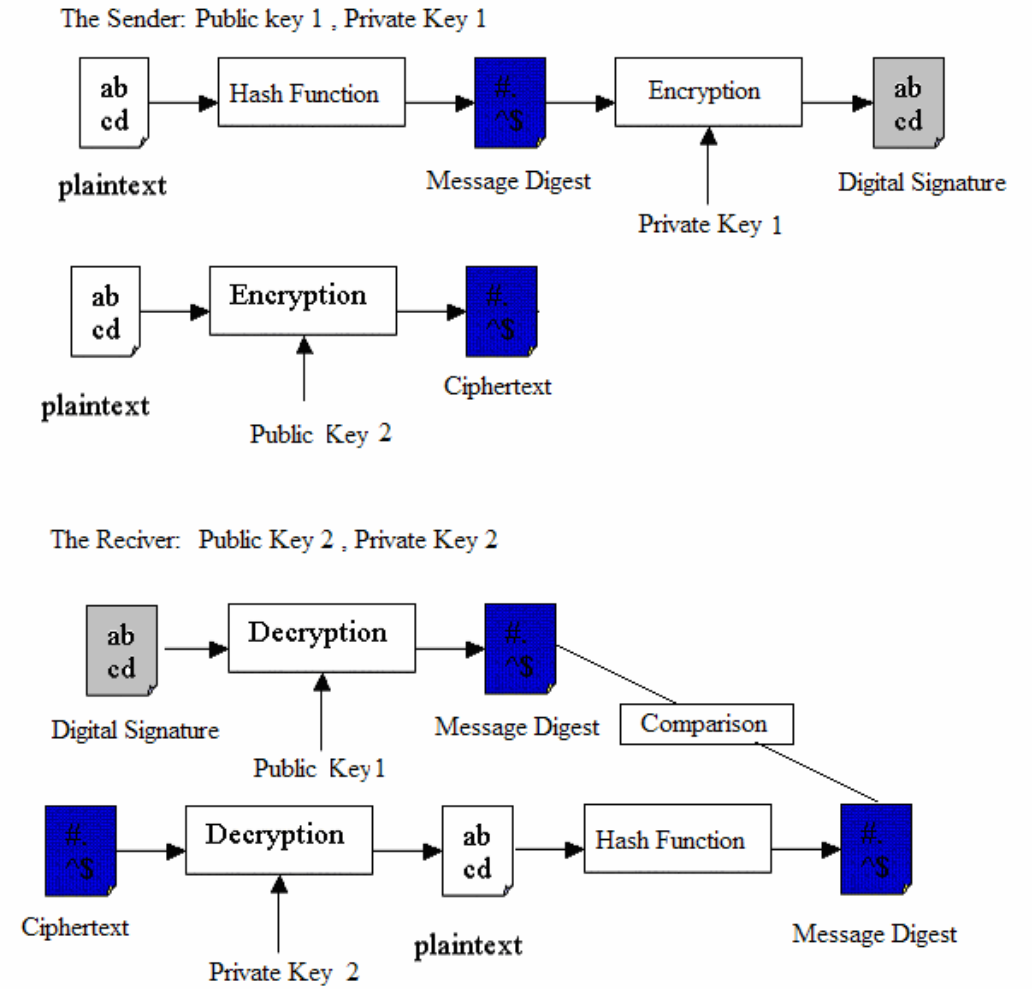


Figure 4-4 Hash Function

4.3 Security Model

In this thesis, a new security model which contains different mechanisms to protect the SMAG system in all phases is implemented and designed. Most current security mechanisms detect attacks [41, 26]. But, the model mechanisms prevent such attacks. The security model protects the mobile agent from any risks. It also protects all service providers from a malicious mobile agent. Moreover, it provides a secure transferring of the mobile agents and communication messages. The following mechanisms are introduced in the security model:

- Generation Mechanism and Agent Virtual Machine to protect a service provider.
- View Security Mechanism to protect a mobile agent.
- Simulator Mechanism to protect data of the mobile agent.
- Constant Data Mechanism to protect read only data.
- Encryption Mechanism , Controller and Safe-Data-digest mechanism to protect transferring of the mobile agents and communication messages.

4.3.1 Generation Mechanism and Agent Virtual Machine

A Generation Mechanism (GM) is a new mechanism which is designed and implemented in the SMAG system. In the most current mobile agent systems, the users handle the mobile agent behaviour. In some of them, the user inherits the basic function of the system and he/she inputs his/her code behaviour for the mobile agent. From security view, the user may put malicious code. So, the service provider faces some security problems. The GM generates the mobile agent behaviour in the mobile agent base. The generation of the mobile agents will be in an isolated place from the user. So, the mobile agent contains a secure code. By using MASL, the mobile agent base generates the mobile agent and this language prevents the user from representing any malicious task. The mobile agent behaviour is represented by intermediate code generations. The agent virtual machine (AVM) can only execute these codes. The service providers and the mobile agent base host the AVM. This mechanism is new in the mobile agent systems because the mobile agent doesn't carry the executed code. Also, it is acceptable in distributed system application. Java programming language uses Java Virtual Machine (JVM) to execute classes that are generated by different machines [2].

4.3.2 View Security Mechanism

A View Security Mechanism (VSM) is a new security mechanism to protect the mobile agent. This mechanism allows the mobile agent to perform its duties securely in each service provider. The main idea of this mechanism is to use cryptography in protection like some current mechanisms but by different ways [26]. The SMAG system supports the VSM to protect the mobile agent. The mobile agent base recognizes the behaviour of the mobile agent in groups. The group is built according to the execution place. The Keylets mechanism proposed to partition a mobile agent according to type of the task [50]. But, this method fails to protect the mobile agent completely. Also, the key revocation is not good in quality. In addition, it requires a complicated mechanism to categorize tasks of the mobile agent. On the other hand, the VSM partitions the mobile agent according to an execution place. This method avoids complicated transactions. The group of processes represents the task that is executed in one service provider. By using VSM, the mobile agent hides its behaviour when it arrives to the service providers except a group that performs on. The form of the mobile agent in this case is called a **view**. The VSM generates different views of the mobile agent during its journey. The view depends on the service providers. Also, each service provider owns at least one view of the mobile agent. Moreover, the service provider can not manipulate with the view that is owned by another. In case the mobile agent has n service providers S1, S2,…, S***n*** in its the itinerary table, the VSM will generate n views V1 , V2 , …, V***n*** during the journey. So, the service provider S1 can manipulate only with V1. S***i*** can manipulate only with V***i*** and so on. Each view will be generated when the mobile agent arrives to the service provider, for example, when the mobile agent arrives to service provide S***i***, the mobile agent behaviour is converted to the view V***i***.

The VSM uses the cryptography mechanisms. Exactly, it uses the symmetric encryption mechanism [71, 72] which is described in this chapter. This mechanism is used rather than asymmetric mechanism for performing reasons. By using this mechanism, the mobile agent base encrypts the mobile agent as various parts (each part represents tasks in specific service provider) by different secret keys. To explain the mechanism, some requirements and assumptions are defined as follows:

The mobile agent base has multi secret keys which are used in encrypting and decrypting the mobile agent. They must be saved securely in the mobile agent base. It provides each part of the mobile agent by a copy of one secret key for encrypting that part after the mobile agent completes its works in that service provider. In case the mobile agent visits **n** service providers, the mobile agent base must have the following secret keys:

Secret keys in the mobile agent base = { H_{Sk1} , H_{Sk2} ,... H_{Skn}}

The keys are generated by the mobile agent base. The mobile agent will take all copies of these secret keys. If the mobile agent visits the service provider multi times, the secret key is used in every time.

Each service provider (S) generates the secret key Ssk, such as:

$S_1 \longrightarrow S_{Sk1}$, $S_2 \longrightarrow S_{Sk2}$, …, $S_n \longrightarrow S_{Skn}$.

Each service provider provides the mobile agent base with a copy of its secret key. So, the following set is stored in the mobile agent base:

$\{S_{Sk1}, S_{Sk2}, \ldots, S_{Skn}\}$.

Before the mobile agent starts its journey, the mobile agent base uses the secret keys of the service providers to encrypt each part of the mobile agent by using one secret key. When the mobile agent arrives to the service provider, it decrypts its part of the

mobile agent by using its secret key which is generated. So, the service provider will not be able to decrypt other parts that specify to other service providers.

The SMAG system generates the mobile agent behaviour from the user description through MASL. After the user writes his/her specification, the generator of the system can specify the execution locations of each group of the tasks. The mobile agent can revisit the service provider many times, so the generator will isolate the tasks in each time. Each group is executed in one location called **SubAgent (SA).** The SubAgent consists of two items: the first item is intermediate code generations (ICGs) for specific service provider in one visit. The second item is data which represents the input data requirements and the output storage variables which are used to store the results of the executions. So, if the mobile agent visits **n** service providers, it will have **n** SubAgents **SAs** in the mobile agent. The **sa*i*** represents the SubAgent of the service provider **S*i*** in one visit then:

$$sa_1 \longrightarrow \mathbf{S1},\ sa_2 \longrightarrow \mathbf{S2}...\ sa_n \longrightarrow \mathbf{S}n\ ,$$

The SubAgents in the mobile agent are represented as n × 1 matrix in the mobile agent for n service provider:

$$SA = \begin{pmatrix} sa_i \\ sa_2 \\ . \\ . \\ . \\ sa_i \\ . \\ . \\ . \\ sa_n \end{pmatrix}$$

Before the SMAG system dispatches the mobile agent to the first station in its journey, all SubAgents must be encrypted by different secret keys of the service provider. The result of this procedure is ciphers data matrix C of the SA for n service

provides. So, if SA is converted to C, the mobile agent hides all its SubAgents. The following equation is true:

$$C_1 = sa_1 \times S_{Sk1}\ ,\ C_2 = sa_2 \times S_{Sk2}\ ,\ldots,\ C_n = sa_n \times S_{Skn} \quad \text{.....(1)}$$

$\times$ is encryption operation.

The VSM uses the secret key which is generated by service provider i **(S*i*)** to encrypt the SubAgent **i** (sa_i).

To generate C matrix which represents the encryption of n SubAgents, the VSM represents each secret key S_{ski} which is generated by service provider ***i*** in the matrix (n x n), n is the number of the service providers in a journey, such as:

$$s_{skii} = \begin{cases} s_{skpp} & if \quad i = p \\ o & if \quad i \neq p \end{cases}$$

Also the VSM represents each SubAgent as follows:

$$SA_i = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ sa_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \textbf{for i = 1 ,2, …, n}$$

The VSM generates C matrix by using equations (1) and the following equation:

$$C = \sum_{i=1}^{i=n} S_{ski} \times SA_i$$

To prove this equation:

The right side:

$$\sum_{i=1}^{i=n} S_{ski} \times SA_i \quad = (S_{sk1} \times SA_1) + (S_{si2} \times SA_2) + ... + (S_{ski} \times SA_i) + ... + (S_{skn} \times SA_n)$$

……(2)

$$(S_{sk\,1} \times SA_1) = (\sum_{a=1}^{a=n} \sum_{b=1}^{b=n} s_{skab} \times sa_b)$$

$$\begin{pmatrix} s_{sk\,11} \times sa_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} c_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

then :

$$(S_{ski} \times SA_i) = (\sum_{a=1}^{a=n} \sum_{b=1}^{b=n} s_{ski(ab)} \times sa_{i(b)})$$

$$= \begin{pmatrix} 0 \\ 0 \\ \vdots \\ s_{sk(ii)} \times sa_{(i)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ c_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{……(3)}$$

for i = 1,2,…,n

By using equation (2) and equation (3):

$$C = \begin{pmatrix} c_1 \\ 0 \\ . \\ . \\ . \\ 0 \\ . \\ . \\ . \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ c_2 \\ . \\ . \\ . \\ 0 \\ . \\ . \\ . \\ 0 \end{pmatrix} + \ldots + \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ . \\ . \\ . \\ c_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ . \\ . \\ . \\ 0 \\ . \\ . \\ . \\ c_n \end{pmatrix}$$

Then:

$$C = \begin{pmatrix} c_1 \\ c_2 \\ . \\ . \\ . \\ c_i \\ . \\ . \\ . \\ c_n \end{pmatrix} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. (4)$$

Each SubAgent carries a copy of the secret key that is generated by the mobile agent base to encrypt itself again after the mobile agent complete its work in that place. For security reasons, the VSM does not give the service provider the copy of that key and the encryption process is done under the mobile agent control. This procedure is executed immediately and automatically after the mobile agent completes its works in that place to prevent the service provider from altering the final results. The following secret keys are stored in SubAgents:

$sa1 \longrightarrow Hsk1$, $sa2 \longrightarrow Hsk2$,… , $san \longrightarrow Hskn$.

Before, the mobile agent starts its journey, all SubAgents take the form of the C matrix. This form protects the SubAgents from the malicious service providers. Also, this representation helps to generate several views from the C matrix. This is the main idea in VSM mechanism. When the mobile agent arrives to service provider Si, the C matrix will convert to the V*i* in order allow the service provider to manipulate with the SubAgent sa*i* only. The service provider uses the secret key **S*ski*** to decrypt ***Ci*** and it can not decrypt other SubAgents in the C matrix. It will be converted to **V*i*** in the service provider **S*i***:

$$v_i = \begin{pmatrix} c_1 \\ c_2 \\ . \\ . \\ . \\ sa_i \\ . \\ . \\ . \\ c_n \end{pmatrix} \qquad \text{.....(5)}$$

for $I = 1, 2, \ldots, n$

The following equations are true for decryption:

- The VSM uses the secret key **S*ski*** to decrypt **C*i*** in order to retract SA***i***

$S_{skj} \times c_i = sa_i$ if i= j for I = 1,2, … , n .....(6)

$S_{skj} \times c_i = c_i$ if i ≠ j for I = 1,2, … , n .....(7)

× decryption process.

To prove equation (5) which the VSM can generate each **v*i*** from convert C matrix by using equations (4), (6) and (7)

$$S_{pki} \times C = \begin{pmatrix} S_{pk_1} \times c_1 \\ S_{pk_2} \times c_2 \\ \cdot \\ \cdot \\ \cdot \\ S_{pki} \times c_i \\ \cdot \\ \cdot \\ \cdot \\ S_{pkn} \times c_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ sa_i \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{pmatrix} = V_i$$

For i = 1,2,..,n

The following equation represents the view in the service provider ***i*** :

$$v_i = \begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ sa_i \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{pmatrix}$$

So, each service provider has the views of the mobile agent such as:

S1 $\longrightarrow$ ***V1*** , ***S2*** $\longrightarrow$ ***V2*** , ..., ***Sn*** $\longrightarrow$ ***Vn*** ,

There is a new question here: What is the form of the C matrix after the mobile agent completes its duties and leaves its current location, for example, the service provider S***1***? The answer is that all SubAgents are still encrypted except ***sa1*** which is decrypted in S1 by using S***sk1***. So, if the mobile agent continues its journey with this picture, the SubAgent ***sa1*** may face attack by the next malicious service providers. For security reasons, the security model protects the SubAgent again by using the symmetric encryption mechanism to encrypt ***sa1*** by secret key of the mobile agent base ***Hsk1*** which described above. So a mobile must encrypt the SubAgent before it leaves its location to another station. By using the secret keys that are stored in the

SubAgents, the VSM encrypts again the SubAgent. The following equation encrypts the SubAgent:

$$c_i^* = sa_i \times H_{ski} \quad \textbf{for i = 1,2,..,n}$$

C_i^* Is encryption data of SubAgent **sa*i*** by using H_{bki}.

The C matrix of the mobile agent takes the following form after the mobile agent visits k service providers:

$$C = \begin{pmatrix} c_1^* \\ c_2^* \\ . \\ . \\ . \\ c_k^* \\ c_{k+1} \\ . \\ . \\ c_n \end{pmatrix}$$

When the mobile agent returns home, it needs to decrypt all SubAgents to display all information. To perform this, it uses the secret keys **Hsk**. The following equations are used to perform this procedure:

To decrypt each SubAgent sa*i* :

$$sa_i = c_i^* \times H_{skj} \qquad if\,(i = j) \text{ .....(8)}$$

$$c_i^* = c_i^* \times H_{skj} \qquad if\,(i \neq j) \text{ .....(9)}$$

× decryption process. for I =1,2 .., n.

