SECURITY MANAGEMENT AGAINST CLONING MOBILE PHONES

Mirela Sechi Moretti Annoni Notare[1] *Azzedine Boukerche*[2] *Fernando A. S. Cruz*[1] *Bernardo G. Riso*[1] *Carlos B. Westphall*[1]

1 Network and Management Laboratory
Federal University of Santa Catarina (UFSC)
{mirela, cruz, riso, westphal}@lrg.ufsc.br

2 Department of Computer Sciences
University of North Texas
boukerche@silo.csci.unt.edu

***Abstract**: This work presents the development of a distributed security management system for telecommunication networks. The system consists in reducing the use of cloned mobile telephones using three main techniques: (1) An ISO Formal Technique (LOTOS) is used to specify and validate the system; (2) A Pattern Recognition Technique is used to classify the telephone users into classes in order to identify if a call does not correspond to the patterns of a specific user; and (3) Distributed Object Technique is used for the implementation of this distributed system (i.e., manager and agents).*

Keywords: Distributed Management, Telecommunication Security, Formal Description Technique, Pattern Recognition, CORBA.

1. INTRODUCTION

The security management service is responsible for providing a safe environment for both the operation and management of resources in a domain [14, 15]. Safety and Security are two reliability properties of a system. A ‘safe’ system provides protection against errors of trusted users, while ‘secure’ system protects against errors introduced by untrusted users [1]. A comprehensive network security plan must encompass all the elements that make up the network and provide important services: Access (authorized users), Confidentiality, (information remains private), Authentication (sender is who he claims to be), Integrity (message has not been modified in transit) and Nonrepudiation (originator cannot deny that he sent the message) [4].

Our main objective is to augment the security in telecommunication networks, avoiding frauds of cloned mobile phones. In order to program a non genuine mobile cloned phone,for instance to debit calls from a genuine mobile phone, one only needs to buy a piece of portable radio equipment called a scanner, which registers the frequency in which mobile phones operate in its immediate surroundings. The person committing the fraud may, for example, park his car around a shopping center, jot down various frequencies, transfer the data to clones and then pass them on to whomever may be interested [7].

The present work makes use of formal description techniques to specify, validate (employing simulations, testing and verifications) and translate from specification code to implementation code. The specification is made in stepped refinements, using automatic tools to verify each refinement. LOTOS (Language of Temporal Ordering Specification) is the formal description technique (FTD) used by the Eucalyptus Toolbox employment. In addition, pattern recognition techniques are used to classify the telephone users into classes according to their usage logs. Such logs contain the relevant characteristics for every call made by the user. From this classification it is easier to identify if a call does not correspond to the patterns of a specific user, and thus, identify whether the call was effected by a non-genuine caller. As a consequence, the immediate identification of a fraud (as opposed to the moment of receiving the monthly bill) will reduce losses for both users and carriers. We are convinced that the distributed systems which make use of this classified database can uncover frauds with greater ease than conventional systems, when a call is outside of the pattern of a particular user, that is, when a possible fraud occurs. Pattern Recognition techniques are used by the MatLab tool employment. With this software, neural network algorithms (such as k-means, p-nearest neighbour and gauss) are implemented. Moreover, due to the characteristics of the telecommunication networks – distributed and heterogeneous – our system uses the CORBA architecture. The ODP/OMG CORBA (Open Distributed Processing/Object Management Group Common Object Request Broker Architecture) is a management technology for distributed objects. It provides a basic structure for distribution and has demonstrated its ability for the support of important functions required in telecommunication and services management networks. CORBA is totally object oriented and foresees independence and portability of protocols due to the APIs.

This paper is organized as follows. In Section 2, the formal description technique employment for specification and validation is presented. In Section 3, the pattern recognition technique usage for users classification is shown. In section 4, some relevant aspects of C++ and Java languages with CORBA support used for implementation are described. Finally, The conclusion follows.

2. USING FORMAL DESCRIPTION TECHNIQUES

Our system uses the LOTOS FDT [2], an ISO and actual standard which can describe both abstract data types and behaviour, to enhance rigour in the procedures and obtain specification, validation (simulations, testing, verifications) and automatic translation from LOTOS code to C code.

In order to validate our Security System Against Cellular Cloning, which we refer to as SSCC system, we make use of the CADP tool (Caesar Aldébaran Development Package) [6] available within the Eucalyptus toolbox. The procedure used to obtain the correction proofs between refinements generates the following two automata: `SSCC.AUT` and `SSCC_DET.AUT`. These two automata aim at proving correctness of the system in conform with ISO 8807 [2] and US DoD ClassA1 [14]. In our work, attention was given to the behaviour aspects but did not include the abstract data types description. This position is justified due the great number of works related with data types (using ASN.1/GDMO – Abstract Syntax Notation One/Guidelines for the Definition of Managed Objects, for instance) and the lack of works related with the behaviour formal description.

2.1 The SSCC Most Abstract Specification

Initially, in the highest abstraction level, the SSCC system can be observed as a black box, with two communication gates (gate `mail` and gate `phone`), to send messages to the users. The gate `mail` is used by the SSCC to send alarms of possible frauds to the user by surface mail. The gate `phone` allows the SSCC to use the mobile phone to send the same alarm. The specific advantage of sending alarms by phone is the immediate notification, the specific advantage of mail alarm is security.

