CHAPTER 3

SYSTEM METHODOLOGY

3.1. Overview

This chapter explains about system development and design methodologies which are used in designing SAAIDS. The system design begins with system requirement analysis, system analysis, system design and database design. All these phase of system design are including Use Case Diagram, Activity Diagram, Sequence Diagram, Package Diagram, Class Diagram, Graphical User Interface (GUI) and Database Design.

3.2. Methodologies for System Development and Design

3.2.1. Prototype Model

Usually, researcher determines requirements for a system variously and does not specify input and output accurately. In this matter, researcher does not able to determine either the process model used will fulfill those requirements. Therefore, Prototype Model is the best process model in this case (Kendall & Kendall, 2002).

Prototype Model is the process model used in this development. Prototype Model as shown in Figure 3.1 provides a guideline seem like several cycles in completing development process. Prototype paradigm begins with requirements gathering. The researcher determines the main objectives of the projects, gathers all needed requirements and expected outcomes. Then, instant design is built. This instant design focuses on system presentation which is viewable to the user. For example, input and output format. The instant design leads to the design of the prototype.

After that, the instant design will be tested to ensure that all requirements are fulfilled. If there are any mistakes or misses requirements occurred, it will be repaired and completed in cyclic design. The cycles are depending on satisfaction of user on all requirements and at the time the developer can get better understanding on requirements needed and will do a better process in the next time. Practically, Prototype Model provides a mechanism to determine system requirements. If a functional prototype has been built, existing program and tools will be used in upgrading the system.

Prototype Model has several advantages compared to other process models. The advantages are stated as follows:

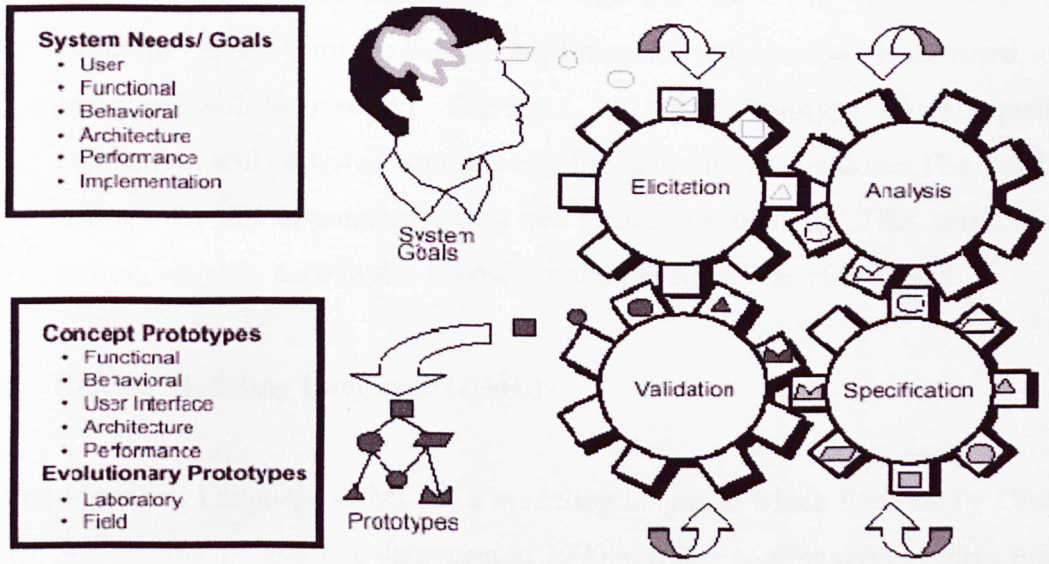

Figure 3.1: Prototype Paradigm

a. Cyclic system development

Prototype Model allows cyclic phase until user and system requirements are fulfilled. Meanwhile, Waterfall Model is not allowed cycles in any phase of development.

b. Concurrent system development

System development that used Prototype Model allows concurrences. For example, every development team has tasks in developing their phases and therefore, every developer team do not need to wait for other teams to continue to next phase. This is the opposite site of Waterfall Model which is not allowing the development of next phase before the existing on running phase not completed.

c. Support component or object based development

Prototype Model is able to support component or object based system development which is being called as Object Oriented Approach (OOP). This is different to structured approach. By using Prototype Model, system is seen as an object collection, not function collection.

d. Continuous system quality validation

When the design development is completed, user will do tests on the prototype. When error or lack in fulfillment of requirements is detected, the prototype will be repaired. Therefore, by using Prototype Model, quality verification will be tested continuously for each time cyclic done. The suitable analogy for this approach is "you see it and you trust it". This verification process which is used in this model is not allowed in Waterfall Model.

3.2.2. Unified Modeling Language (UML)

Unified Modeling Language (UML) is a modeling language which founded by Grady Booch, Jim Rumbaugh and Ivar Jaconson in 1990's. It is a combination of three main object oriented methods which are Booch created by Grady Booch, Object Modeling Technique (OMT) by Jim Rumbaugh and Object Oriented Software Engineering

(OOSE) by Ivar Jacobson (Kendall & Kendall, 2002). Table 3.1 views UML and Components.

In UML, object and link are referred as different components. The main component in UML is called thing although in many other modeling techniques used the word object (Kendall & Kendall, 2002). Structural Things provides a guideline for creating a model and links between models. Behavioral Things describes how the object is working meanwhile Group Thing is allowing notation adding in diagrams.

Relationship is a linker between objects. Meanwhile, Structural Relationship is combining objects in Behavioral Diagram. Structural Diagram is viewing links between classes and Behavioral Diagram is viewing interaction between human (actor) and Object or model is referred as use case.

Table 3.1: UML and Components

UML Category	UML Elements	UML Details
Things	Structural Things	Classes, Interfaces, Collaborations, Use Cases, Active Classes, Components, Nodes
	Behavioral Things	Interactions, State Machines
	Grouping Things	Packages
	Annotational Things	Notes
Relationship	Structural Things	Dependencies, Aggregations , Associations, Generalizations
	Behavioral Relationship	Communicates, Includes, Extends, Generalizes
Diagrams	Structural Diagrams	Class Diagrams, Object Diagrams, Component Diagrams, Deployment Diagrams

	Behavioral Diagrams	Use Case Diagrams, Sequence Diagrams, Collaboration Diagrams, Statechart Diagrams, Activity Diagrams

The special characteristic in UML is this method able to view mapping structure from requirement analysis until design phase and finally the implementation phase. UML gives consistent notations which ease the model to be used for user communication. Other benefit is UML able to view the system behavior, interaction between objects and system design.

The UML is chosen as one of system development method because of several advantages as follows:

a. UML is supported by well known used Prototype Model.
b. The object oriented approach which used by UML ease the understanding on requirement, specification and system design.
c. It is suitable for either big or small scale system development.
d. UML is a focused method which is widely used in current software development.

3.3. System Requirement Analysis

Requirement analysis intends to determine needed requirements for the system. All specifications either for software or hardware are being analyzed to document standard requirements for analysis and design phase. When all the requirements identified, use case module and activity diagram is designed. All kinds of requirements information translated to the use case and activity diagram by using Unified Modeling Language (UML). The requirements are categorized into two categories which are functional requirements and non-functional requirements.

3.3.1. Functional Requirements

Functional requirements are a set of requirements that enable system to fulfill functional tasks which are needed by user in overcoming the issues and problems mentioned before. In this system, the functional requirements are categorized into four main components which contain several modules as mentioned in architecture before.

3.3.1.1 Security

a. Authentication module must ensure that only authorized personal can install and remove an agent by using existing username and password with Single Sign-On and Java Authentication and Authorization Service (JAAS) technology.
b. Encryption module must be able to encrypt all messages being sent and safely decrypt when it reach their destination using Elgamal Encryption.
c. Digital Signature module must be able to make sure signing process between agents being well done using SHA1 with Elgamal Digital Signature Algorithm.

3.3.1.2 Transceiver

a. Agent Communication module must ensure that all messages sent in system are secured by implementing Elgamal encryption algorithm and the messages queuing is structured and organized. The response duration in completion of agent communication must be within 5 second.
b. Agent Verification module must ensure that all agents in system are authorized agent by implementing SHA1 with Elgamal digital signature Algorithm and Pattern Matching technique and must run accurately at decided time periodically. The response duration in completion of agent verification must be within 10 second.

3.3.1.3 Detection

a. Information filter module must gather all information needed from information sources to be analyzed in Data Analysis.
b. Data Analysis module will analyzes information with signature analysis technique gathered by Information in real-time basis.