In order to decrypt all SubAgents:

$$SA = C \prod_{i=1}^{i=n} H_{pki} \qquad \text{for I =1,2 .., n. ……...(10)}$$

To prove equation (10):

the right side of the equation:

$$C\prod_{i=1}^{i=n} H_{pki} = \begin{pmatrix} c_1^* \\ c_2^* \\ \vdots \\ c^* \\ \vdots \\ c_n^* \end{pmatrix} (H_{sk1} \times H_{sk2} \times \cdots \times H_{skn})$$

$$= \begin{pmatrix} c_1^* \times H_{sk1} \\ c_2^* \times H_{sk1} \\ \vdots \\ c_i^* \times H_{sk1} \\ \vdots \\ c_n^* \times H_{sk1} \end{pmatrix} (H_{sk2} \times \cdots \times H_{skn})$$

$$= \begin{pmatrix} sa_1 \\ sa_2 \\ \vdots \\ sa_i \\ c_{i+1}^* \\ \vdots \\ c_n^* \end{pmatrix} (H_{sk(i+1)} \times \cdots \times H_{skn}) = \begin{pmatrix} sa_1 \\ sa_2 \\ \vdots \\ sa_i \\ c_{i+1}^* \\ \vdots \\ c_n^* \end{pmatrix} = SA$$

The following problem may occur:

When the mobile agent arrives to the service provider Si, for example, the service providers may attack the mobile agent by altering the value of the secret key which is stored in the SubAgent **sa*i***. To avoid this problem, the VSM computes data-digest by using the hash function described in this chapter and it saves this data in SubAgent. The mobile agent also carries the secret key. So, it generates again the data-digest

before it encrypts the SubAgent **sa*i***. In addition, it makes a comparison between two data-digests. If the result is true that means the secret key is not altered and it can complete the encryption process. Otherwise, it doesn't use the secret key because the mobile agent base can not be able to decrypt the SubAgent **sa*i*** and it can notify the mobile agent owner by that. Figures 4-5 and 4-6 show the security model items in the SMAG system and service providers.

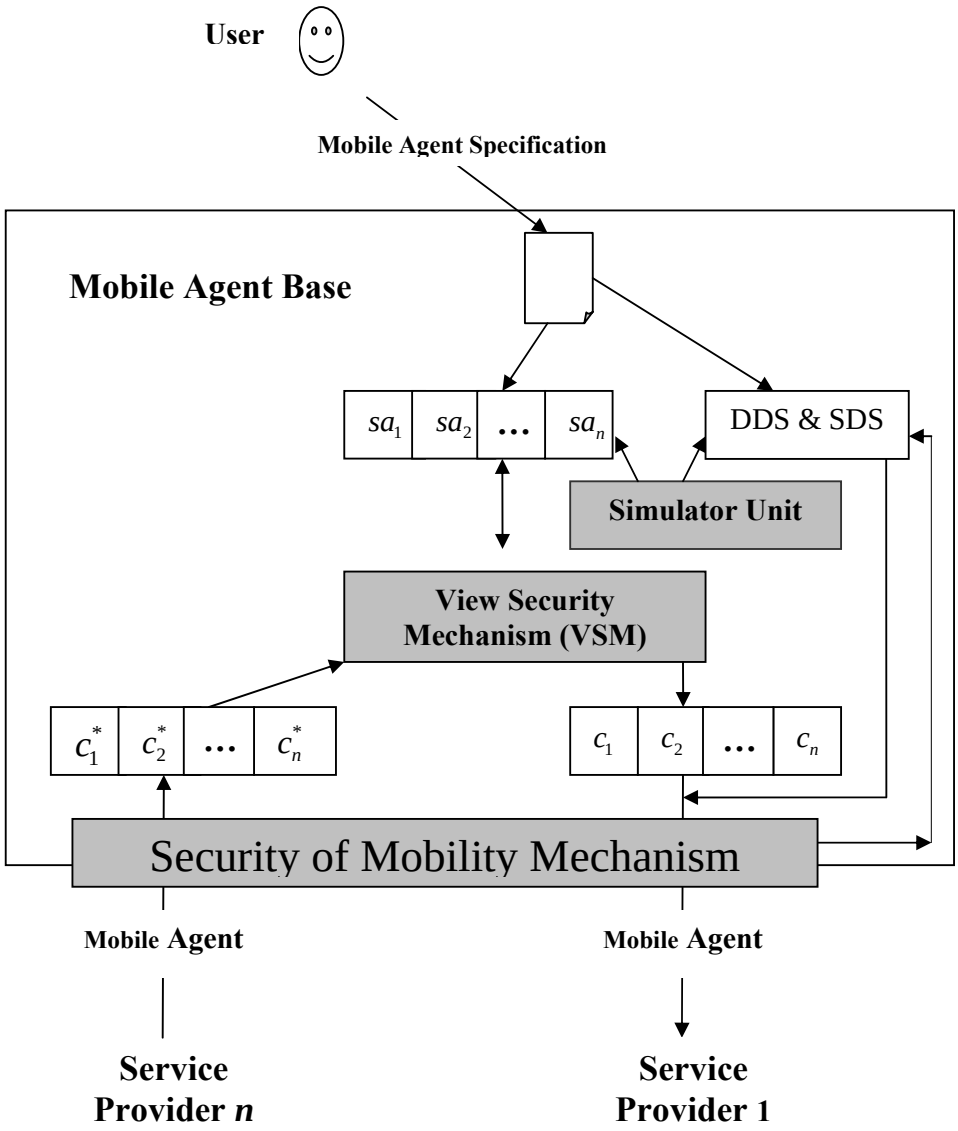


Figure 4-5
Security Model Architecture in Mobile Agent Base

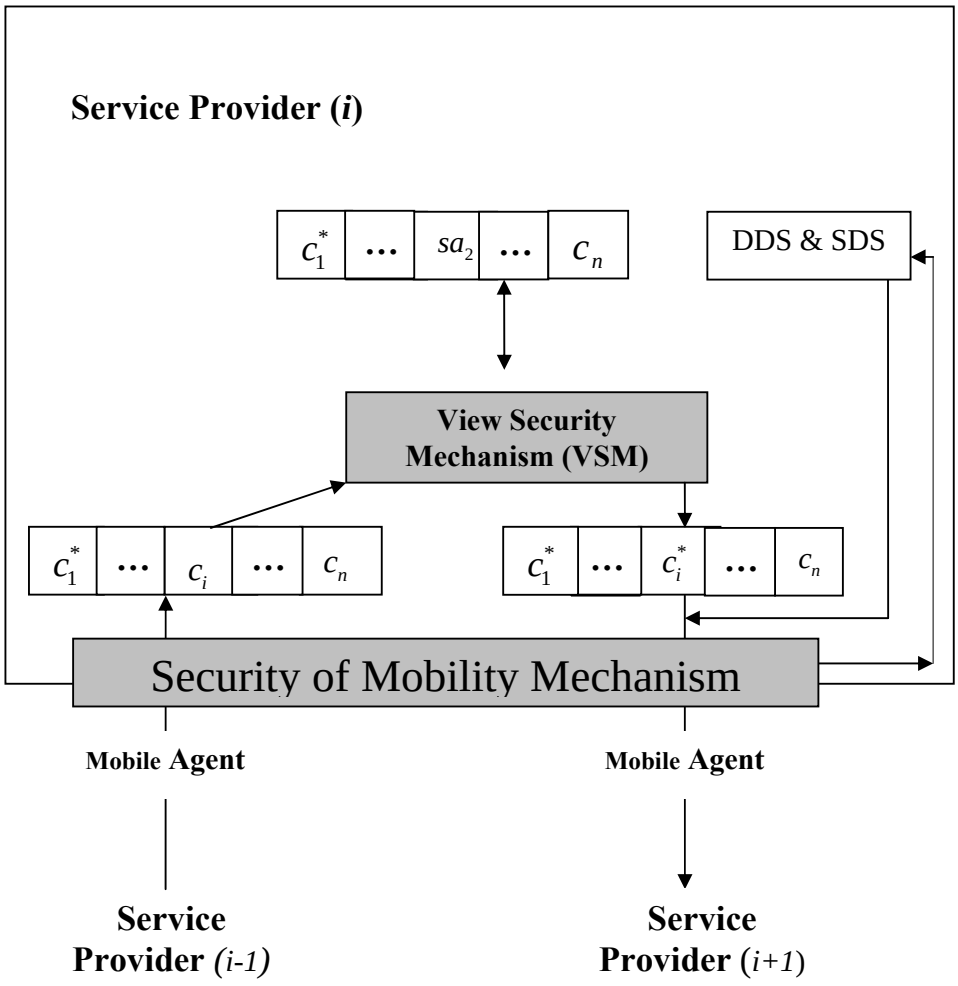


Figure 4-6
Security Model Architecture in the Service Provider (*i*)

Figure 4-7 shows the architecture of SubAgent **sai**:

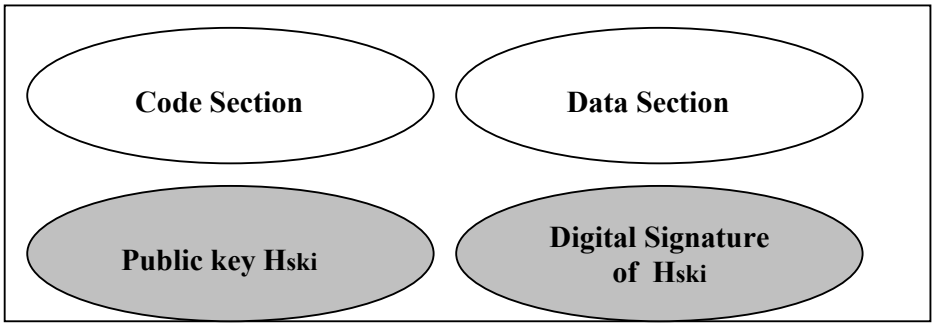


Figure 4-7 Secret Key in a SubAgent

4.3.3 Protection of Mobile Agent Data

The mobile agent has others data items such as general data, identification information and an itinerary table. These items need the protection because they are not encrypted during the mobile agent visits to the service providers. The mobile agent behaviour depends on these values and any modification reflects the mobile agent objectives. The security model protects these items by classifying them into two categories. The first is Dynamic Data (DD) that is changed during the mobile agent journey. This change occurs by the execution of different tasks in different places. The updating of DD is done under the mobile agent control. The second class is Constant Data (CD). It is constant during the mobile agent journey. The mobile agent prevents any action to alter CD. The security model provides two mechanisms for each class as follows:

4.3.3.1 Simulator Mechanism

The DD is a part of the mobile agent. It uses to store variable data. Before the mobile agent starts its journey, the mobile agent base gives these variables initial values. The modifications of these values must be changed according to the executions of some processes in the mobile agent. The results of these executions reflect first in SubAgents before they reflect on the DD. So, when the mobile agent returns home, the DD with new values and a serial of results are stored in the SubAgents individually. A new mechanism is designed in this thesis called Simulator Mechanism (SM). The SM can simulate the mobile agent works in changing the DD. By taking the results in each SubAgent and the initial values of the DD as inputs, it performs a set of actions on them. The SM finds these actions in SubAgents. It makes a comparison between its result and the information coming with the mobile agent. The result will give true or false. If true it means the DD is correct. If it is false, it means

the values of the DD are attacked. The SM can make debugging by using a sequence of actions step by step in each SubAgent in order to know where the defect is. This approach is very useful for detecting the violation on DDS.

4.3.3.2 Constant Data Protection Mechanism

In this point, some ideas proposed by existing mechanisms are used, for example, Read-only and Append-only mechanism [47]. The security model uses Constant Data Mechanism (CDM) to protect CD. This mechanism is designed in this thesis. Each mobile agent has constant information, for example, identification information, authority information and an itinerary table. A Data-Digest is used for authentication. By using the hash function, the mobile agent base computes the Data-Digests for each CD member before dispatching the mobile agent. It saves these data in the mobile agent base. When the mobile agent returns home, the mobile agent base computes again the date-digests. If any differences exist in the data or not, then the mobile agents owner's knows that. Figure 4 shows the architecture of the mobile agent which will visit n service providers. The coordinator unit makes coordination between different items in the mobile agent. It provides the mobile agent with some facilities for example, the encryption mechanism, the updating DD and so on.

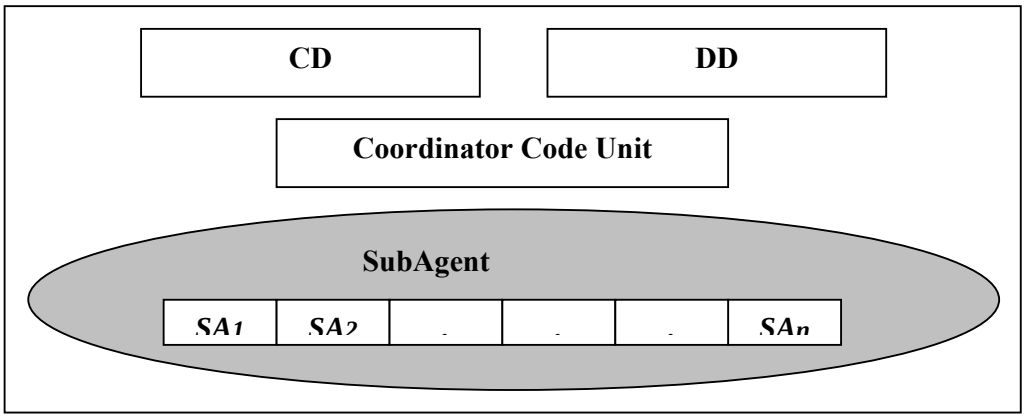


Figure 4-8 Items of the mobile agent with n SubAgents

4.3.4 Mobility Protection

A mobile agent can move from a place to another under its control. Before it leaves, it is converted into binary data. A communication between entities is important in the SMAG system. The communication message is represented as object. Also, it is converted to the binary data before it is sent to its target place. So, the mobile agents and the messages data may face different types of attacks which are mentioned in chapter 1. The SMAG system needs a mechanism which has ability to protect these data during their mobility in the network media.

There are two points needed to be solved in this area of security:

1. Security of transferring data (the mobile agent and communication message).
2. Security of the mobile agent against service provider. When the service provider destroys or alters the mobile agent.

There are two mechanisms to solve the above problems:

4.3.4.1 Encryption Mechanism for Secure Data Transfer

This part of the security some ideas that are introduced by some mechanisms, for example, Code Signing [30, 32]. This mechanism is designed and implemented in this thesis. It uses the symmetric encryption mechanism described in this chapter [70]. It assumes there are sender (the mobile agent base or the service provider) and receiver (the mobile agent base or the service provider). The sender and the receiver must have a secret key. The secret key must be kept securely and a copy of it is sent to the receiver. The following steps are done to the mobile agent and the message data:

1. The sender generates a paired of keys (private key and public key).

2. The sender uses the hash function to generate the data-digest from the data that wishes to send it securely by using the private key.
3. The sender uses the secret key to encrypt the data-digest (digital signature), original data and the public key.
4. The sender sends the encrypted data to the receiver.
5. By using the secret key, the receiver decrypts the encrypted data to get the data-digest (digital signature), the original data and the public key.
6. By using the hash function and public key, the receiver computes again the data-digest of the decrypted data.
7. By making a comparison between two data-digests, the receiver knows if they match or not.

By using the above steps, the receiver receives correct and secure data from the sender.

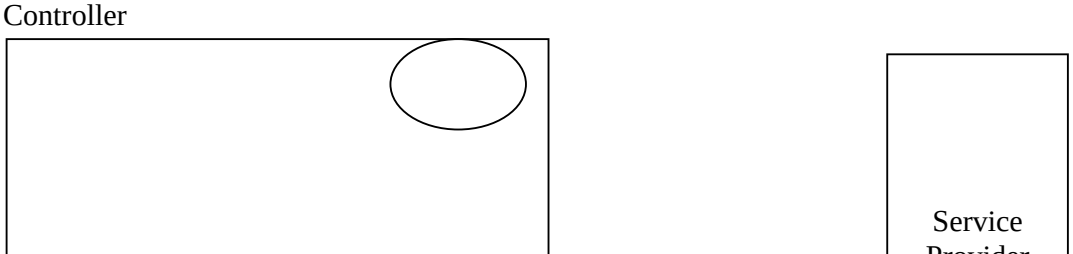
4.3.4.2 Controller and Safe-Data-Digests Mechanism

A Controller and Safe Data-Digests Mechanism (CSDM) are designed and implemented in this thesis. It solves a big problem of the mobility security that most current systems have failed to solve. The problem appears when the service provider attacks the visiting mobile agents by destroying or altering them. The use of this mechanism is optional to the user when he/she uses the mobility statement. If the user writes "move to service_provider_name through controller" that means the mobile agent travels to the controller of the service provider before the service provider itself. The controllers are trusted hosts that the SMAG system is used to protect the mobile agents. When the mobile agent arrives to the controller, the controller achieves as following steps:

- It saves a copy of the mobile agent in its storage media.