3.3.1.4 Monitor

a. Administrator module must provide a Graphical User Interface (GUI) for administrator for controlling agents and providing reports through log files. In controlling agents, administrator is able to start, pause, resume and stop an agent in the system.
b. Log files module for detection, communication and verification must provide records needed information for administrator.
c. Detection Response module must react to intrusion detection constantly and effectively.
d. Verification Response module must react to faulty verification which mean if an unauthorized agent is detected constantly and effectively.

3.3.2. Non-Functional Requirements

Non-functional requirements for this system are concern with how well the system performs in suits with the architecture design in overcoming the issues and problems mentioned before. This would include the connection form, system structure and system topology. The non-functional requirements are described as below:

3.3.2.1 P2P Connection

This system has to be designed in P2P connection form to avoid architecture issues which related to single point of failure problems, consistency and duplication of information and delay on information sending. It would overcome multilevel authorization problem from hierarchical structure too.

3.3.2.2 Fully Distributed and Agent-Based System

SAAIDS is a distributed agent-based system. Agent has to be deployed distributable to enhance the scale of network area coverage. Monitoring and detection is done using an agent-based approach, where response decisions are made at the point of analysis.

3.3.2.3 Autonomous System

This system must run autonomously without human intervention in all tasks especially in doing intrusion detection, agent communication and agent verification process including detection and verification responses.

3.3.3. Use Case Diagram

UML consists several graphic elements which are combined to be a diagram. One of diagrams existing in UML is Use Case Diagram. Use Case Diagram used in mapping system behavior which means how the developing system functions and responses to its environment. Use Case model only describes what is system do without describes how the system doing it. Figure 3.2 shows Administrator Use Case Diagram and is followed by Use Case Description in Table 3.2 -3.4.

3.3.3.1 Administrator Use Case Diagram

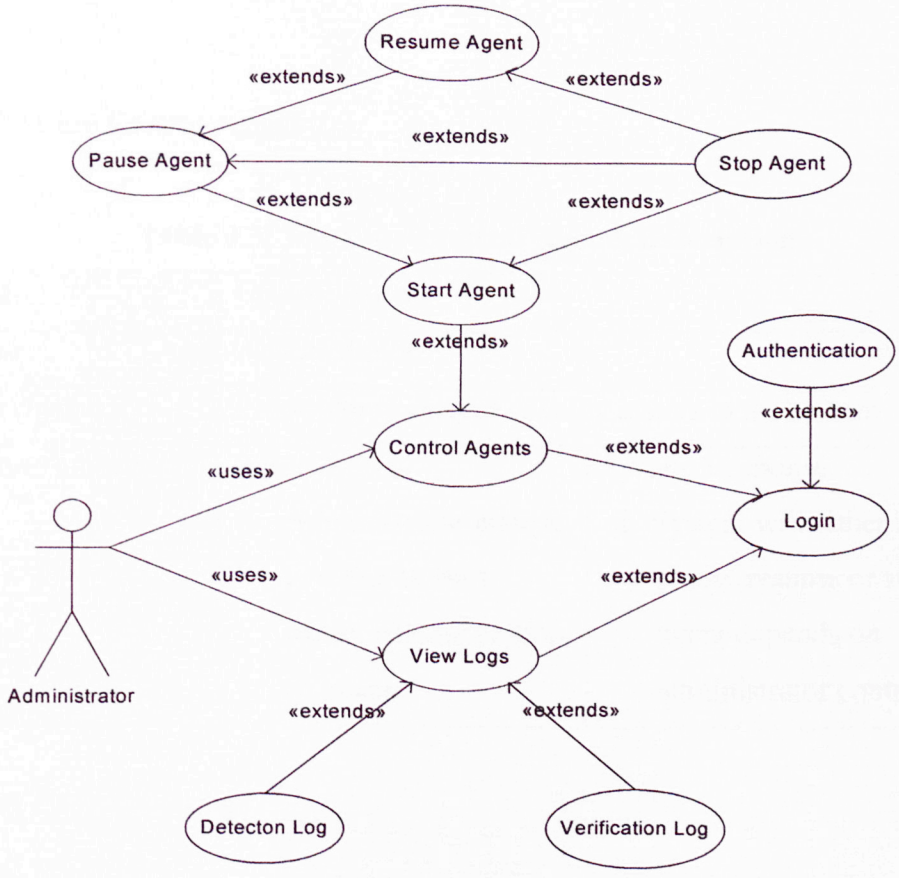

Figure 3.2: Administrator Use Case Diagram

a. Use Case Login

Table 3.2: Use Case Login Description

Name	Login	
Actor	Administrator	
Description	Procedure to log into the system	
Course of Events	Actor Action i. Administrator enters username and password.	System Response ii. System verifies username and password with Authentication module. iii. If the password correct, administrator logged into the

		system. If not, request for correct username and password.

b. Use Case Control Agents

Table 3.3: Use Case Control Agents Description

Name	Control Agents	
Actor	Administrator	
Description	Procedure to control existing agents in the system	
Course of Events	Actor Action i. Administrator choose to either to start, pause, resume or stop an agent.	System Response ii. System will either start, pause, resume or stop the agent depends on administrator controls.

c. Use Case View Logs

Table 3.4: Use Case View Logs Description

Name	View Logs	
Actor	Administrator	
Description	Procedure to view logs	
Course of Events	Actor Action i. Administrator chooses to view Detection Logs or Verification Logs.	System Response ii. System will either view Detection Logs or Verification Logs depends on administrator chooses.

3.3.4. Activity Diagram

Activity diagram is used to show events flow when activities occurred. In this system, activity diagram involves activities based on every functional requirements represents

four main components in the system stated before which are Administrator Activity Diagram, Intrusion Detection Activity Diagram, Agent Communication Activity Diagram and Agent Verification Activity Diagram. These entire activity diagrams are shown in Figure 3.3 – 3.6 in next pages.

3.3.4.1 Administrator Activity Diagram

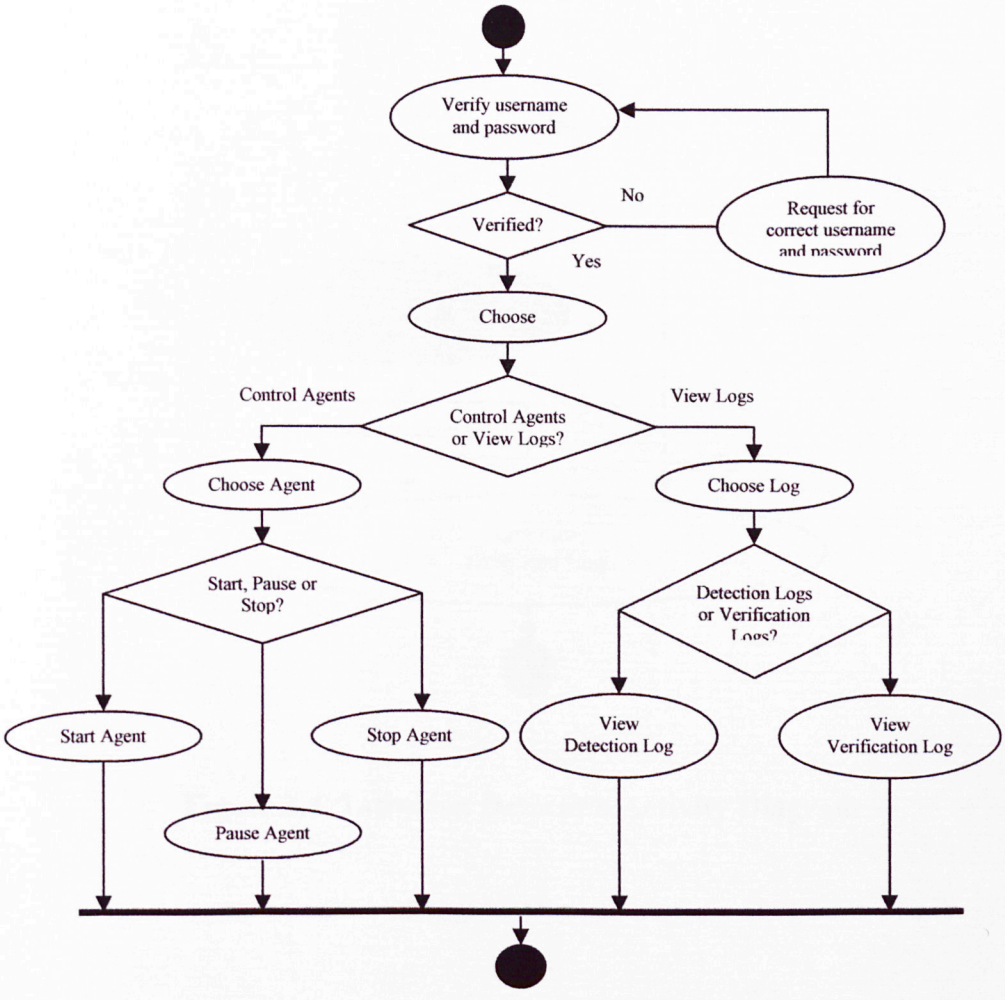

Figure 3.3: Administrator Activity Diagram

3.3.4.2 Intrusion Detection Activity Diagram

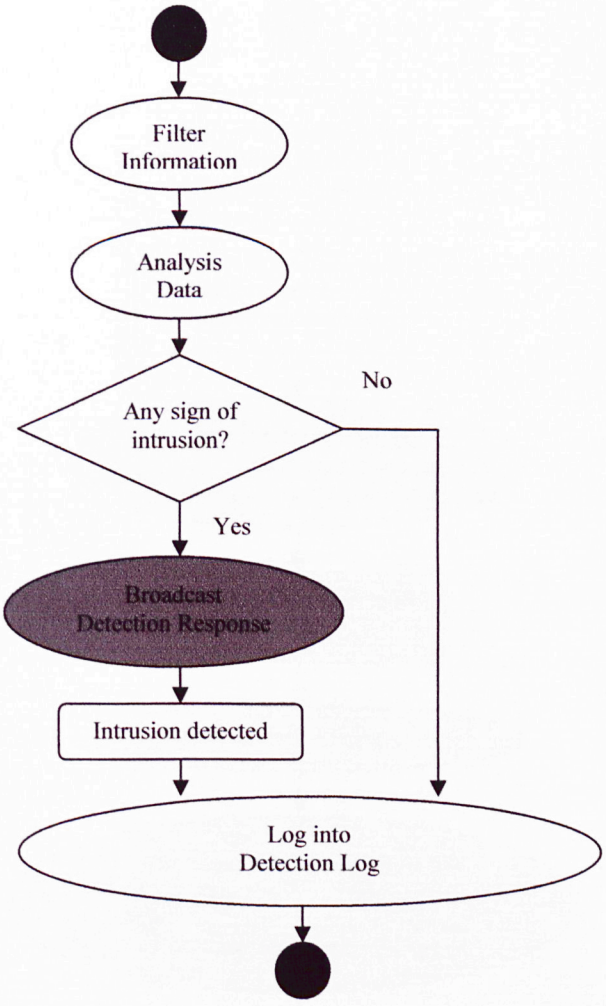

Figure 3.4: Intrusion Detection Activity Diagram

3.3.4.3 Agent Communication Activity Diagram

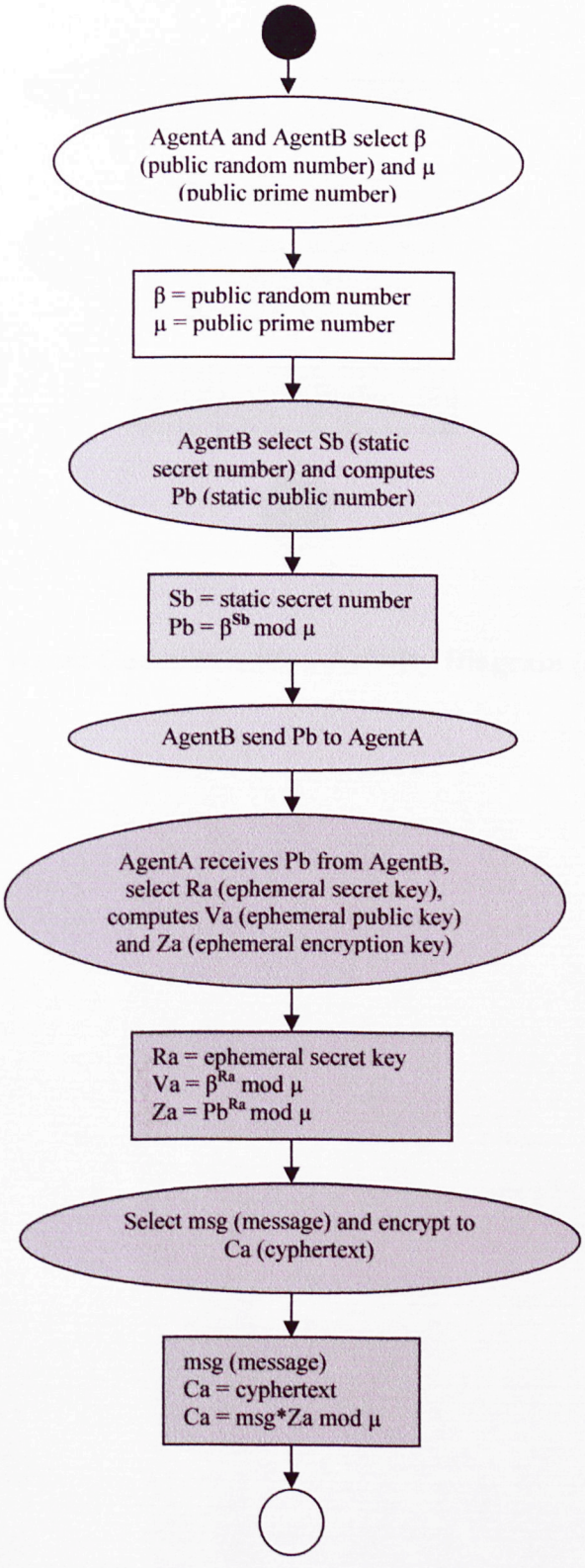

Figure 3.5: Agent Communication Activity Diagram (continues)

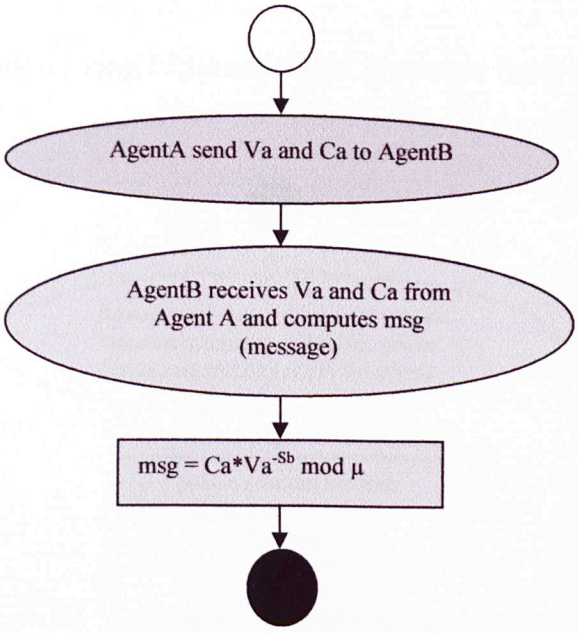

Figure 3.5: Agent Communication Activity Diagram (continued)