- It receives a Safe-Data-Digests (SDD) from the visiting mobile agent. The SDD is generated data from the mobile agent. The controller uses the Hash Function to generate these data from SubAgents and read only data. So, the controller generates a data-digest for each SubAgent by the Hash Function except the SubAgent that will perform in the service provider.

After that, the controller dispatches the mobile agent to service provider and waits for specific time. In case the service provider prevents the mobile agent from returning to controller, it uses the copy version of the mobile agent and the mobile agent continues its journey. Also, it registers the name of mobile agent in a list to destroy it in case the old version comes after the specific time. It can notify the mobile agent owner by that. On the other hand, the mobile agent returns from the service provider to the controller, it generates again the Safe-Data-Digests and the controller verifies the two Safe-Data-Digests. In case there is no change between two data then the mobile agent is not attacked by the service provider. So the mobile agent continues its journey normally. But, if there is a change between two data then there is attack by the service provider. So, the controller uses the original version copy of the mobile agent and it can notify the mobile agent owner by this situation. The mobile agent can continue its journey later on. Figure 4-9 shows this mechanism



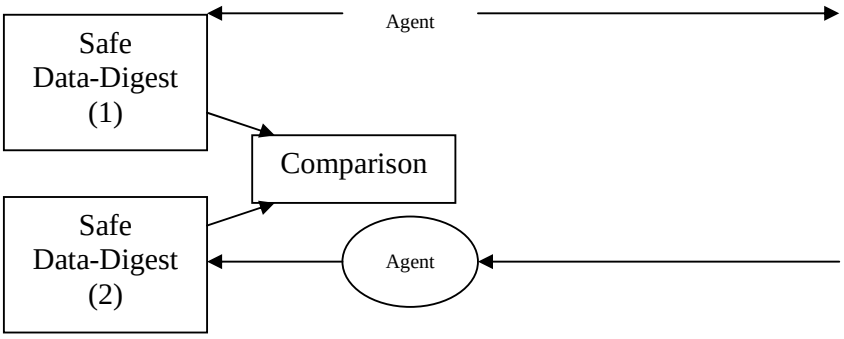


Figure 4-9 CSDM mechanism

Chapter 5

SMAG System Implementation

5.1 Foundation and Background

This chapter presents some foundations for the SMAG system that is designed and implemented in this thesis. The C# (C sharp) is used as a base programming language

and the .NET framework for implementing the SMAG system. Some concepts of the mobile agent system are also reviewed by using programming concepts and some C# mechanisms. Moreover, this chapter describes briefly, the cryptography mechanisms which are used in the security area of the SMAG system such as, the symmetric encryption mechanism, the public-key encryption mechanism, the digital signature mechanism and digital certificate.

5.2 C# as a Choice of Programming Language

A mobile agent system needs a rich language that provides it by many features. The language must have the ability to implement different properties of the mobile agent system, for example, the mobile agent is represented as an object, mobility, communications, and so on. It must also be easy to control and doesn't suffer from performance and security. Microsoft Company has developed C# as an object oriented programming language in visual studio dot net [71]. This language uses the .NET framework as a platform. Moreover, it is an object oriented language and it is a web-enabled. Also, it embodies today's concern for productivity, security, and robustness. Moreover, it can use a large number of classes which are available in the .NET framework and these let the language work in a large domain and in different application types. By using C#, different application types can be developed, such as [72]:

- Console Applications: A project of this type can execute entirely in a DOS command windows.
- Windows Applications: Through this type, developers can develop applications like a traditional windows application. It can be used in stand-alone and distributed application.

- ASP.NET Projects: Active Server Page (ASP) is a Microsoft technology for creating browser based applications. By using the browsers, this type of projects can be executed. Besides ASP code, HTML and other languages, the C# code can be used to develop the application.
- Web Services: Web Services projects provide information to other application and components. It uses Simple Object Access Protocol (SOAP) in its work.
- .NET Components: a component library of the classes that can be used by other programmes. Through this type of project, the developers can add new components.

There are also additional types of project. The SMAG system uses some of these types of applications. For example, for implementing the mobile agent base and service providers, it uses console application type, the ASP type for implementing the client node application and it uses the .NET framework Component to implement the mobile agent class, the encryption class, etc.

5.3. .NET framework

The .Net Framework is a platform that makes it easier for programmers to use different languages (C#, C++, Jscript.NET and Visual Basic.NET). It represents a unified, object-orientated set of the services and libraries available to different applications from different languages in the visual studio .NET. By using Common Language Specification (CLS) that is provided by Microsoft for .NET, the vendors can develop compilers that produce a standardized binary code format. The .NET Framework offers a number of benefits to a developer, such as a consistent programme model, direct support of security, simplified development efforts and easy application deployment and maintenance.

. NET framework consists of three key elements as follows:

- Common Language Runtime (CLR): This component is a layer between the application and the operating system, Figure 5-1 list the components of the .NET framework. It provides a variety of execution services that include thread management, component life time management, default error handling and memory management. It also compiles the code before its executions.
- Common Type System (CTS): This component provides a component set of data types. By using the advantage of CTS, the .NET programming language enables developers to use their languages built–in data types. Figure 5-1 shows this component.
- .NET Class Library: This component contains hundreds of the classes which are divided into namespaces. The root namespace of the .NET Class Library is called ***System***. It contains core classes and data types.

5.4 Mobile Agent as Object

The SMAG system represents a mobile agent as object. The mobile agent class describes this object. By using DotNet component type project, the class of the mobile agent is developed and compiled to get a version of Dynamic Link Library DLL for the mobile agent that can be used by other parts of the system. Any part of the system must use a copy of this version in order to be able to manipulate with the mobile agents. To avoid any security problems, the compiler gives to any DLL version of this class a unique number as stamp. The mobile agent class consists of the data and method members. Before the mobile agent starts its journey, the system creates an object of this class and prepares it with all information according to its owner's requirements. This object can move among hosts. .NET framework has two important classes: The Serialize and the Deserialize class.

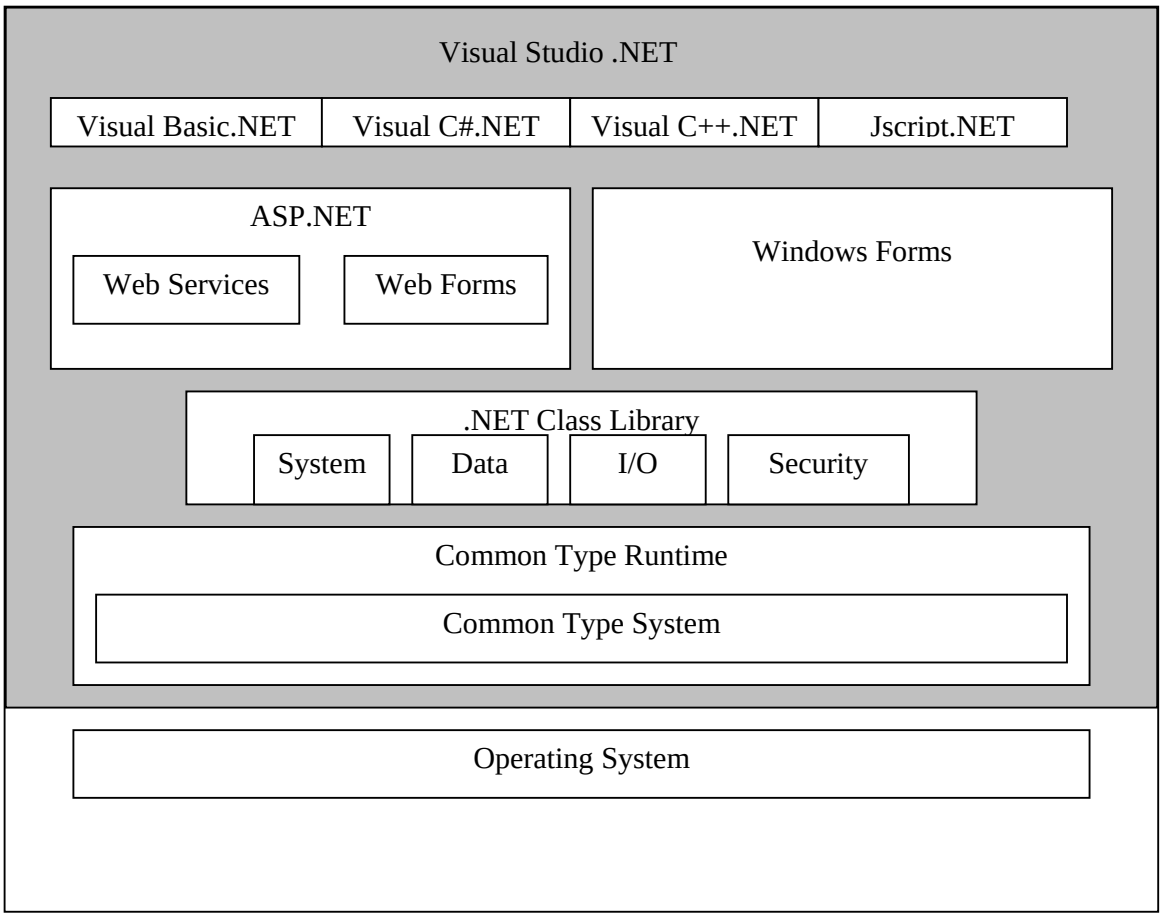


Figure 5- 1 Components of the .NET framework.

These classes are used to covert the object of mobile agent to binary data format. So, the binary data of the object can be transmitted over the network transmission media. The Serialize class converts the mobile agent to the binary data and the Deserialize class converts the binary data to the object again. Figure 5-2 shows this operation.

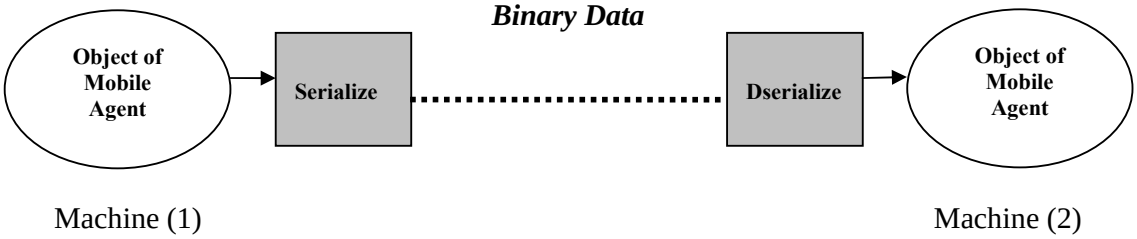


Figure 5-2 Mobility Mechanism

5.5. Cryptography Mechanism and .NET framework

The .NET framework has several classes that provide different mechanisms of cryptography through them [70]. The Base Class Library's System.Security.Cryptography namespace contains the common symmetric encryption (i.e. DES, 3DES, RC2, Rijndal/AES), asymmetric encryption (i.e. RAS, DSA) and hash algorithm (i.e. MD5, SHA1, SHA256, ASH384, ASH512). Also, the digital certificates are available in System.Security.Cryptography.X509 namespace.

Figure 5-3 shows the three levels inheritance pattern of the symmetric encryption. The root level abstract class is:

SymmetricAlgorithm (System.Security.Cryptography.SymmetricAlgorithm)

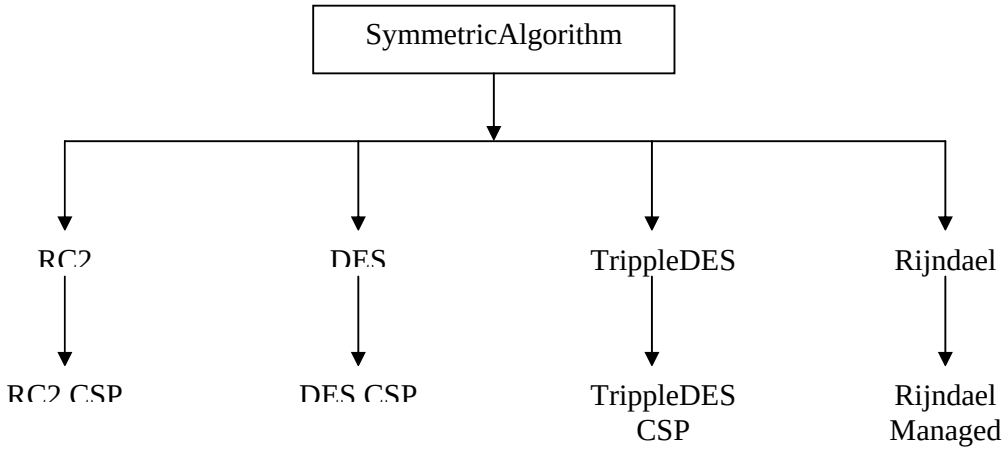


Figure 5-3 Symmetric Algorithms

Figure 5-4 shows also the three levels inheritance pattern of the asymmetric encryption. The root level abstract class is AsymmetricAlogorithm.

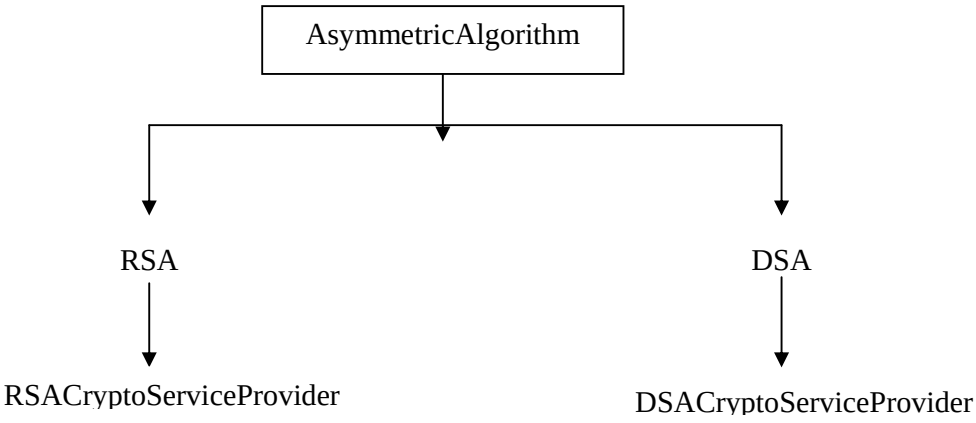


Figure 5-4 Asymmetric Algorithms

Figure 5-5 shows hash algorithms. The root class is HashAlgorithm is abstract and it has an overloaded method to compute message-digest.

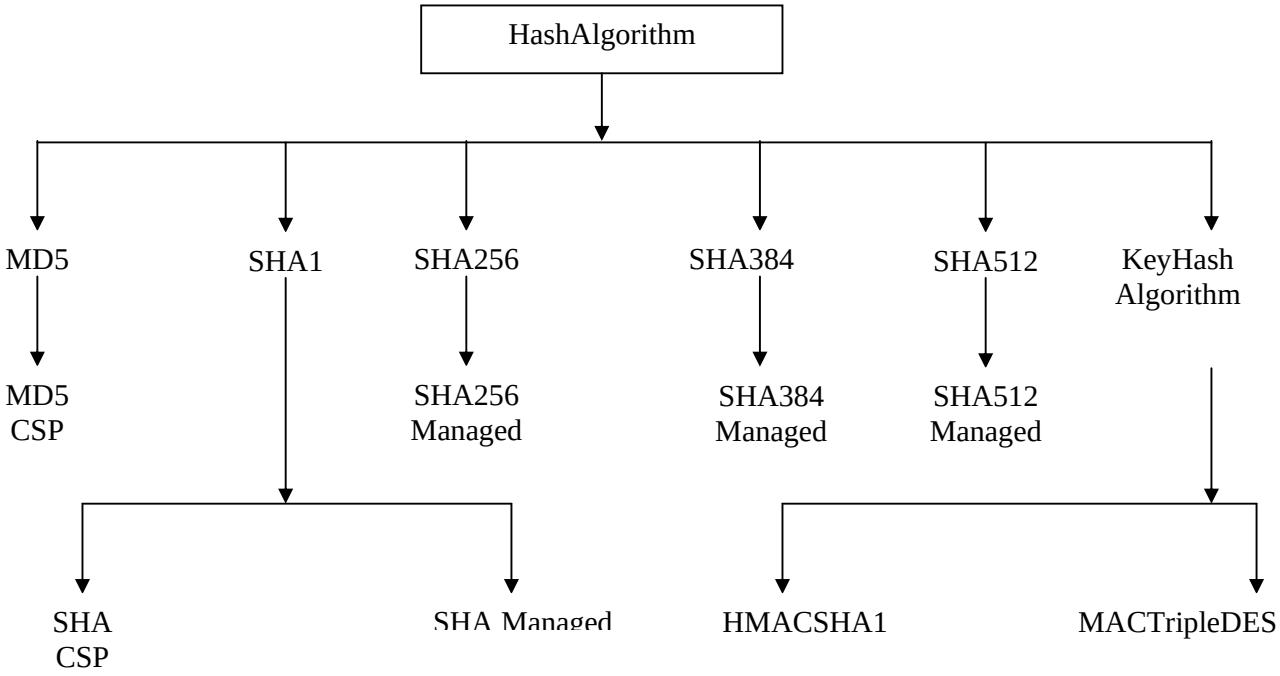


Figure 5-5 Hash Algorithms

5.6 System Classes Description

This chapter describes the SMAG system implementation. The implementation covers what are contributed in this thesis. By using C# and .NET framework, the SMAG system has been implemented. The users can use the system through friendly web forms. So, it is very simple to use. It needs only to be familiar with MASL syntax and services description that are provided by the service providers. The system consists of several classes. Each one has specific role as following:

- Code Statement Class.
- Encrypted Statement Class.
- Mobile Agent Class.
- Agent Virtual Machine Class.
- Agent Container Class.
- Agent Data Transfer Class.
- Digital Certificate Class.
- System Administrator Class.
- Encryption Class.
- Run Services Class.
- Resource Connection Class.
- Mobile Agent Base Class.
- Mobile Agent Base Interface Class.
- Service Provider Class.
- Controller Class.