3.3.4.4 Agent Verification Activity Diagram

a. Agent Verification Using Elgamal Digital Signature Activity Diagram

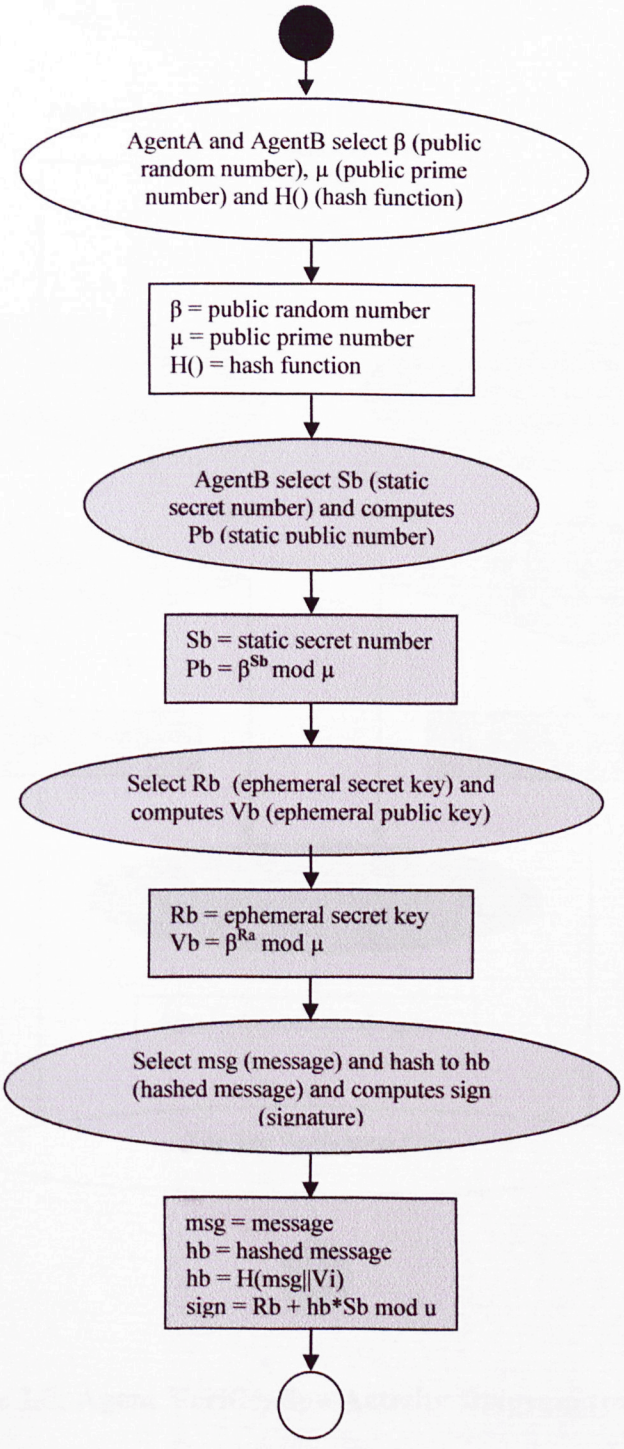

Figure 3.6: Agent Verification Activity Diagram (continues)

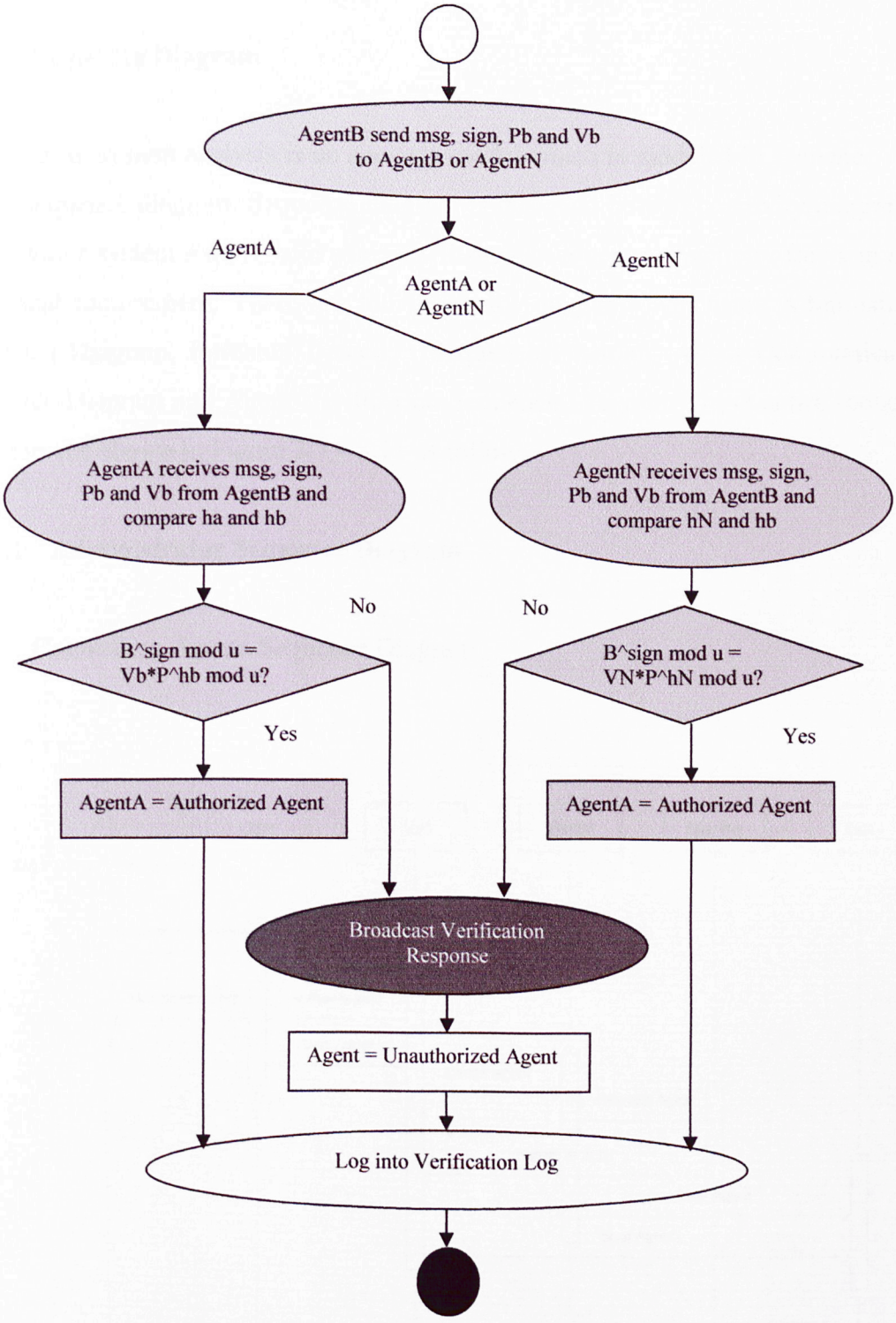

Figure 3.6: Agent Verification Activity Diagram (continued)

3.4. System Analysis

3.4.1. Sequence Diagram

Output from system analysis is an analysis model which is modeled in sequence view called sequence diagram. Sequence diagram is designed based on activity diagram in the previous system requirement analysis. It shows sequence of activity flows in each functional requirement. There are four sequence diagrams which are Administrator Sequence Diagram, Intrusion Detection Sequence Diagram, Agent Communication Sequence Diagram and Agent Verification Sequence Diagram. These entire sequence diagrams are shown in Figure 3.7 – 3.11 as follow:

3.4.1.1 Administrator Sequence Diagram

a. Controlling Agents Sequence Diagram

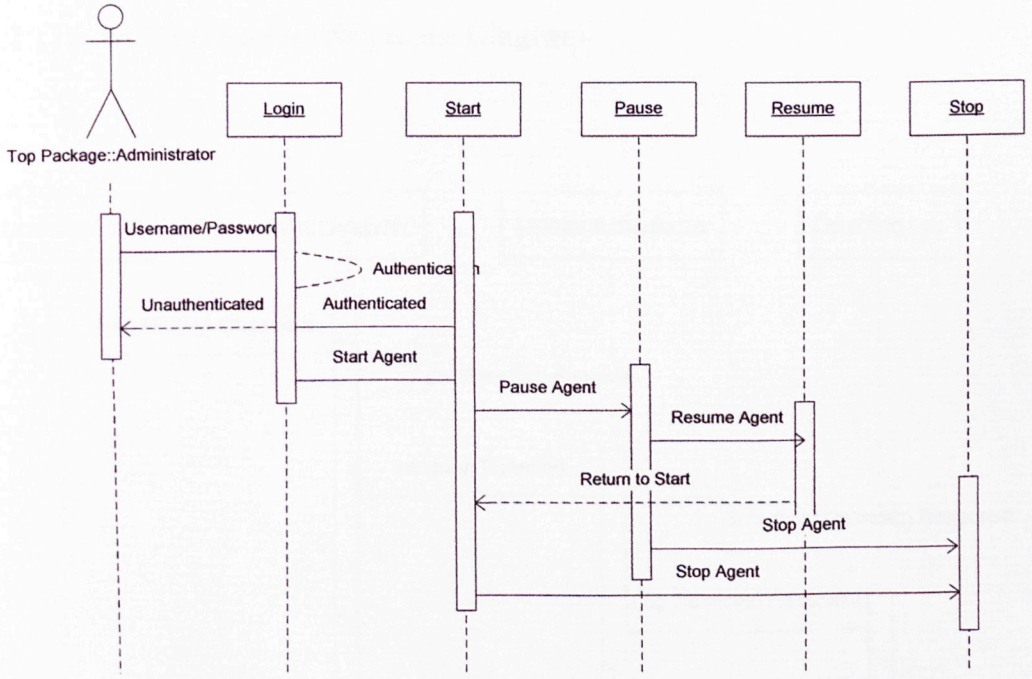

Figure 3.7: Controlling Agents Sequence Diagram

b. Viewing Logs Agents Sequence Diagram

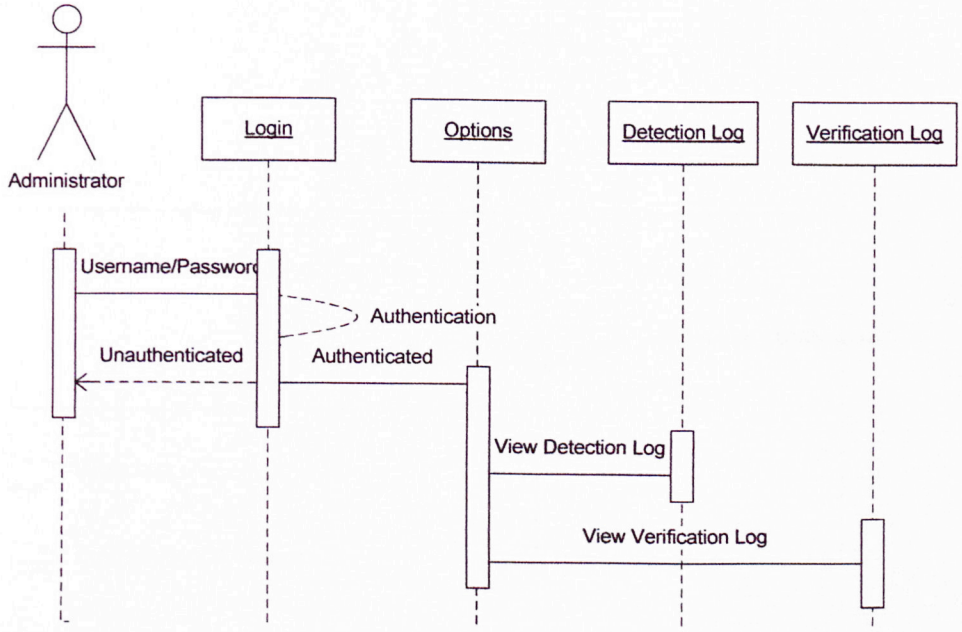

Figure 3.8: Viewing Logs Sequence Diagram

3.4.1.2 Intrusion Detection Sequence Diagram

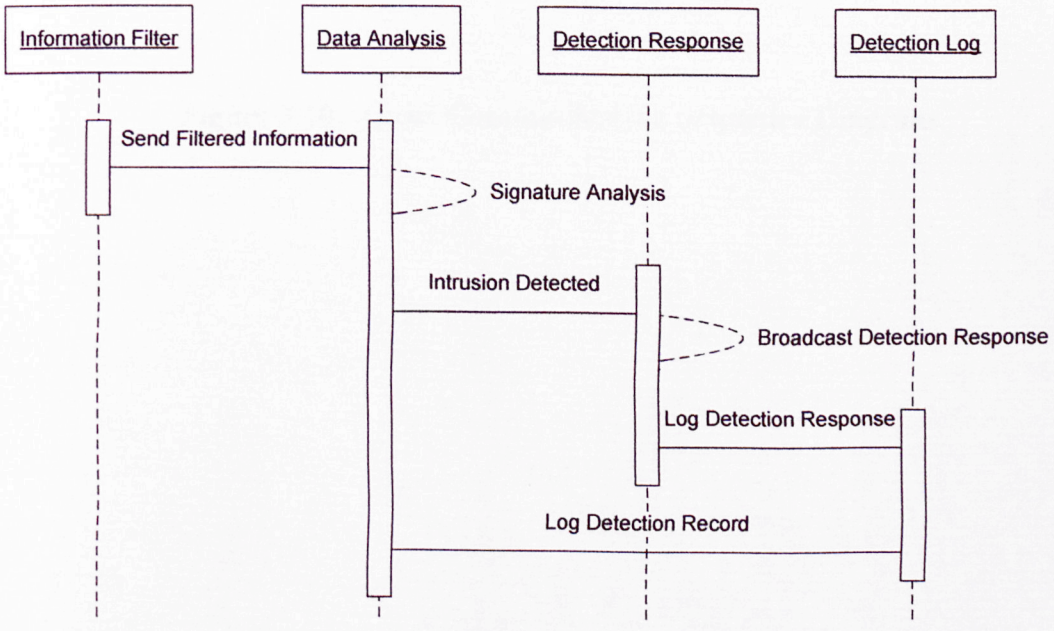

Figure 3.9: Intrusion Detection Sequence Diagram

3.4.1.3 Agent Communication Sequence Diagram

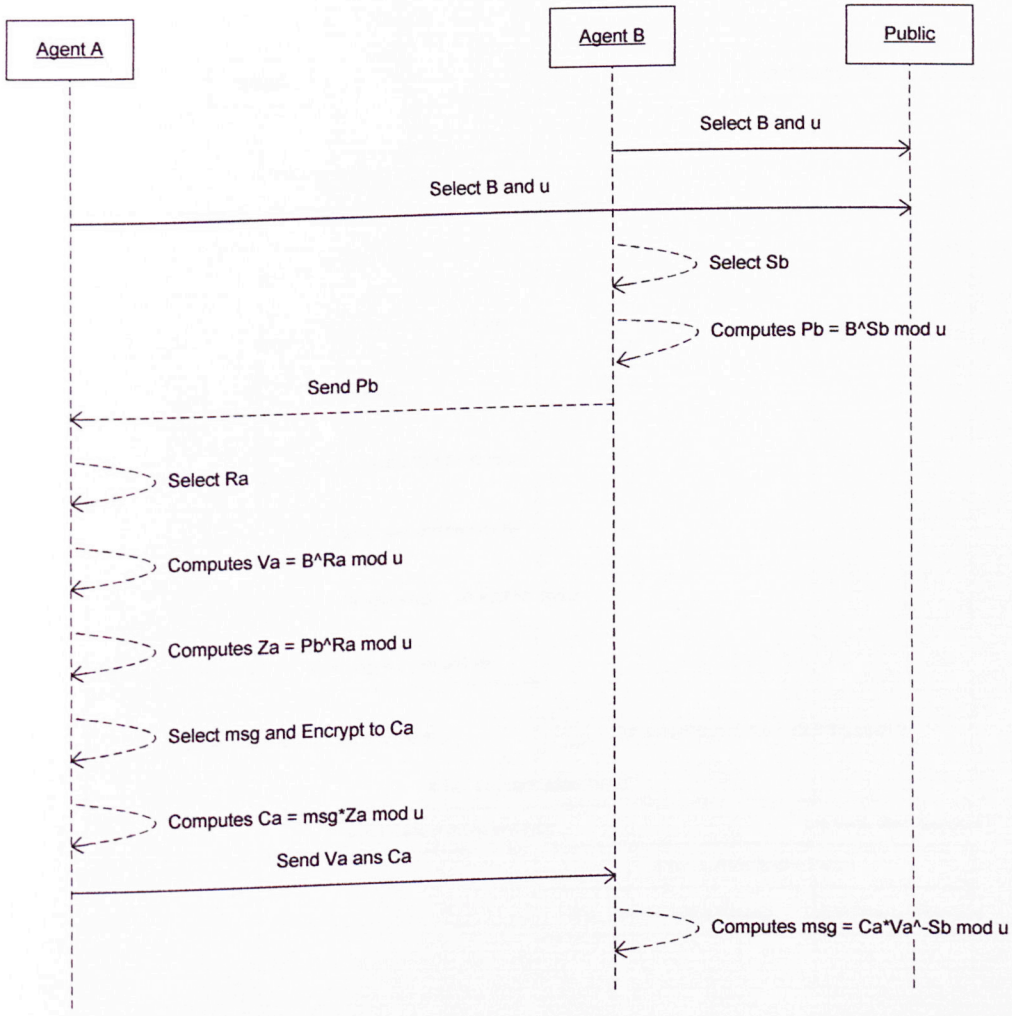

Figure 3.10: Agent Communication Sequence Diagram

3.4.1.3 Agent Communication Sequence Diagram

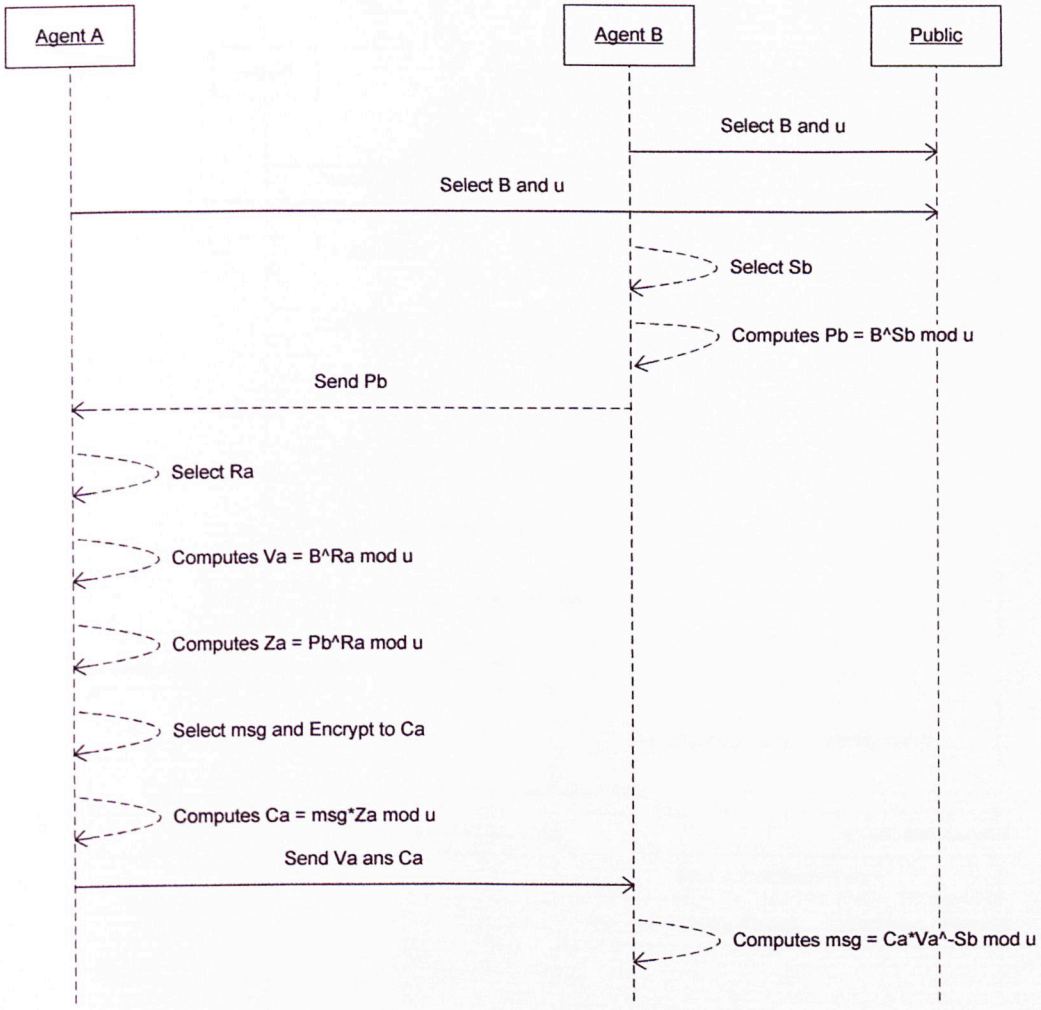

Figure 3.10: Agent Communication Sequence Diagram

3.4.1.4 Agent Verification Sequence Diagram

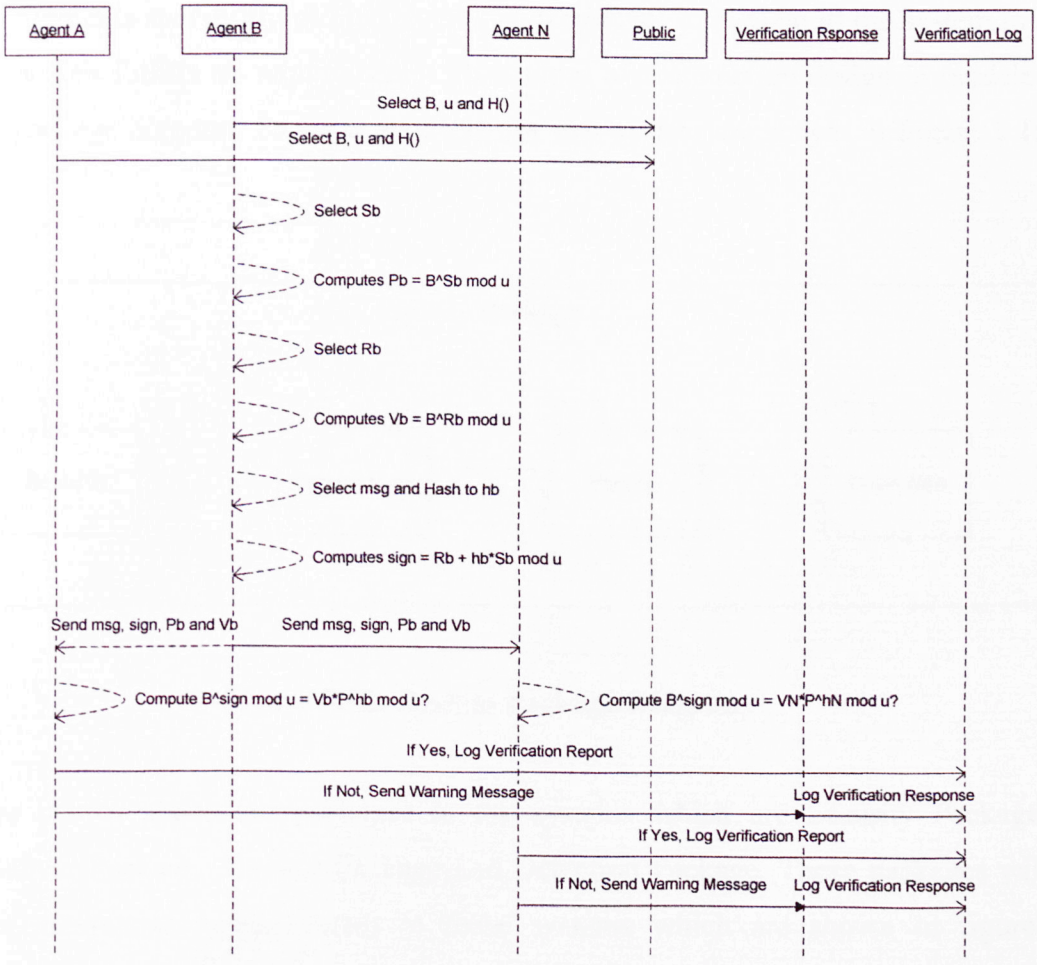

Figure 3.11: Agent Verification Sequence Diagram

3.5. System Design

When system requirement analysis and system analysis is completed, system design will take place. This phase is important where all informal ideas are analyzed and specified will be designed to be useful detail information in system implementation. Complete design specification used to be a mass communication between developer and system because it will cover all important information for system development.

3.5.1. Package Diagram

Architecture design intends to ease design process for subsystems and modules based on requirements specification. This technique defines all subsystem in the system to a design which fulfills all requirements. The output of architecture design is modeled into a package diagram. Package diagrams for this system are shown in Figure 3.12 below:

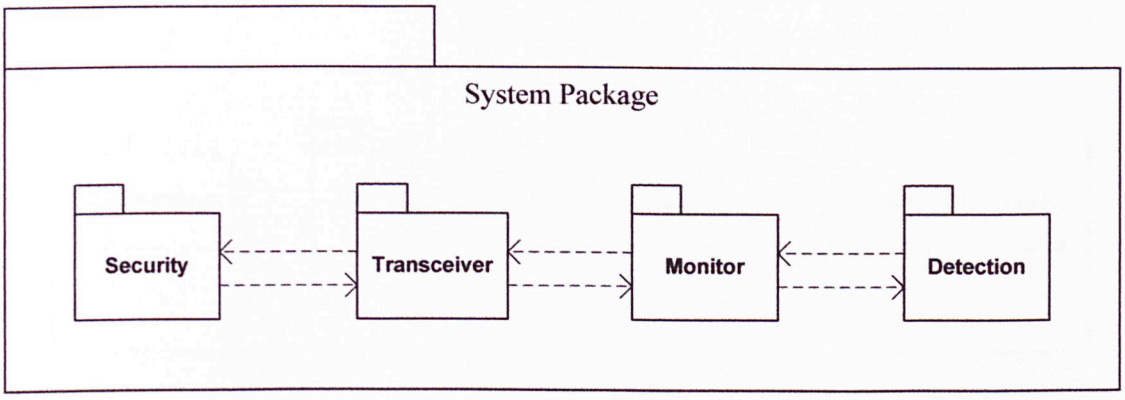

Figure 3.12: System Package Diagram

Figures shown above are packages in the systems which are Security Package, Transceiver Package, Monitor Package and Detection Package. These packages will design subsystems (subpackages) in these systems which are shown in figures follows:

3.5.1.1 Detection Package

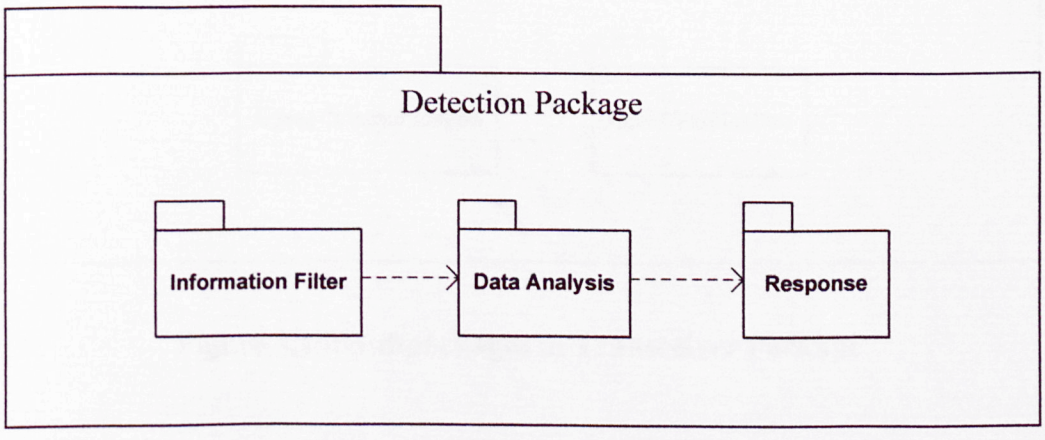

Figure 3.13: Subpackages in Detection Package

Figure 3.13 shows Detection Package diagram.

3.5.1.2 Monitor Package

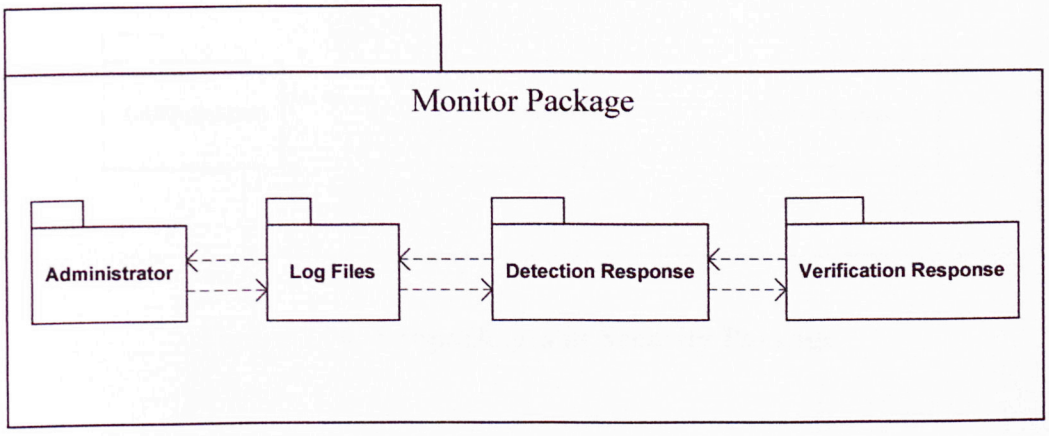

Figure 3.14: Subpackages in Monitor Package

Figure 3.14 shows Monitor Package diagram.

3.5.1.3 Transceiver Package

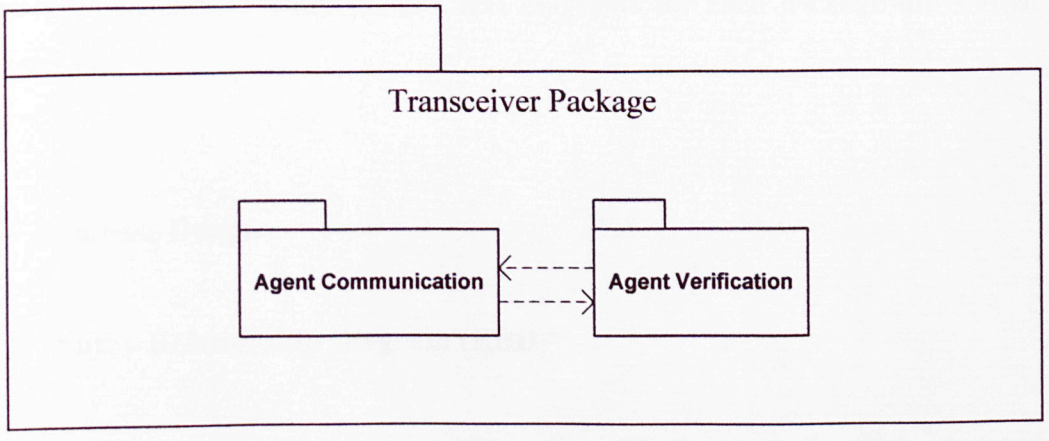

Figure 3.15: Subpackages in Transceiver Package

Figure 3.15 shows Transceiver Package diagram.

3.5.1.4 Security Package

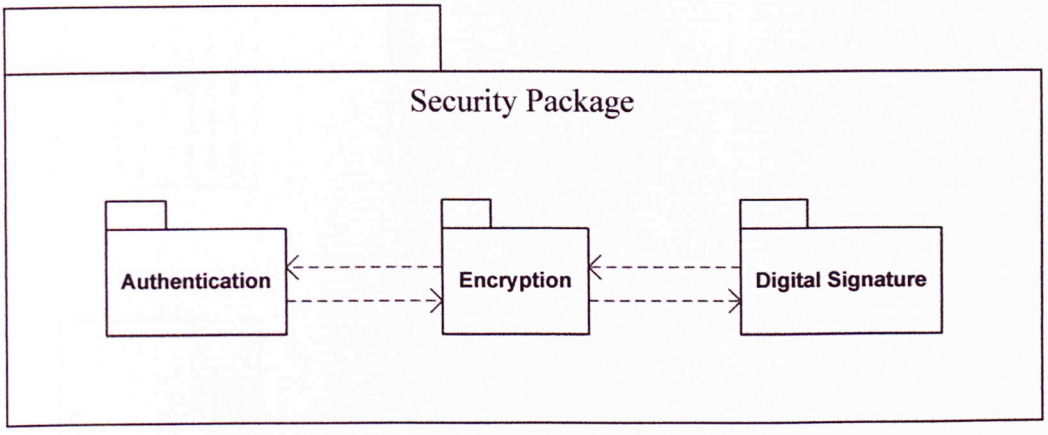

Figure 3.16: Subpackages in Security Package

Figure 3.26 shows Security Package diagram.

3.5.2. Class Diagram

Based on package diagram, class diagrams are designed to ease software program development process by dividing into classes. A class may represent a function in functional requirement or more and several classes also may represent only one function in functional requirement. Class diagrams for each package are shown in Figure 3.17.

3.6. Database Design

3.6.1. Entity Relationship Diagram (ERD)

The Entity Relationship diagram provides a pictorial representation of the major data objects, entities and relationships between them. The following is the table of each entity available in the system represents the data dictionary of the database system. Primary and foreign keys are the most basic components on which relational theory is