The following paragraphs describe the content of each one.

5.6.1 Code Statement Class (CSC)

This class is a statement code structure. The users describe their tasks through MASL. Each task consists of at least one statement. By using objects of CSC, the system generates mobile agent behaviour. The mobile agent can start its journey after the system represents all tasks of the user in this structure. The CSC contains the following members:

5.6.1.1 Statement Code Member (SCM)

This member represents the type of the statement, for example, if SCM equals "1" that means a mobility statement, if equals "3" it means an assignment statement and so on.

5.6.1.2 Server Name

This member specifies the place that the statement will be executed on. The place can be the mobile agent base or the service provider.

5.6.1.3 Parameters

Parameters are used to store statement data inputs and statement execution results. For example, if the statement is a mobility statement then this statement must store the destination name, an IP address and a port number in the parameters.

5.6.1.4 Statement Summary

After the system completes a statement execution, the mobile agent makes a summary of this execution and it stores this information in this member.

5.6.1.5 Secret Key

Before the mobile agent starts its journey, the mobile agent base provides each statement with a secret key that is used to encrypt the statement again after the execution of the statement is done.

5.6.2 Encrypted Statement Class (ESC)

Before the mobile agent starts its journey, the system converts all statements from CSC to ESC structure. This class contains an encrypted data of the CSC object and additional information. By using secret keys which are generated by the locations that statements will be executed on them, the mobile agent base encrypts the CSC objects of the statements. This class consists of the following members:

5.6.2.1 Encrypted Statement

In this member, the system stores the encryption data of the CSC object. By using Encryption class, the system generates this data.

5.6.2.2 Statement Serial Number

This member is an integer number which is used to numbering the statements of the mobile agent.

5.6.2.3 Complete Statement Flag

This member is a Boolean variable. The true value means this statement was executed and the false value means the statement is not executed yet.

5.6.2.4 Server Name

This member specifies a place that the statement will be executed on. The place can be the mobile agent base or the service provider. The system stores this item redundancy for security reason.

5.6.3 Mobile Agent Class

This class describes a mobile agent structure that can move among hosts. It represents user tasks in each host. Also, it contains identification information, an itinerary table, execution data and some assistant methods for the execution. They are described as follows:

5.6.3.1 Identification Information

This member consists of some information about the mobile agent identification. For example, a mobile agent owner name, a mobile agent name and a mobile agent base name.

5.6.3.2 Journey Information

A Journey Information represents information about an itinerary table and a current host name. The itinerary table is distributed among objects of the CSC and ESC. The current host name is changed during the journey.

5.6.3.3 General Variable Arrays

These arrays are used to represent names, types and values of transactions in each place. The variable name is created before the mobile agent starts its journey. The values of these variables are changed according to statements execution.

5.6.3.4 Encrypted Statement array

This member represents user tasks in each place. Each element of this array is an ESC object.

5.6.3.5 Set and Get Current status

These methods are used to change and to get a current status of the mobile agent.

5.6.3.6 Assistant methods

These methods are used to provide the mobile agent with some functions like, getting and changing the current host name, getting and putting ESC in the Encrypted statement array, getting identification information and so on.

5.6.4 Agent Virtual Machine Class

This class represents the execution engine. It executes a statement that is represented in CSC object. The execution is done according to statement code type. This class transfers an execution to the Run Services Class if the type of the statement is a

mobility statement or a call service statement. After the statement execution is completed, this method encrypts again the statement by using encryption class. Also, it stores the encryption data in the mobile agent by using assistant methods. This class is hosted by the mobile agent base and the service providers.

5.6.5 Agent Container Class (ACC)

By using, this class, the mobile agent is protected against attacks during its movement among different places. It is used to carry three items as follows:

5.6.5.1 Mobile Agent Object

The ACC object carries an object of the mobile agent among different parts of the system (the mobile agent base or the service Providers). This item represents the mobile agent object.

5.6.5.2 Signature Data

An ACC object receiver uses this data to authenticate that the mobile agent has arrived without any attack.

5.6.5.3 Public Key

A public key is used to verify the Mobile Agent Object and the Signature Data.

5.6.6 Agent Data Transfer Class (ADTC)

This class is used to carry an ACC object after the system encrypts it. Also, it contains some members as follows:

5.6.6.1 Data

This member represents an encryption data of the ACC object. The constructor of this class uses the encryption class and Serialize class to generate this data.

5.6.6.2 Server Name

This item represents the destination of the Data member. So, the receiver will know from where the mobile agent information comes.

5.6.7 Digital Certificate Class

This class is used to publish the public keys of different parts in the system through the system administrator. The public key is used to exchange secret keys. It contains the following members:

5.6.7.1 Server Name

This member represents an owner of the public key.

5.6.7.2 Public Key

This member contains the public key data that is generated by the mobile agent base or the service provider. Also, it is represented in RSAParameters object.

5.6.8 System Administrator Class

This class publishes digital certificates among different parts of the system. Also, it maintains the system name services. Moreover, it keeps tracks of the system transactions. It consists of the following members:

5.6.8.1 Main Method

The main method of this class has a listener which accepts the digital certificate from different destinations. It stores these certificates in array by using Serve Certificate Method. Also, it publishes these certificates by using Send Digital Certificate Method.

5.6.13 Encryption Class

This class consists of seven methods which are used in the encryption and the decryption tasks. Also, it uses the symmetric encryption mechanism and the asymmetric encryption mechanism. The descriptions of these methods are as follows:

5.6.13.1 Send Digital Certificate Method

This method is used to publish digital certificates among different parts of the system. Also, it receives the digital certificate, the IP address and the port of the digital certificate owner.

5.6.13.2 Encrypt Agent Method

This method uses the symmetric encryption mechanism. It receives the agent container object. By using RijndaelManaged class and the secret key of the sender, it encrypts the agent container object. An output of this method is cipher data which is stored in ADTC object. Also, Encrypted Agent method generates the private and the public key by using RSACryptoServiceProvider class. The private key is used to generate the signature data from a mobile agent object which is stored in the agent container. Also, the public key is stored in the agent container before it is encrypted.

5.6.13.3 Decrypt Agent Method

This method uses the symmetric encryption mechanism. It receives an encrypted data of the agent container object. By using RijndaelManaged class and the secret key of the sender, it decrypts the agent container. An output of this method is the agent container object which contains a mobile agent object. Before it gets the agent container, this method verifies the signature data to be sure that the mobile agent arrives without any attack by using the public key. Also, it uses RSACryptoServiceProvider class to achieve that.

5.6.13.4 Encrypt Statement Method

This method uses the symmetric encryption mechanism. It receives the code statement object and the secret key which is generated by the service provider. By using RijndaelManaged class and the secret key, it encrypts the code statement object. An output of this method is the encrypted statement object which is stored in the mobile agent object.

5.6.13.5 Decrypt Statement Method

This method uses the symmetric encryption mechanism. It receives the encrypted statement object and a secret key. By using RijndaelManaged class and the secret key, it decrypts the code statement. An output of this method is the code statement object which is executed by the Agent Virtual Machine.

5.6.13.6 PKI Encrypt Method

This method is used to exchange the secret keys for symmetric encryption tasks. It uses the asymmetric encryption mechanism. By using RSACryptoServiceProvider class and the sender private key, it encrypts the secret key. This method receives array of bytes which are represented the secret key. The output is a cipher text of the secret key.

5.6.13.7 PKI Decrypt Method

This method is used to exchange secret keys for the symmetric decryption tasks. It uses asymmetric encryption mechanism. By using RSACryptoServiceProvider class, it decrypts a cipher data of the secret key by using a sender public key. This method receives array of bytes which are represented the cipher data. The output is the secret key of the sender.

5.6.14 Run Services Class

This class is used by the mobile agent base and the service providers. It provides the system with some services. These services are represented in three constructor methods. Each one has specific role as follows:

5.6.14.1 Digital Certificate Service Method

This method receives the digital certificate. Through the listener, it receives socket information as array of bytes. By using the Deserializing class, it reconstructs the digital certificate. So, the output of this method is the digital certificate.

5.6.14.2 Mobile Agent Service Method

This method receives the agent data transfer object. Through the listener, it receives socket information as array of bytes. By using the Deserializing class, it reconstructs the agent data transfer object. This object contains a cipher data of the gent container. By using Decrypt Agent Method, the service method generates the agent container which carries the mobile agent object. So, the output of this method is the mobile agent.

5.6.14.3 Resource Service Method

This method works as interface to service provider resources and the mobility service. In case the code statement type is the Call Service then this method will execute it by using Service Connection Class. Also, this method executes the code statement if the mobile agent requests to leave its current location. After the code statement is executed, this method uses the encrypt method to encrypt the code statement object again. In case the code statement is the mobility statement type, this method creates the new agent data transfer that carries the mobile agent object. After that, it serializes the object of the agent data transfer and sends it to the target place by using the IP address and the port number that are stored in code statement object.

5.6.14.4 Resource Connection Method

This class represents several methods. It locates in the service provider or the mobile agent base. The mobile agents can only access resources of the places through this method. So, each service provider represents the call of its services in this method.

5.6.15 Mobile Agent Base Class

This class represents the mobile agent base. The mobile agent object starts and finishes its journey in this class. Also, this class uses some classes which are described above. It consists of the following members:

5.6.15.1 Socket Data

The class has a listener that accepts information. This information is stored in the Socket Data member.

5.1.11.2 Serve Agent Method

This method receives the socket data and it delivers it to the mobile agent service method in Run Service Class in order to get the mobile agent object. Also, it gets code statement objects that will be executed in the current location. This method decrypts the code statements by using decrypt method. In addition, it delivers the code statement after it has been encrypted to the Agent Virtual Machine. In case the code statement is the Call Service or the mobility statement, it calls the resource service method of the Run Service Class. When the mobile agent returns from its journey, this method stores all the results carried by the mobile agent in the mobile agent base database.

5.6.11.3 Main Method

This method controls all actions of this class. It defines a listener according to specific IP address and Port number. Through infinite loop, it accepts information from different parts of the system. When it receives information, it creates a new object of the mobile agent class. By using Thread class, it creates a new thread object for processing this information. Each thread calls Serve Agent method.

5.6.16 Mobile Agent Base Interface Forms

Users use these forms to interact with system. They are created in an ASP.NET project, as following:

5.6.12.1 System Login Web Form

Through this form, the user can access the system. He/she must have a user account. The system authenticates this information. In case the user successes to access the system, he / she goes to a System Operation Web Form. Otherwise, the system prevents him/her from accessing. Also, through this web form, the user can move to the User Registration Web Form to get new account information. Figure 5-6 shows this form.

5.6.12.2 User Registration Web Form

The user uses this form to get account information in order to access the system. This form receives some information about the user and it stores it in the system database. Also, the user can specify a login name and a password. Figure 5-7 shows this form.

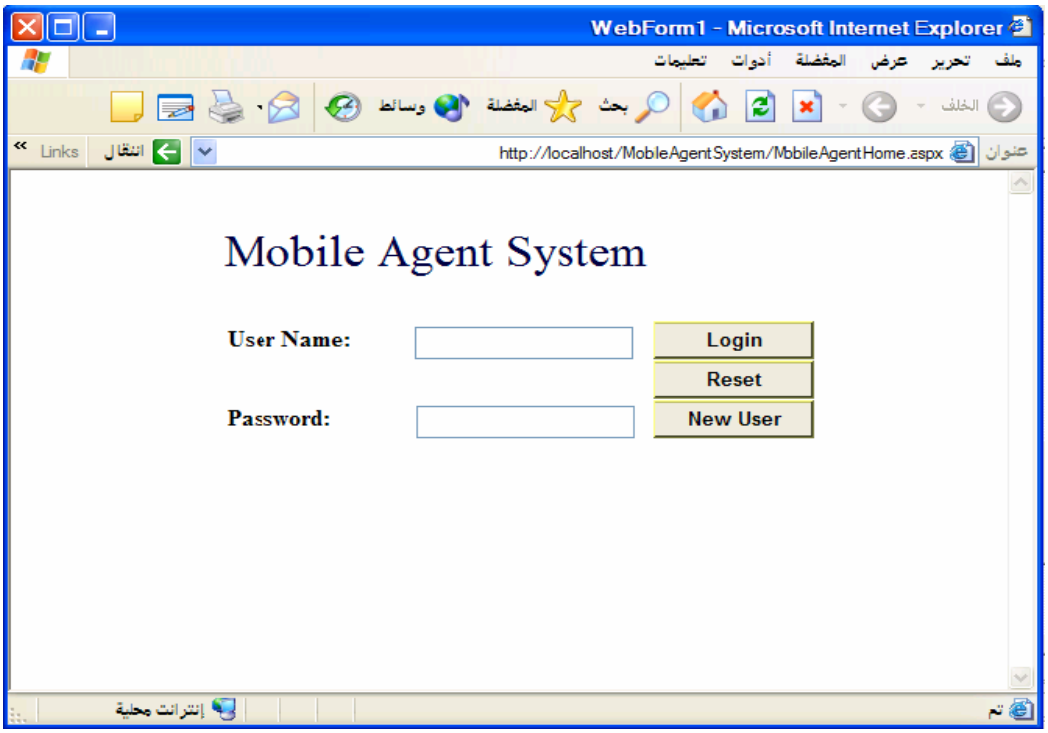


Figure 5-6 System Login Web Form

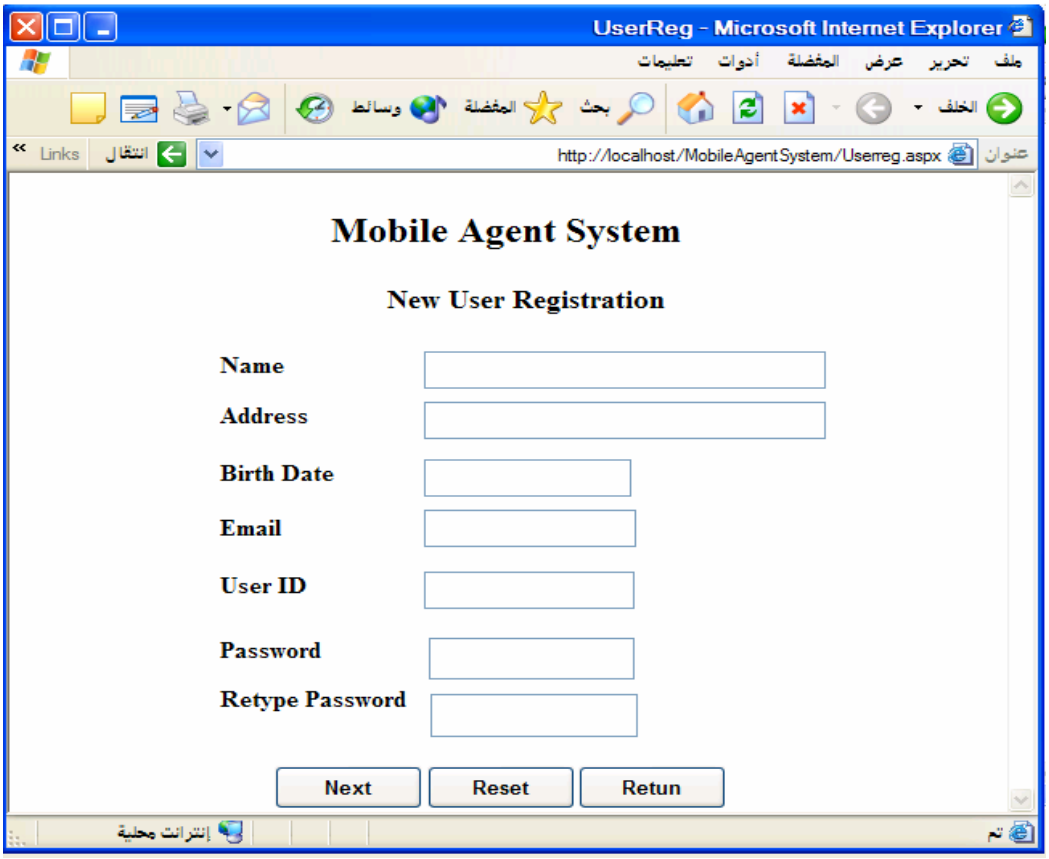


Figure 5-7 User Registration Web Form

5.6.12.3 System Operations Web Form

In case the user succeeds to access the system, this web form is the next step. It is a control panel of operations. Also, it allows to use different operations provided as web forms as following:

- Mobile Agent Builder Web Form.
- Mobile Agent Displayer Web Form.
- Report Web Form.

Figure 5-8 shows this web form.

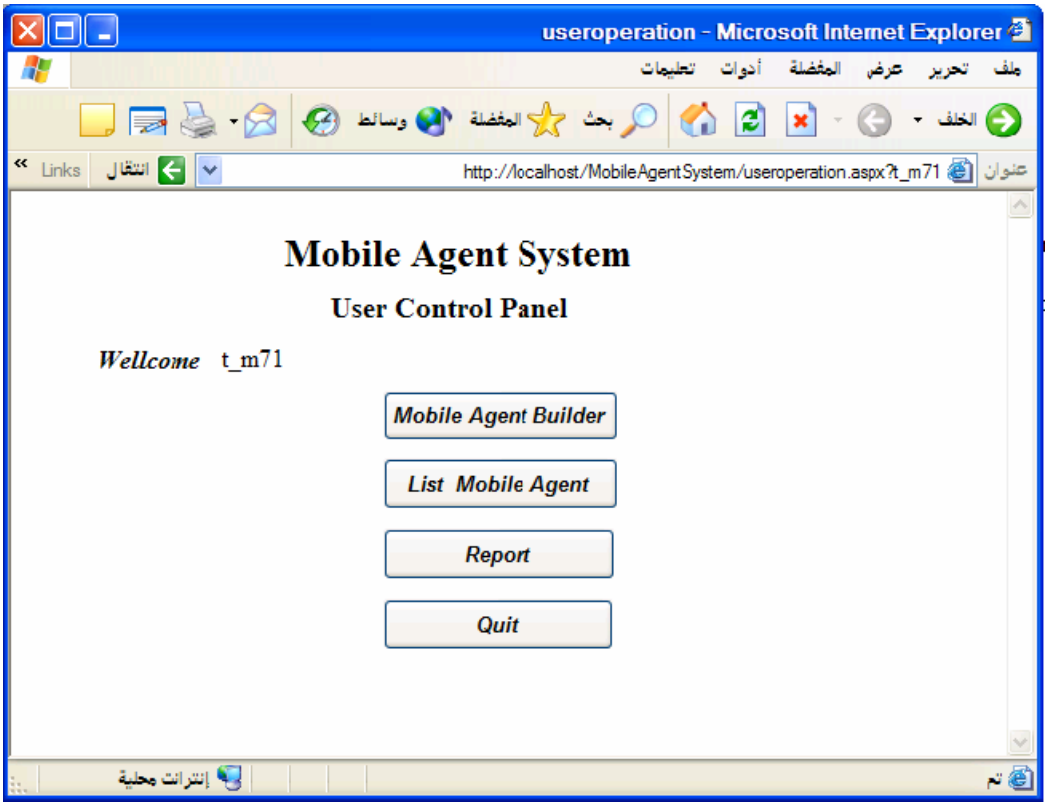


Figure 5-8 System Operations Web Form

5.6.12.4 Mobile Agent Builder Web Form

By using this form, the user can write a specification of the mobile agent, generating mobile agent behaviour and dispatching the mobile agent object. The specification must be through MASL syntax. The generator checks the syntax of the specification. Also, it generates all code statement objects that are equivalent to the specification. The dispatcher creates the new mobile agent object. It provides the object by Encrypted Statement objects. It generates that by using Encryption class, different secret keys and statement codes. Also, by using the Encryption class, it encrypts the mobile agent object by another secret key before this object moves to the next service provider. Figure 5-9 shows this web form.

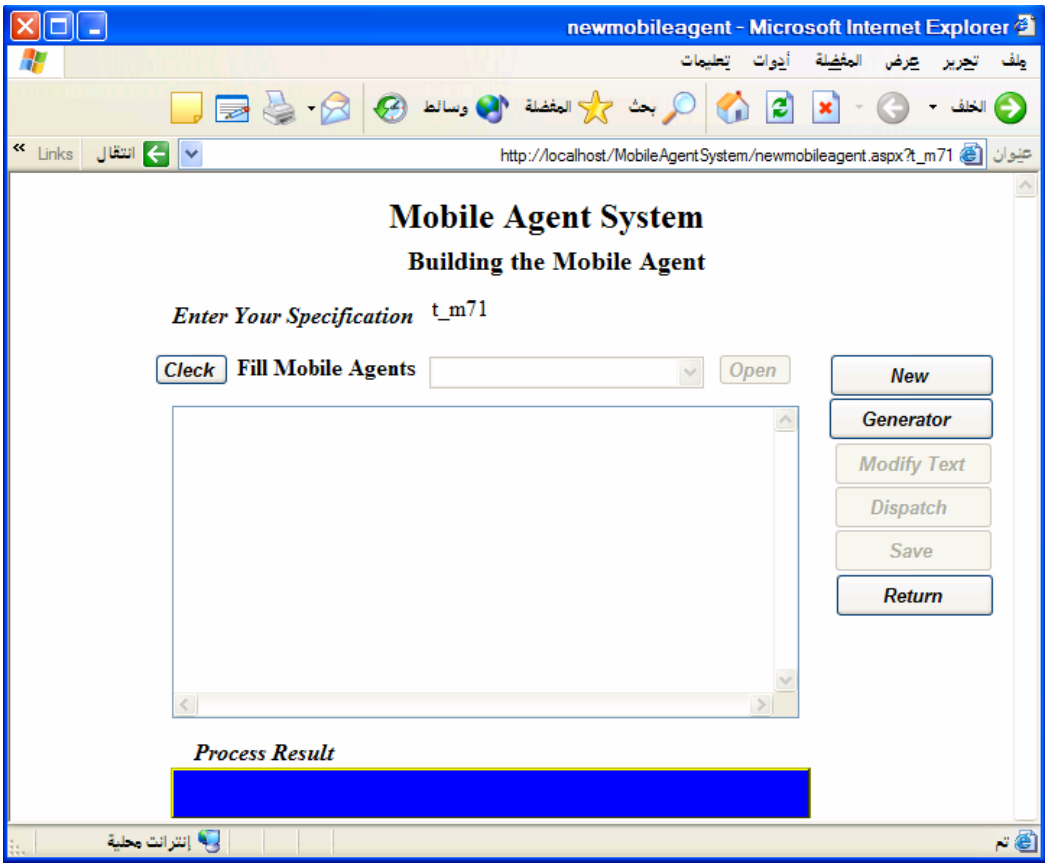


Figure 5-9 Mobile Agent Builder Web Form

5.6.12.5 Mobile Agent Displayer Web Form

A user can show his/her mobile agent specifications through this form. Also, this form allows the user to manage his/her mobile agent specification. Figure 5-10 shows this form.

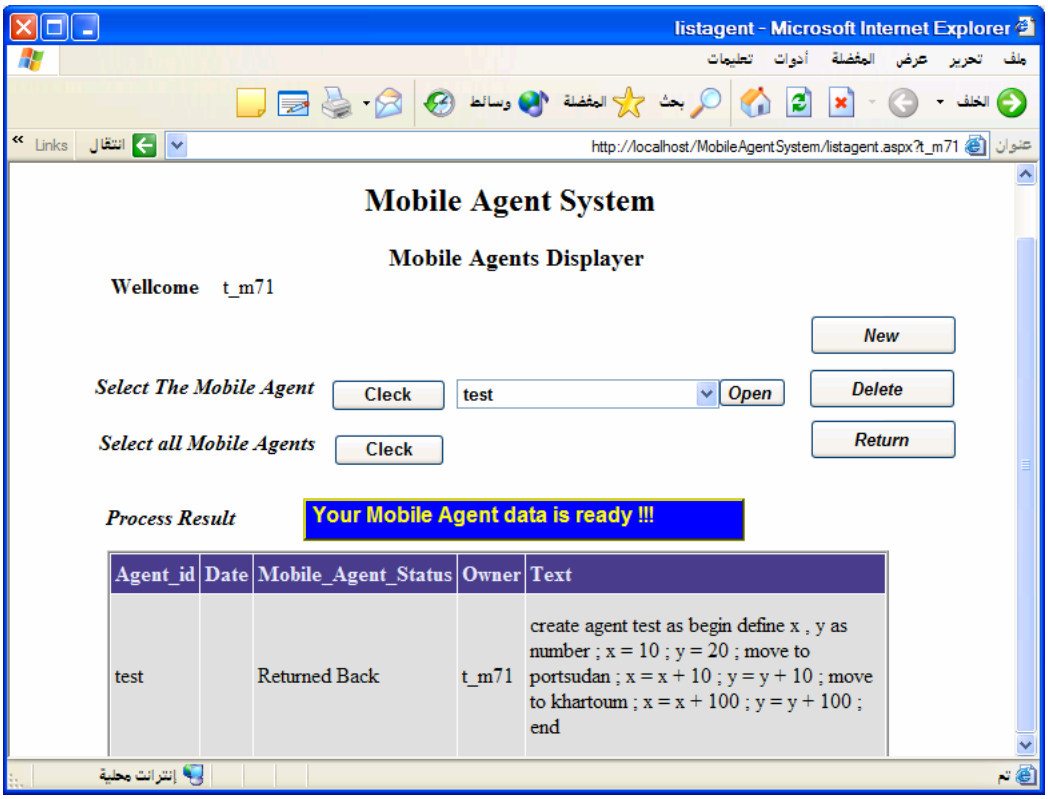


Figure 5-10 List Mobile Agent Web Form

5.6.12.6 Report Web Form

Through this web form, the user can get the results of the mobile agent journey. Also, the user can get a report about the mobile agent in each place that it visits. Figure 5-11 shows this web form.

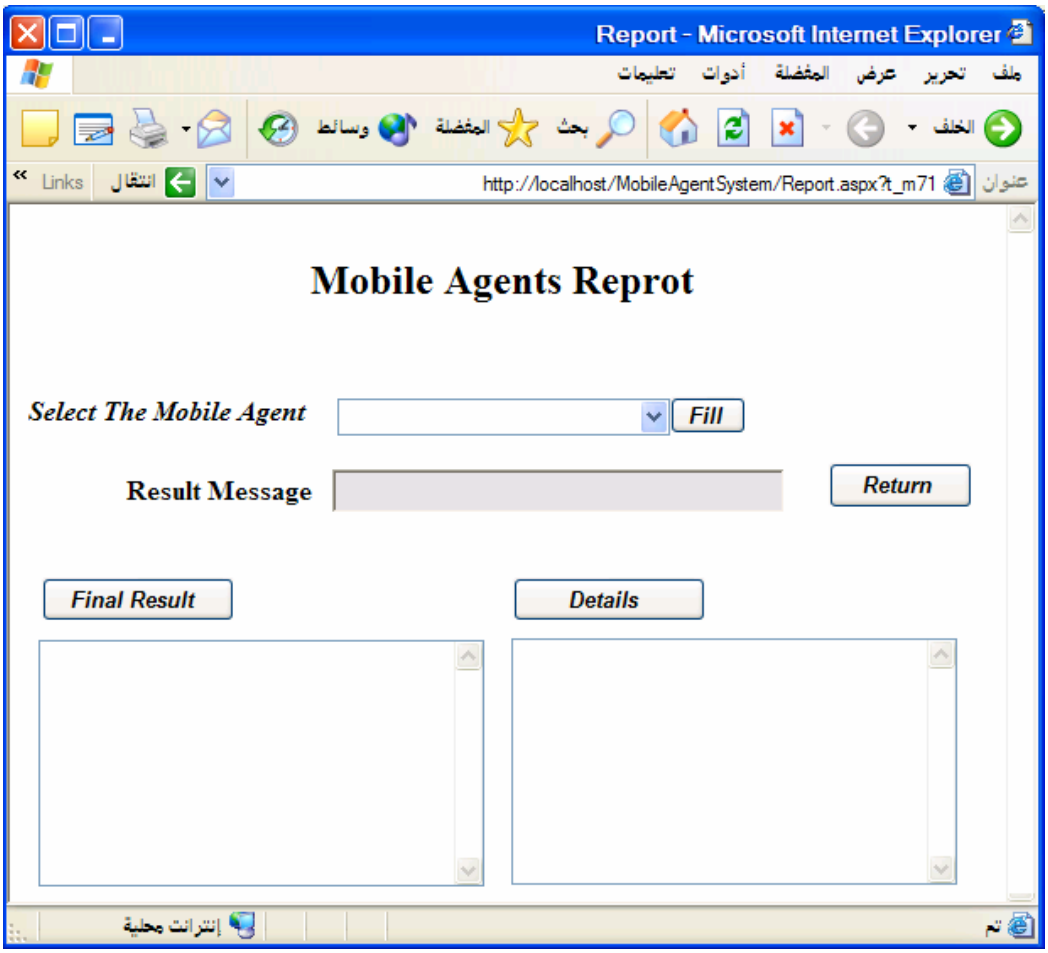


Figure 5-11 Report Web Form

5.6.13 Service Provider Class

This class represents a place that provides visiting mobile agents by services. This class uses Run Services Class to provide the services. The services may be different from a service provider to another. It consists of the following members:

5.6.13.1 Socket Data

This class has a listener that accepts information. This information is stored in a Socket Data member.

5.6.14.2 Serve Agent Method

This method receives a socket data and it delivers it to the mobile agent service method of the Run Service Class in order to get the mobile agent object. Also, it gets from the mobile agent code statement objects that will be executed in the current location. This method decrypts the code statement objects by using decrypt method. In addition, it delivers the code statement object after decrypting to the Agent Virtual Machine. If the code statement is a call service type or a mobility statement type, the system will call Resource Service method of the Run Service class to execute it.

5.6.14.3 Main Method

This method controls actions of this class. It defines a listener according to an IP address and a Port number. Through infinite loop, it accepts information from different parts of the system. When it receives information, it creates a new object of the class. By using Thread class, it creates a new thread object for processing this information. The thread calls Serve Agent method of the object.

5.6.15 Controlled Mobile Agent Class

An object of this class represents mobile agent information that is hosted by the controller before the mobile agent moves to specific service provider. It consists of the following members:

5.7.14.1 Mobile Agent Name

This member consists of the mobile agent name plus the owner's name.

5.7.14.2 Mobile Agent

This member represents the visiting mobile agent object.

5.7.14.3Arrival Time

This member represents an arrival time of the mobile agent object to the controller.

5.7.14.4 Changed Range

This member is for storing information about all code statements that will be changed on arrival to the service provider. The controller accepts modifications in these encrypted statement objects.

5.7.14.5 Public Keys Array

This member is used to store all public keys as RSAParameters objects that are used to generate signatures for encrypted statement objects in the mobile agent except that will be executed in the service provider.

5.7.14.6 Signatures Data Array

This member is used to store all signatures data of the encrypted statement objects of the mobile agent object except that will be executed in the service provider.

5.7.14.7Constructor Method

A constructor method initials all the above members. It receives the mobile agent object. Also, it generates the mobile agent name. Moreover, it stores the mobile agent object, the arrival time, the changed range, the public keys array and the signature data array by using RSACryptoServiceProvider class.

5.7.15 Controller class

A controller class is used to protect the mobile agent object from malicious services providers. Before the mobile agent moves to the service provider, it can be stored in the controller machine (trust place) according to the user's request. This class protects the mobile agent object when the service provider prevents it to continue its journey, the service provider is out of the service or it attacks the mobile agent. The Controller class consists of the following members:

5.7.15.1Socket Data

This class has a listener that accepts information. This information is stored in the Socket Data member.

5.7.15.2 Controlled Mobile Agent List

This member is an array of the controlled mobile agent objects. Each element represents the mobile agent object that visits the controller before the service provider.