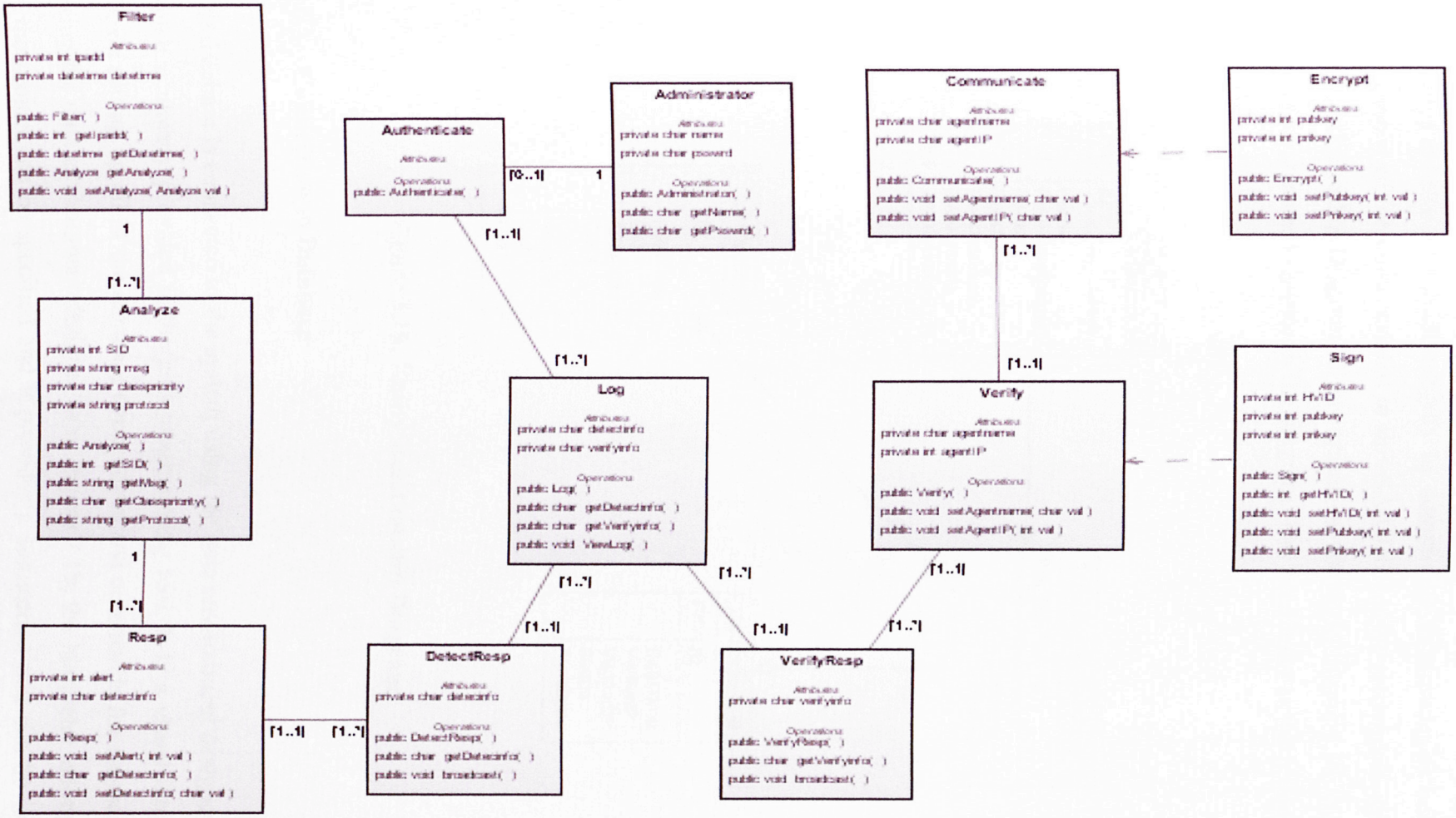

Figure 3.17: SAAIDS Class Diagram

based. Primary keys enforce entity integrity by uniquely identifying entity instances. Foreign keys enforce referential integrity by completing an association between two entities. Details of the tables existed in this SAAIDS can be referred at Appendix E. An Entity Relationship Diagram of this system as shown in Figure 3.18. It represents the relationship between entities in the diagram. The bold letter is the primary key for each entity.

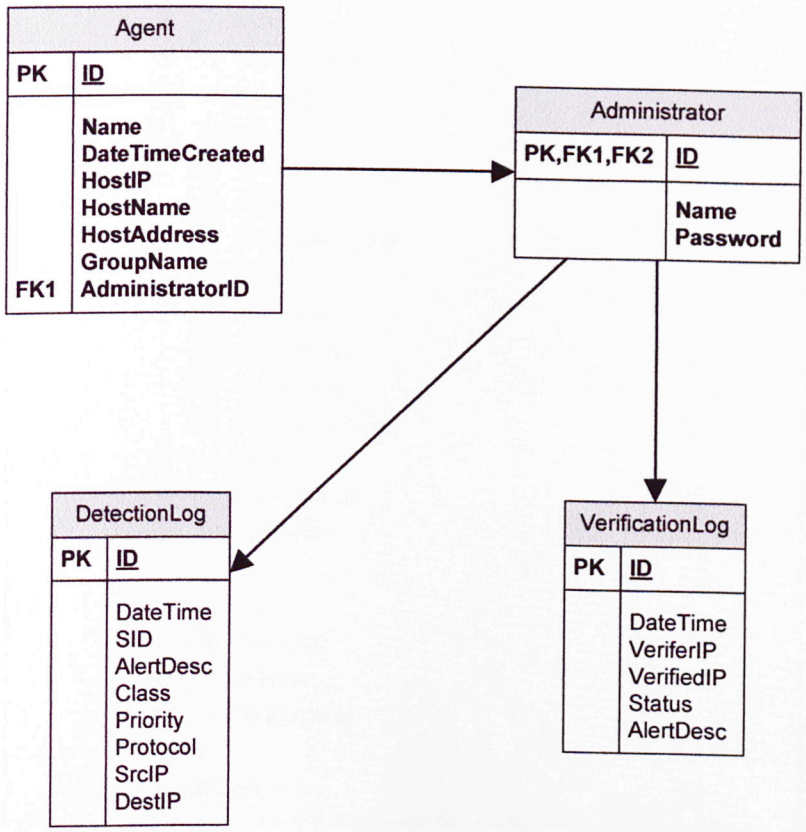

Figure 3.18: Entity Relationship Diagram

3.6.2. Connection to Database

The database is connected to the system using the data environment designer facilities which is already provided in the java programming tool. It is a visual interface that allows setting up all of the database connections and commands for database access. Using the data environment designer as in Figure 3.19, the hierarchy among the data in the project can be specified and it provides a connection that can be used by all forms in the project.

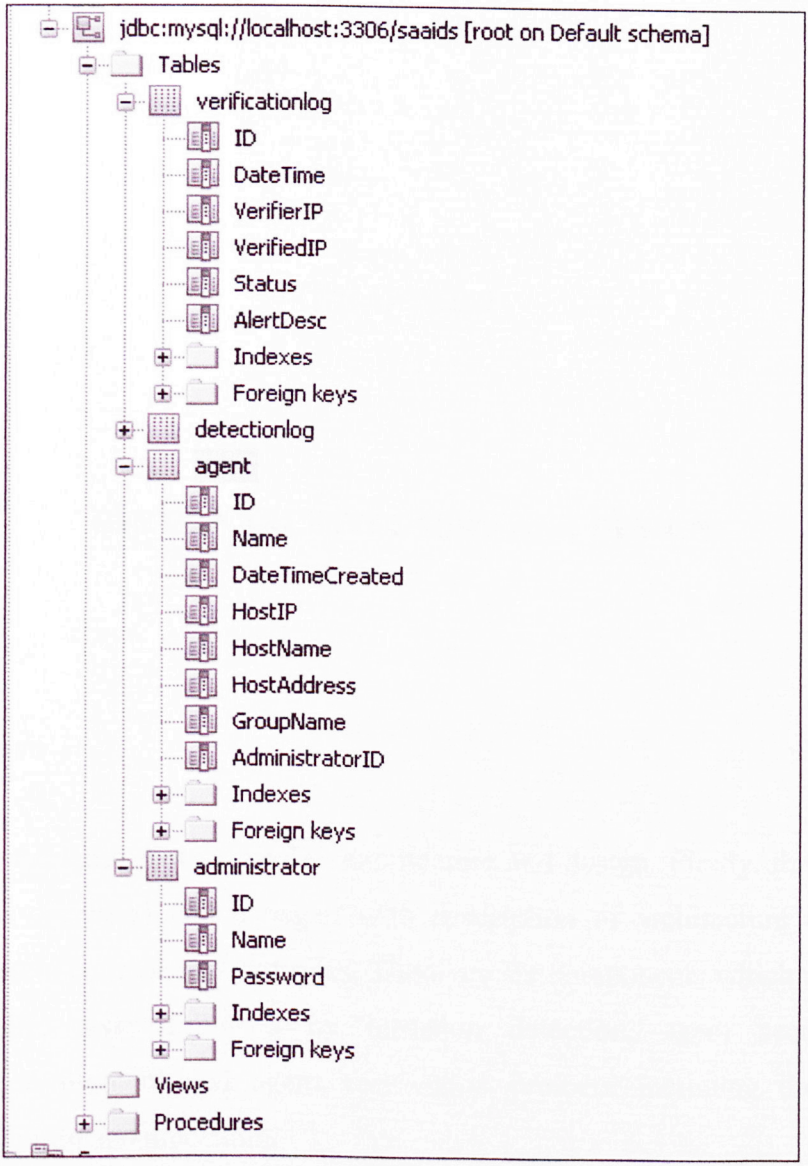

Figure 3.19: Data Environment Designer