5.7.15.3 Mobile Agent Check Method

This method receives the controlled mobile agent object and the mobile agent object as parameters. The controlled mobile agent represents the mobile agent object before it moves to the service provider. The mobile agent object represents the mobile agent object when it returns to the controller. By using the two objects, the method detects any attack may occur to the mobile agent object. In case an attack occurs, the method allows the mobile agent object which is stored in the controlled mobile agent object to continue the journey. Otherwise, the mobile agent object continues the journey.

5.7.15.4 Dead List Member

This list contains the mobile agent names that will be killed if they visit the controller.

5.7.15.5 Found method

This method receives the controlled mobile agent list and the mobile agent object. It checks if the mobile agent object is stored in the controlled mobile agent list or not. If it available, it replies true and the position of it in the list as integer number. Otherwise, it replies false and -1 as the integer to indicate that the mobile agent does not exist.

5.7.15.6 Mobile Agents Monitor method

This method monitors all mobile agents that are stored in the controlled mobile agent list. For each mobile agent that is stored the list more than specific time that means the service provider prevents the mobile agent from returning to the controller. So, it allows the mobile agent that is stored in the controlled mobile agent list to continue its journey. These operations are executed in an infinite loop. After that, it saves the mobile agent name in the dead list in order to kill it in case it returns after that from the service provider.

5.6.15.7 Serve Agent Method

This method receives the socket data member and it delivers it to the mobile agent service method in the Run Service class in order to get the mobile agent object. It uses Found method to check if the mobile agent exists in the list or not. In case the mobile agent is in the list, it uses the mobile agent check method to detect any attack in the mobile agent. Otherwise, it stores the mobile agent object in the list. After that, it dispatches the mobile agent object to the service provider.

5.6.15.8 Main Method

This method controls actions of this class. It defines a listener according to the specific IP address and port number. It creates a thread object and it executes mobile agent monitor method on it. Also, through infinite loop, it accepts information from different parts of the system. When it receives information, it creates a new object of the controller class. By using Thread class, it creates a new thread object for processing this information. Each thread calls the serve agent method.

5.8 System Algorithms

As described in above, many classes are built in the SMAG system. the following paragraphs show some algorithms of the main classes control the system life cycle. These classes are the mobile agent base Interfaces class, the mobile agent base class, the service provider and the controller class.

5.7.1 Mobile Agent Base Interfaces Algorithm

The logic steps of this algorithm are as follows:

1. A user enters his/her specifications which are represented mobile agent tasks through the mobile agent builder web Form.

2. The system generator checks the syntax of the specification.

2.1 If the syntax are correct, the generator will generate the code statement objects which are represented the specifications.

2.2 Otherwise, the system will report syntax error to the user and the user must correct that error to continue.

3. The system dispatcher does the following:

3.1 It encrypts each code statement object by using Encryption class and the secret key of the service provider that statement will be executed on. The result of this operation is the Encrypted Statement objects.

3.2 It creates a new mobile agent object and it provides the mobile agent with the encrypted statement objects.

3.3 It transfers the mobile agent to the mobile agent base.

4. End

5.7.5 Mobile Agent Base Algorithm

There are two cases in this algorithm. The first case, the mobile agent comes from the mobile agent interface class to start its journey. The second case, the mobile agent completes its journey and returns home.

The logic steps of algorithm in the first case:

1. The mobile agent base receives the mobile agent object from the mobile agent base interface.
2. Saving a current status of the mobile agent.
3. By using the resource service method in the run service class, it dispatches the mobile agent object to the first service provider.
4. End.

The logic steps of algorithm in the second case:

1. The mobile agent base sets a machine IP address and a port number.
2. It creates and starts the listener.
3. For each data comes to the mobile agent base

   3.1.1 The listener is ready to accept information through the AcceptSockect method.

   3.2 The socket method accepts the information.

   3.3 Creating a new object of the class.

   3.4 It stores this information in the Data Socket member of the Object.

   3.5 Creating a new thread object.

   3.6 Starting the thread with executing the serve agent method of the class.

   3.7 By using the Mobile Agent Service method, the new mobile agent object is created according to the socket data member.

   3.8 Extracting all information from the mobile agent object.

3.9 Saving all information in the system database

4. End.

5.7.6 Service Provider Algorithm

The following algorithm is used by each service provider:

1. The service provider sets a machine IP address and a port number.
2. It creates and starts the listener.
3. For each data comes to the service provider:

3.1.2 The listener is ready to accept information through the AcceptSockect method.

3.2 The socket method accepts the information.

3.3 Creating a new object of the class.

3.4 It stores this information in the data socket of the object.

3.5 Creating a new thread object.

3.6 Starting the thread with executing the serve agent method of the object.

3.7 By using the mobile agent service method, the new mobile agent object is created according to the socket data member.

3.8 specifying encrypted statement objects that will be executed in the service provider.

3.9 For each encrypt statement object:

3.9.1 Decrypting the encrypted statement by using encryption class and the secret key of the service provider to get code statement object.

3.9.2 Executing the code statement object by using the agent virtual machine.

3.9.3 In case the type of the code statement is the mobility type:

3.9.3.1 Saving the current status of the mobile agent.

3.9.3.2 Encrypting this code statement again by using the encryption class and the secret key that is stored in the code statement to generate the encrypted statement object again.

3.9.3.3 Dispatching the mobile agent to the next station by using the resource service method in the run service class.

3.9.4 In case the type of the code statement is a call service:

3.9.4.1 Executing the code statement by using the resource connection method of the run service class.

3.9.5 Encrypting this code statement again by using the encryption class and the secret key that is stored in the code statement to generate encrypted statement object again. This is done, in case the code statement type is not a mobility statement.

4. End.

5.7.7 Controller Algorithm

The following algorithm is used by each controller:

1. Creating a new thread object.
2. Staring thread with executing the mobile agents monitor method of the class.
3. the controller sets a machine IP address and a port number.
4. It creates and starts the listener.
5. For each data comes to the service provider.

5.1 The listener is ready to accept information through the AcceptSockect method.

5.2 The socket method accepts the information.

5.3 Creating a new object of the class.

5.4 It stores this information in the data socket member of the object.

5.5 Creating a new thread object.

5.6 Starting thread with executing Serve Agent method of the object.

5.7 By using the mobile agent service method, a new mobile agent object is created according to the socket data member.

5.8 By using Found method, the controller checks if the mobile agent in a controlled mobile agent list or not.

5.8.1 If the mobile agent exists in the list:

5.8.1.1 Getting its position in the list.

5.8.1.2 By using the mobile agent check method, the controlled mobile agent object and the mobile agent object, the controller checks if the mobile agent is attacked by the service provider or not.

5.8.2 If the mobile agent doesn't exist, the controller creates a new object of the controlled mobile agent class and it saves it in the list

5.9 The controller dispatches the mobile agent to the next station (the service provider or home)

6. End.

Chapter 6

SMAG Usability: Distributed Bookshop System Using Mobile Agent

6.1 Introduction

One of the applications that the mobile agent system deals with is the e-commerce applications. This application type is chosen in order to illustrate the SMAG system. The e-commerce application allows users to perform business transactions. The e-commerce applications use the network to achieve its tasks [14]. The mobile agent can help in this field by locating appropriate offering, negotiating and executing business transactions on behalf of its owner. In order to test the SMAG system environment, The Bookshop System has been developed by using the SMAG system as e-commerce application. The main objective of the Bookshop system is to exercise the different operations provided by the SMAG system.

6.2 Bookshop system

A bookshop system is a distributed application system that allows users to deal with many bookshops. It uses the SMAG system components to achieve its duties. Through the bookshop system, the users can perform many transactions by using the mobile agent technology, such as, searching books information and buying books. In the following paragraphs, the bookshop system infrastructure is described by using the SMAG system components:

6.2.1 System User

A system user is a person who has specific requirements, for example, buying books, looking for a book and getting a book price. He/she wants a mobile agent to perform these requirements on him/her behalf. The user must have an access account in order to be able to use the bookshop system. The system user must be familiar with MASL syntax to specify a mobile agent algorithm. Also, he/she must know the services description in each bookshop to achieve transactions through them.

6.2.2 Mobile Agent

A mobile agent in the bookshop system represents a user in each bookshop that the mobile agent visits. It performs business transaction on behalf of the user. By using the bookshop web site, the user inputs his/her requirements specification through MASL. The mobile agent base of the bookshop system generates the mobile agent according to this specification.

6.2.3 Bookshop Web Site

The system user can access the bookshop system through the internet. A client node of the system is represented as a web site. The web site provides the user with different operations, for example, receiving business transactions specification, requesting to generate the mobile agent, requesting to dispatch the mobile agent and managing the mobile agents.

6.2.4 Bookshop Agent Base

This component represents the mobile agent base. It achieves different operations that are requested through the bookshop site. It generates the mobile agent according to the user specification and dispatching the mobile agent to the first station by using its itinerary table. Also, this component receives and stores the mobile agent when it returns from a journey.

6.2.5 Bookshop

A bookshop represents a place that sells books. In the SMAG system, it represents the service provider. The bookshop has a database that maintains books information. It receives visiting mobile agents and provides them with business services. Each bookshop has three services as follows:

1. Book-Found Service: this service checks if a book is available or not. It accepts the book information and it replies "found" or "not found".

2. Get-Book-Price service: this service gets a price of the book. It accepts the book information and it gives the price of the book.
3. Buy-Book: through this service the mobile agent can buy the book. It accepts book information, the user's address and the credit-card number and it answers "success" or "Failure". The Success value means a transaction is successfully achieved. The Failure value means a transaction has failed.

Each bookshop registers itself in a Bookshops administrator to be a member of the bookshop system. Through the Bookshops administrator, it publishes these services and their descriptions.

6.2.6 Bookshops Administrator

This component represents the system administrator. It saves all system components information, for example, bookshops IP addresses and port numbers, and services name and descriptions. Also, it exchanges security information among different parts of the system.

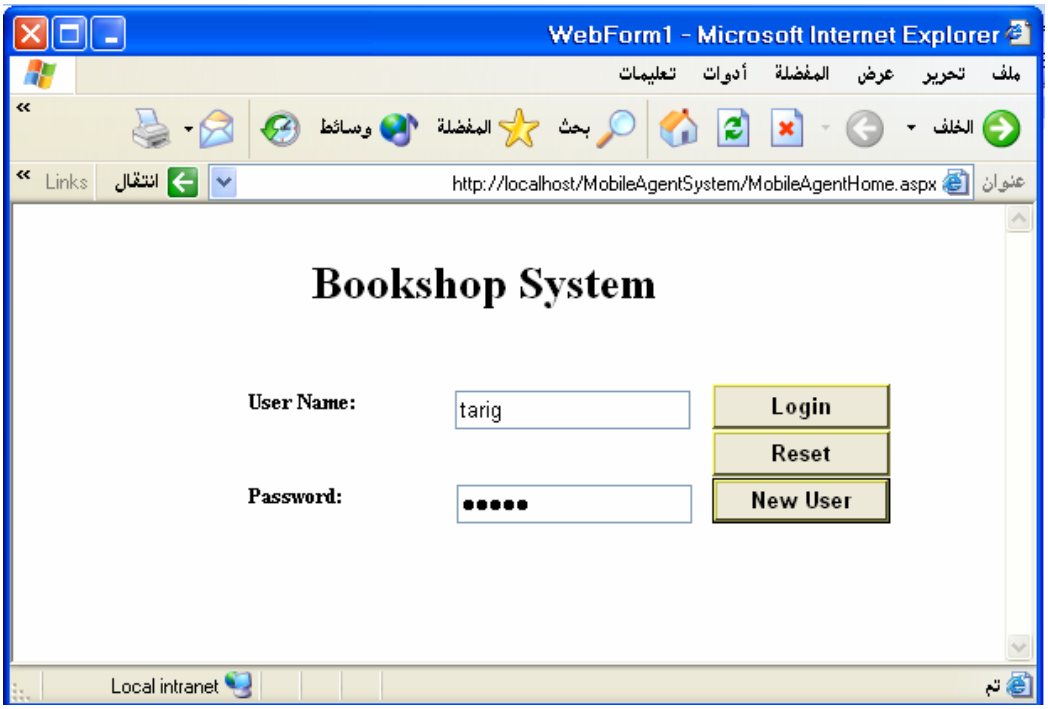
6.3 Scenarios

To demonstrate the bookshop system transactions, different scenarios are shown. Each scenario represents tasks that the user needs to achieve through the bookshop system. The scenarios assume there are two bookshops: PortSudan bookshop and Juba bookshop. All these bookshops are members in the system. Each one has the Book-Found service, the Get-Book-Price service and the Buy-Book service.

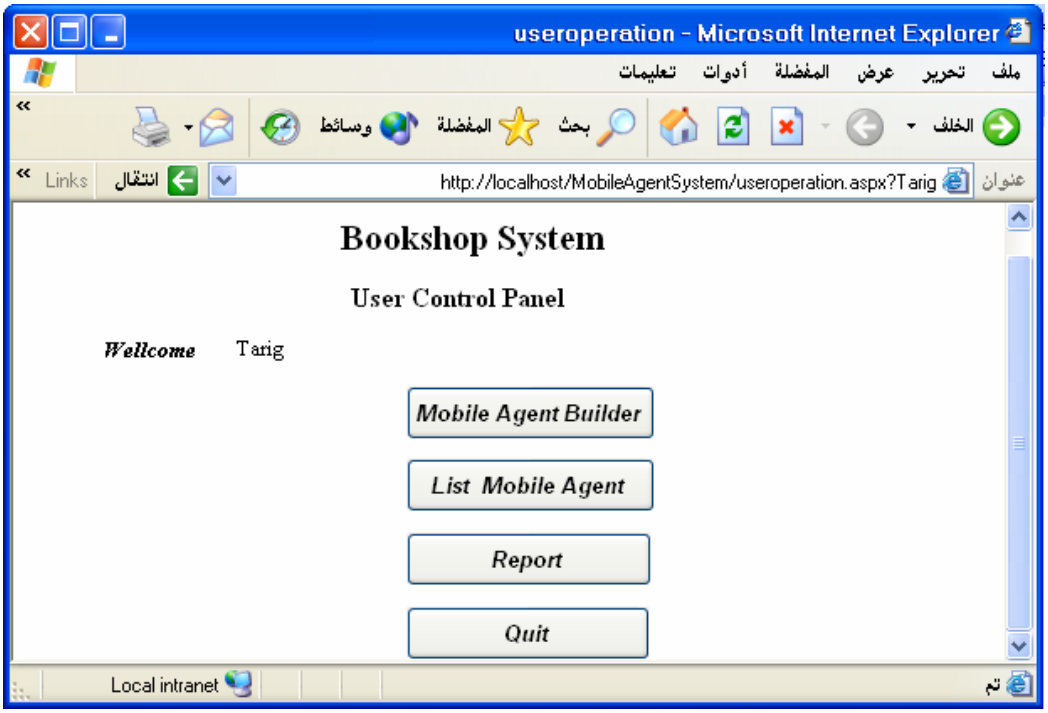
6.3.1 Scenario No. 1

A user wants the price of java book from ProtSudan bookshop. The following forms show the user requirement:

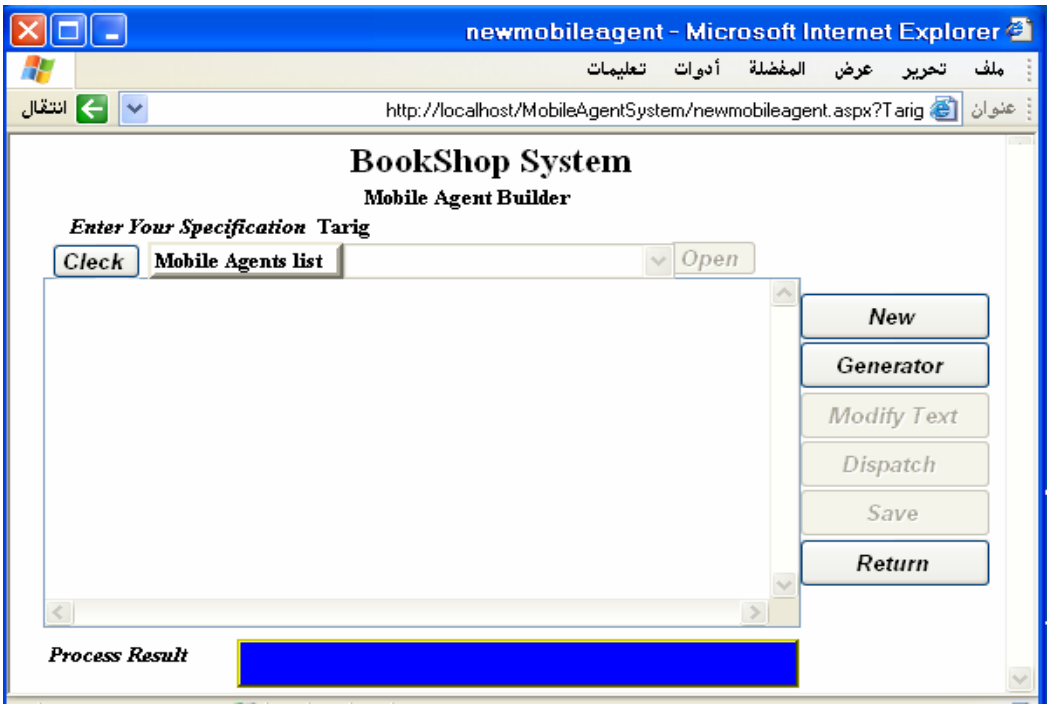
By using the user ID and password the user accesses the system:



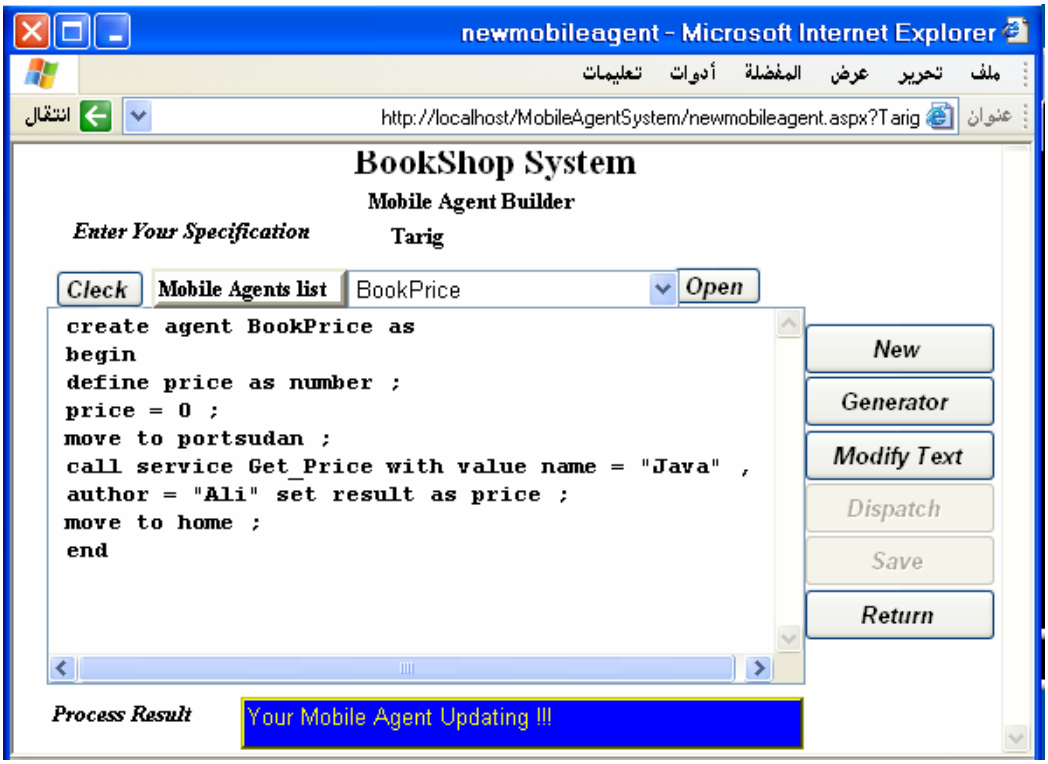
The user clicks the mobile agent builder:



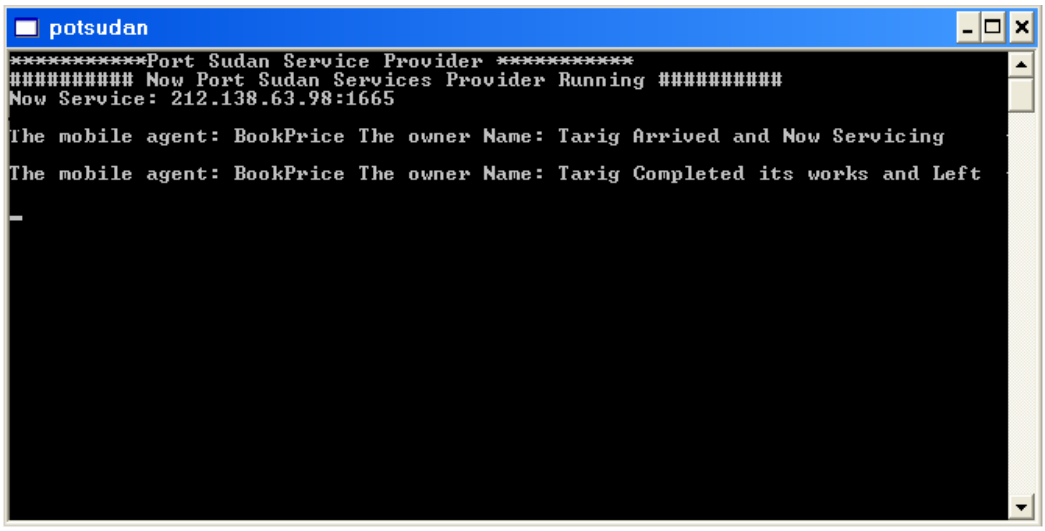
The mobile agent builder form:



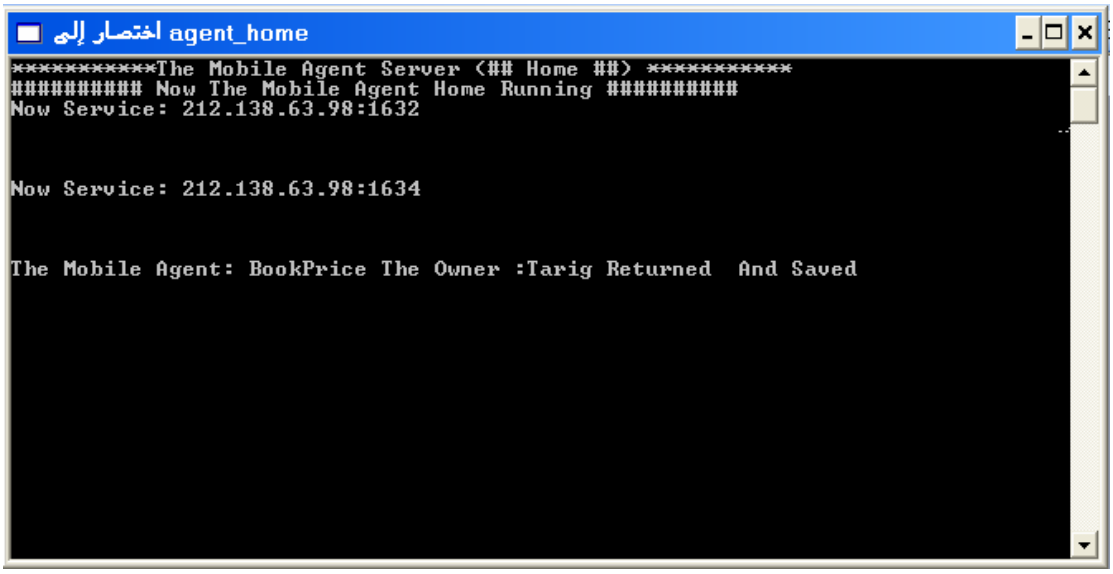
The user writes his the requirement specification:



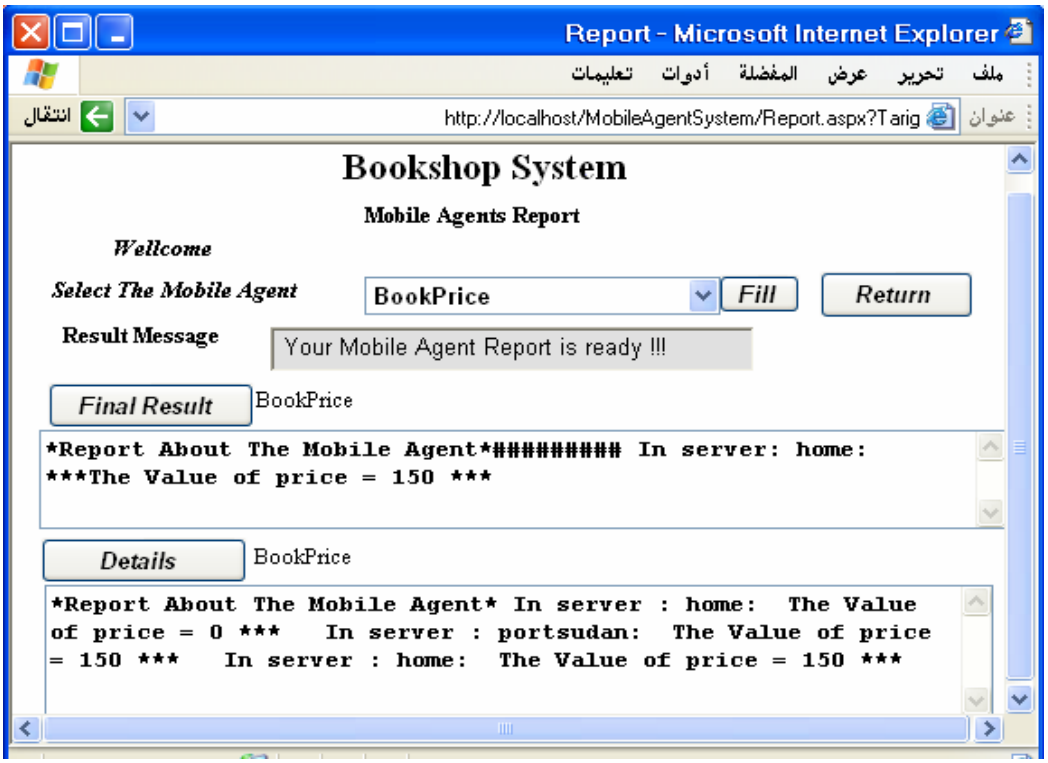
The mobile agent completes its tasks in ProtSudan Bookshop and it leaves.



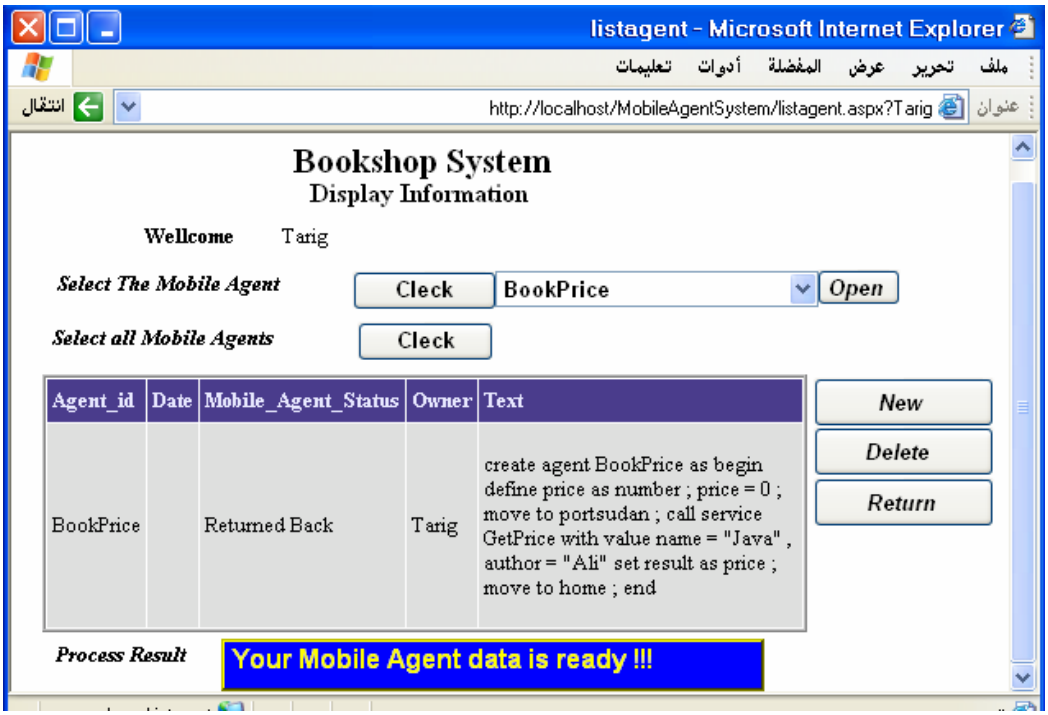
The mobile agent arrives home and it is kept.



The report of the mobile agent after it returns home:

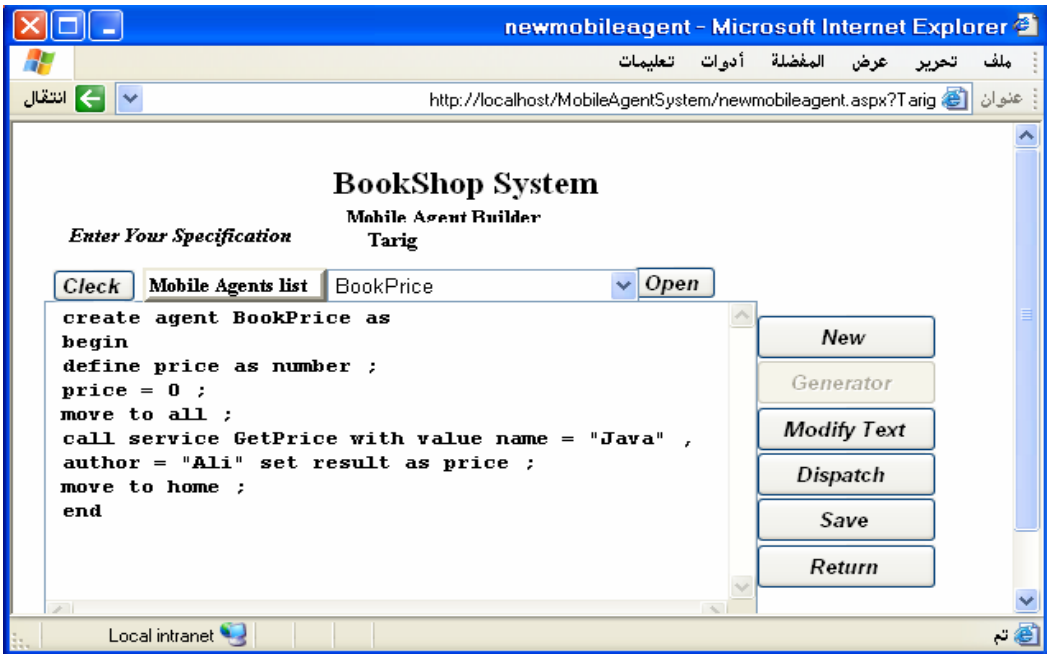


The mobile agent information:



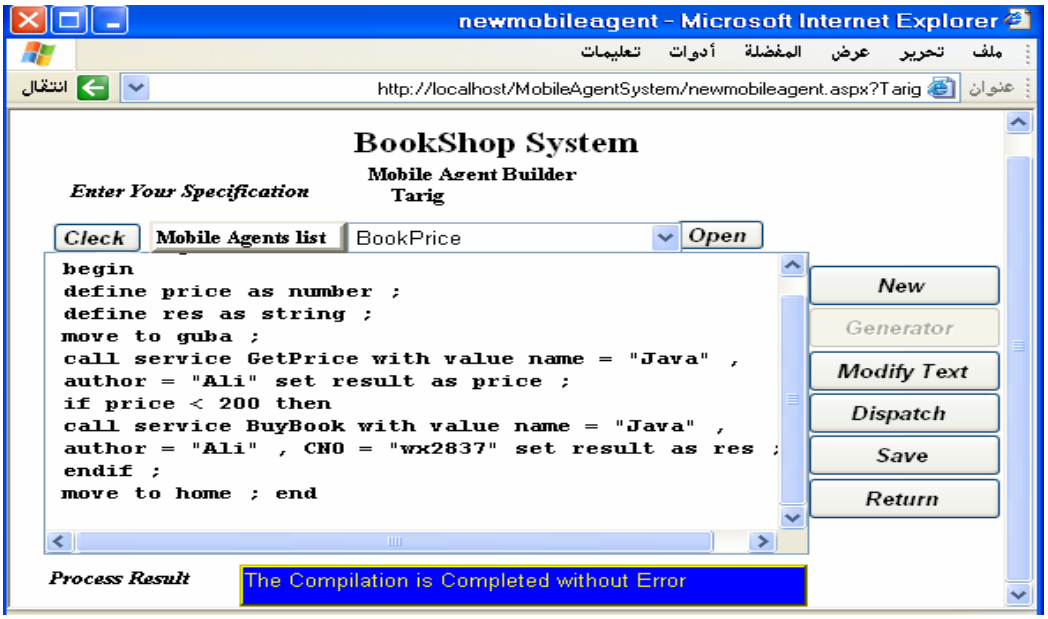
6.3.2 Scenario No. 2

The user needs Java book price from all Bookshops:



6.3.3 Scenario No. 3

The user wants to buy the Java book from Juba bookshop if the price is less than $200:



Conclusions and Future Works

1. Introduction

A mobile agent is an autonomous piece of code that can migrate under its control from a machine to another in a heterogeneous network. It is a new technology for computers to communicate. By using the mobile agent system, the users can achieve many and different tasks in different places. The mobile agent model uses a remote programming concept (asynchronous). The remote programming concept allows the mobile agent to execute its task in hosts (remote machines). So, the mobile agent system components need protection. This thesis introduced a new mobile agent model that protects all the parts of the mobile agent system. The model is implemented in the SMAG system that is easy to use and power to achieve transactions.

2. Thesis Contributions

In this section, brief descriptions of the contributions provided by this thesis are given:

- A new Mobile Agent Specification Language (MASL) is designed and implemented. The user of the SMAG system can use this language to describe his/her requirements that will be achieved by a mobile agent. This language is very simple and flexible to use. The syntax of the language statements are appropriated to the mobile agent concepts. By using this language, users cannot be able to represent any malicious operations.
- The SMAG system is designed and implemented out to provide a generic model for a mobile agent system that can be worked in different distributed system application areas. The model includes architectures of several components. The SMAG system provides the users of the system with friendly interface screens. Through these interfaces, they describe and control their

mobile agents. It also provides many functions such as mobility, communication, storage …etc of the mobile agents.

- The SMAG system generates a mobile agent according to their owner's specification through MASL. The generation of the mobile agents is done in places isolated from the users. So, the system will accept only the mobile agents that are generated only by the system. Exactly, the generation is done in the mobile agent base.
- To protect a mobile agent, the SMAG system uses a View Security Mechanism (VSM). The VSM is designed and implemented in this thesis. It uses cryptography mechanisms to protect a mobile agent. The main idea of this mechanism is to encrypt different parts of the mobile agent by different secret keys, so the mobile agent will own different views during its journey. Each encrypted part can be only decrypted by a specific service provider that manipulates this part. Thus, the VSM will completely protect the mobile agent from any illegal operations.
- To protect a mobile agent in case a service provider kills a mobile agent or alters any other parts that are not specified for it, the SMAG system uses controllers and Safe date mechanism to deal with this problem. The controller saves a copy of the mobile agents and data digest before dispatching it to the service provider. When the mobile agent returns to the controller, the mechanism checks any effect that occurs. This mechanism is designed and implemented in this thesis.
- For protecting the information that a mobile agent collects through its journey, the Simulator Unit is designed. When the mobile agent returns home, this unit will check all the information that is collected by the mobile agent by

executing again the algorithm of the mobile agent locally and making a comparison between two results.

- An Agent Virtual Machine (AVM) is one of the contributions that give the model power in the security area. The AVM is an interpreter that interprets mobile agent behaviour in each visited place. For security reasons and to minimize the mobile agent size, this unit is hosted only by the service providers and the mobile agent base. So, the mobile agent carries only code representations executed by the AVM. This approach allows the development in the MASL.
- Dynamic Behaviour Generation: The SMAG system supports dynamic behaviour generations. So, a mobile agent can visit new places that are not predefined. Through the controllers, the dynamic behaviour is generated. Each controller is provided with a garbage collection unit that frees some parts of the mobile agent in order to be used by the dynamic behaviour

3. Future Works

This thesis introduced new security mechanisms to protect the system in all stages. As future works of the research, some points are suggested in this section:

- Most of these mechanisms have been implemented and experimented in the SMAG system. On the other hand, some of these mechanisms are introduced as designed idea such as the Dynamic Behaviour Generation mechanism, the Garbage Collection mechanism and the Simulator mechanism. These mechanisms could be implemented.

- The SMAG system generates the mobile agents by using MASL. The SMAG system model can accept new suggested languages that are more appropriate to several fields.
- In order to increase the efficiency and the performance of the SMAG system, artificial intelligent mechanisms can be added.

References

[1] G. Karjoth, N. Asokan, C. Gulcu , *Protecting the Computation result of Free-roaming Agents*, Proceedings of Second International Workshop, Mobile Agent 98. Verlage Lecture Notes in Computer Science, Vol. 1477, PP 195-207, 1998.

[2] Karnik, Neeran, *Security in Mobile Agent Systems,* Ph.D. dissertation. Department of Computer Science and Engineering, University of Minnesota, 1998

[3] B.H. Tay, A. Ananda, *A Survey of Remote Procedure Calls,* Operating system Review, 24(3), PP 63-79, July 1990.

[4] R. S. Gary, *Agent Tcl: A flexible an Secure Mobile Agent System*, Fourth Annual Tcl/Tk Workshop (TCL 96) (Monterey, California, July 1996), M Diekhans and M Roseman, editors, July 1996.

[5] A. D. Birrell , B. J. Nelson, *Implementing remote procedure call*, ACM Transaction on Computer Systems, 2(1): PP 39-59, 1984.

[6] J. Stamos, D. Gifford, *Remote Evaluation,* ACM Transaction on Programming Languages and Systems, 12(4): PP 537-565, 1999.

[7] J. Vittal, *Active Message Processing: Message as Messengers*, In R. P. Uhlig, editor, Computer Message System, PP 175-195. North-Holland, 1981.

[8] J. S. Banino, *Parallelism and Fault Tolerance in Chorus*, Journal of Systems and software, PP 205-211, 1986.

[9] G. Cornell, C. Horstmann, *Core Java. Sunsoft press*, 1997.

[10] James E. White, *Telescript technology: foundation for electronic marketplace.* General Magic White Paper , General Magic, Inc, 1994.

[11] Ted G. Lewis, *where client / server Software heading?,* IEEE Computer, PP 49-55, April 1995.

[12] D. Chess, B. Grosof, C. Harrison. D. Levine, C. Parris and G. Tsudik, *Itinerant Agents for Mobile Computing*, IBM T. Journal Wastson Research Center, New York , 1995.

[13] R. Gary, D. Kotz, G. Cybenko and D. Rus, *Mobile Agents and state-of-art Systems*, Technical paper, Dartmouth Collage, Hanover, 2000.

[14] A. Lingnau, O. Drobink, *An Infrastructure for mobile agents, requirement and architecture*, Johann Wolfgang Goethe University, Germany, 1995.

[15] D. B. Lange, M Oshima, *Seven Good Reason for Mobile Agent*, Communications of the ACM, 42(3): PP 88-98, 1999.

[16] J. White, *Mobile Agent Telescript*, Technical White Paper, General Magic, 1995.

[17] M. Breugst, T. Magedanz, *Mobile Agents – Enabling Technology for Active Intelligent Network Implementatio*, IEEE Network Magazine, 12(3): PP 53 – 60, 1998.

[18] A. Bieszczad, B. Pagurek, *Towards Plug-and-Play networks with mobile code*, International Conference for computer communication ICCC'1997, France, 1997.

[19] M. Strasser and K. Rothermel, *Reliability Concept for Mobile Agents*, International Journal of Cooperative Information Systems, 7(4): PP 355-82, 1998.

[20] G. Vigna. Mobile Code Technologies, *Paradigms and Applications*, PhD thesis. Politecnico di Milano , Italy, 1997.

[21] A. Fuggetta, G. P. Picco and G Vigna, *Understanding Code Mobility*, IEEE Transactions on Software Engineering, 24(5):342-61, 1998.

[22] J. Baumann, F. Hohl and N. Radouniklis, *Communication Concepts for Mobile Agent System*, Mobile Agents – First International Workshop , MA'97, Germany, 1997.

[23] T Finin, R. Fritzson, D. McKay and R. McEntire, *KQML – A Language and Protocol for Knowledge and Information Exchange*, Technical Report CS-94-02. University of Maryland, 1994.

[24] T Finin, Y. Labrou and Mayfield, *KQML as an Agent Communication Language*, In J. M. Bradshaw and Jeffery, editors, Software Agents, PP 291-316. MIT Press and American Association for Artificial Intelligent, 1997.

[25] W. Lugmayr, *Gypsy: A Component-Oriented Mobile Agent System*. Distributed Systems Group, Technical University of Vienna, Austria. October 1999.

[26] W. Jansen, T. Karygiannis, *NIST Special Publication 800-19-Mobile Agent Security*, Technical paper , National Institute of Standards and Technology, Computer Security Division.

[27] J. Baumann , K. Rothermel, *The Shadow Approach: An Orphan Detection Protocol for Mobile Agent*, Second International Workshop, Germany ,1998.

[28] T. Iizuka, A. Lau and T. Suda, *A Design of Local resource access control for mobile agent in PDA*, IEICE Trans COMMU, VOL. E84-B, 2001.

[29] R. Wahbe, S. Lucco and T. Anderson, *Efficient Software-Based Fault Isolation*, Proceedings of the Fourteenth ACM Symposium on Operating System Principles. ACM SIGOPS Operating System Review, pp 302-216. 1993. *URL:http://www.cs.duke.edu/~chase/vmsem/readings.html*

[30] L. Gong, *Secure Java ClassLoading*, IEEE Internet Computing, 2(6), November 1998.

[31] N. Borselius, N. Hur, M. Kayprynski and C. Mitchell , *A Security Architecture for Agent-Based Mobile Agent Systems*, Mobile Communications Technologies Conference Publication No. 489 IEE, 2002.

[32] A. Rubin and Jr D. E. Geer, *Mobile Code Security. IEEE Internet Computing*, 2(6), November 1998.

[33] J. Algesheimer, C. Cachin, J. Camenish and G. Krjoth, *Cryptographic Security for Mobile Code*, IBM Research , 2003.

[34] B. Hashii , M. Lal , R. Pandey, and S. Samorodin, *Secure System Against External Program*, Internet Computing , 2(6) , November 1998.

[35] J. K. Ousterhout, J. Y. Levy, and B. B. Welch, *The Safe-Tcl Security Model*. In G. Vigna, editor, Mobile Agent and Security, Vol. 1419. 1998.

[36] S. Gritzalis , G Agglis, *Security Issues Surrounding Languages for Mobile Code: Java vs. Safe –Tcl*, Operating System Review, 32(2):PP 16-32, 1998.

[37] G. Neucla and P. Lee, *Safe kernel Execution without Run-Time Checking*, Proceeding of 2nd symposium on Operating System Design and Implementation (OSDI'96), Washington, 1996.

[38] J. Ordille, *When Agent Roam, Who Can You Trust?*, Proceeding of the first conference on Emerging Technologies and Application in Communications, Portland, 1996.

[39] W. Farmer, J. Guttman and V. Swarup, *Security of Mobile Agents: Authentication and State Appraisal*, the 4th European Symposium on Research in Computer Security , pp 118-130, 1996.

[40] G. Karjoth, D. B. Lang, Oshima, *A Security Model for Aglet. IEEE Internet Computing* , 1(4) , 1997.

[41] M. Giansiracusa, *Mobile Agent Protection Mechanisms*, and the Trusted Agent Proxy Server (TAPS). Information Security Research Center, Australia, 2003.

[42] TACOMA. University of Tromso, july 1999. *http://www.tacoma.cs.uit.no/*.

[43] G. Vigna, *Cryptography Traces for Mobile Agents*, In G. Vigna , editor, Mobile Agent and Security, volume 1419, 1998.

[44] A. Suen, *Mobile Agent Protection with Data Encapsulation and Execution Tracing*, Master Thesis, The Florid State University, 2003.

[45] F. Hohl, *Time Limited Blackbox Security: Protection Mobile Agent From Malicious Hosts*, In G. Vigna , editor, Mobile Agent and Security , PP 92-113 ,1998.

[46] T. Sander , C. F. Tschudin , *Protecting Mobile Agent Against Malicious Hosts*,In G. Vigna , editor, Mobile Agent and Security, Vol. 1419, 1998.

[47] Anand Tripathi, Neeran Karnik, *A Security Architecture for Mobile Agents in Ajanta*, Proceedings of the International Conference on Distributed Computing Systems, April 2000.

[48] B. S. Yee, *A Sanctuary for Mobile Agents, In Secure Internet Programming*, PP 261-273, 1999.

[49] J. Riordan and B. Schneier, *Environment Key Generation Toward Clueless Agents*, Technical Report, 1998.

[50] Hock Kim Tan and L. Moreau, *Mobile Code For Key Propagation*, Paper, Notes in theoretical Computer Science 63, UK, 2001.

[51] S. Poslad and M. Calisti. *Towards improved trust and security in FIPA agent plate form*, In Autonomous Agent Workshop, 2000.

[52] V. Roth, *Secure Recording of Itineraries through Cooperative Agents*, Proceeding of the ECOOP Workshop on Distributed Object Security and 4th

Workshop on Mobile Object Systems: Secure Internet Mobile Computation. PP 147-154, France, 1998.

[53] V. Roth, *Secure Recording of Itineraries through Co-operating Agents*, Technical paper, Germany, 2003.

[54] F. B. Schneider, *Toward Fault-Tolerant and Secure Agent*, proceeding 11th International Workshop on distributed Algorithm, Germany, 1997.

[55] D. Milojicic, M. breugst, I. Busse, J. Compbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, Virdhagriswaran and J. White, *MASIF – The OMG Mobile Agent System Interoperability Facility*, Mobile Agent – Second International Workshop, MA '98, Germany, 1998.

[56] M. Richards and N. Southeastern, *The State Security Standards For Mobile Agents*, Decision Sciences Institute, 2002.

[57] Z. Xu, *Survey of Some Mobile Agent System*, Imago Group, 2003.

[58] P. Doemel, *Interaction of Java and Telescript Agents*, Mobile Object System and International Workshop MOS'96, july 1996.

[59] Aglets SDK. IBM Corporation, July 1999. *http.trl.ibm.co.jp/aglets/* .

[60] D. Johansen, R. van Renesse and F. B. Schneider. *An Introduction to the TACOMA Distributed System*, Technical Report 95-23, University of Troms, 1995.

[61] R. S. Gary, *Agent Tcl: A flexible and Secure Mobile Agent System*, PhD thesis, Dartmouth College. June 1997.

[62] S. Nog. S. Chawla and D. Kotz, *An RPC mechanism for transportable agents*, Technical Report PCS-TR96-280, Dartmouth College. March 1996.

[63] J. Y. Leavy and J. K. Ousterhout, *A Safe Tcl Toolkit for Electronic Meeting places*, In Proceeding of the first USENIX Workshop on Electronic Commerce, PP 133-135, July 1995.

[64] B. Yee, *Monotonicity and Partial Results Protection for Mobile Agents*, technical paper, 2002.

[65] Concordia. Mitsubishi Electronic ITA, July 1999. *http://www.meitca.com/HSL/Projects/Concordia /*.

[66] Voyager. ObjectSpace, Incorporated, July 1999. *htpp://www.objectspace.com*.

[67] M. Strasser, J. Baumann, F. Hohl, N. Radouniklis, k. Rothermel, and M. Schwehm, *ATOMAS: Transaction-Open Multi Agent System*, Annual Report. Technical Report. University of Stuttgart, 1997.

[68] M. Strasser, J. Baumann and F. Hohl, *Mole – A Java based Mobile Agent System*, technical paper ,1997.

[69] G. Paramasivam,*Cryptography in Microsoft.NET Part II: Digital Envelop and Digital Signatures*, technical paper, *http://www.c-sharpcorner.com/Code/2002/Dec/DigitalEnvelop.asp*,2002.

[70] G. Paramasivam, *Cryptography in Microsoft.NET Part I: Encryption*, technical paper, *http://www.c-sharpcorner.com/Code/2002/Dec/CryptEncryption.asp* , 2002.

[71] B. Harvey, S. Robinson, J. Templeman, K. Watson, *C# Programming with the Public Beta*, Published by Wrox Press Ltd, UK, 2000.

[72] M. Parihar et al, *ASP.NET Bible, Published by Hungry Minds Inc*, New York, 2002.

[73] G. Paramasivam,Cryptography in Microsoft.NET Part III: Digital Certificates, , technical paper, *http://www.c-sharpcorner.com/Code/2003/Jan/DigitalCertIII.asp* , 2003.

[74] A. Suen, *Mobile Agent Protection with Data Encapsulation and Execution Tracing,* Master Thesis, The Florid State University, 2003.

Appendix

CC