The SSCC system is always active, designating an infinite range of behaviour, and suggests a LOTOS specification with the `noexit` functionality:

```
specification SSCC[mail,phone]:noexit
behaviour SSCC[mail,phone] where
 process SSCC[mail,phone]:noexit:=
  mail;(phone;SSCC[mail,phone] [] i;SSCC[mail,phone])
 endproc
endspec
```

The behaviour of the SSCC system is defined by the process `SSCC`, that can execute an action in the gate `mail`, to send an alarm by surface mail (we consider this action always possible); the sequence is followed by a non deterministic choice with two options. The first option is related to the alarm sent by mobile phone, in the gate `phone` (this action is not always possible) and, following, the process `SSCC` is called recursively in order to treat another case. Because the first option may not be successful - phones do not work properly, out of their area, for instance, after a period of time, an internal action `i` occurs (not observed) and the process `SSCC` is called recursively. The most abstract specification of the `SSCC` system corresponds to a formalization of the user requirements of this system. It is the basis for future refinements of the project.

2.2 SSCC Specification Refinement

The `SSCC` system can be detailed in order to consider two of its most important components: the Manager (represented by the `MANAGER` process) and the Managed Sites Set (represented by the `SITES_SET` process). This refined conception is identified by `SSCC_REF`. The process `SITES_SET` uses the gate `notif` to send notifications to the `MANAGER` process. This, in turn, after receiving a notification (by the gate `notif`) sends alarms to the users (by the gates `phone` and `mail`). `SSCC_REF` identifies the LOTOS specification of this refined conception.

```
SSCC_REF[mail,phone]:noexit
behaviour SSCC_REF[mail,phone]: where
process SSCC_REF[mail,phone]:noexit:=
         hide notif in  SITES_SET[notif]
         |[notif]|   MANAGER[notif,mail,phone]  where
         process SITES_SET[notif]:noexit:=              endproc
         process MANAGER[notif,mail,phone]:noexit:=      endproc
endproc
endspec
```

The behaviour of the `SITES_SET` process can be specified in LOTOS as follows: `notif;SITES_SET[notif].` In this manner, an infinite succession of notifications can be made.

The `MANAGER` process can be specified in LOTOS as follows: `notif;mail;(phone;MANAGER[notif,mail,phone][]i;MANAGER[notif,mail,phone])` And then, it sends alarms to the users after receiving notification. The `SITES_SET` e `MANAGER` processes are combined with the general composition operator (`|[…]|`) usage. In this combination it is shown that both processes share all events that occur in the `notif` gate. The `hide…in` operator usage hides the `notif` internal gate, allowing us to compare the `SSCC` specification with the `SSCC_REF` specification; proving (by observational equivalence) that the last one is a correct refinement of the first.

2.3 Refinement of the SITES_SET Process

The `SITES_SET` process (Managed Sites Set) includes several instances of the same managed site model. Each of these instances corresponds to a LOTOS process that communicates with a `MANAGER` process (System Manager) through the `notif` gate. Consider that each managed site acts alone in sending the possible frauds alarms to the Manager, we can then use the independent composition operator (`|||`) to combine them, obtaining the following LOTOS representation: `SITE_1[notif]  |||  SITE_2[notif]  |||  . . .  |||  SITE_N[notif]` Each one of these managed sites constitutes a distributed agent. For obvious reasons, large cities need more agent sites than small cities.

2.4 Detail of a Managed Site

The adopted model for the managed sites conception considers them as three main elements: a Management Agent, a Reference Baseline and a File with the Telephone Calls. The `SITE_J` process represents a typical management site, with its three main elements. The LOTOS formal specification of this architecture can be presented as follows:

```
process SITE_J[notif]:noexit:=  hide base_j,file_j in
      (BASELINE_J[base_j]  |||  CALLS_FILE_J[file_j])
      |[base_j,file_j[|
      AGENT_J[base_j,file_j,notif]where...
endproc
```

The `hide … in` operator employed in the detailed specification of the `SITE_J` process allows us to compare this specification with another more abstract, of the same site, that does not employ this operator. The `BASELINE_J` and `CALLS_FILE_J` processes act independently. Considering them in a set, these two processes share events, in the gates `base_j` and `file_j`, with the `AGENT_J` process. The `BASELINE_J` process can be run in an infinite sequence of events on its gate `base_j: base_j; BASELINE[base_j]` In a similar mode, the `CALLS_FILE_J` process can perform more actions on the `file_j` gate: `file_j; CALLS_FILE[file_j]` The `BASELINE_J` and `CALLS_FILE` processes do not have complex behaviours. However, the `AGENT_J` process, can demonstrate more complex behaviour: after checking the `CALLS_FILE` (through an action in the `file_j` gate), the `AGENT_J` verifies the certified user characteristics through access to the

BASELINE_J (through an event in the file_j gate). Then, a non-deterministic choice with two options appears: this non deterministic choice is internally solved by the AGENT_J process:

```
file_j;base_j;(i;AGENT_J[base_j,file_j,notif]     []
i;notif;AGENT_J[base_j,file_j,notif])
```

In the first option, the internal event i indicates that nothing abnormal has been detected. In that case, the AGENT_J process is called recursively. The second option represents the AGENT_J behaviour when there is possibility of fraud.

2.5 SSCC Complete Refinement

In order to consider a simple case, we present, as an illustration of the SSCC completed refined, a case in which the managed site set includes only two sites (i.e., SITE_1 and SITE_2). See the Figure 2.1.

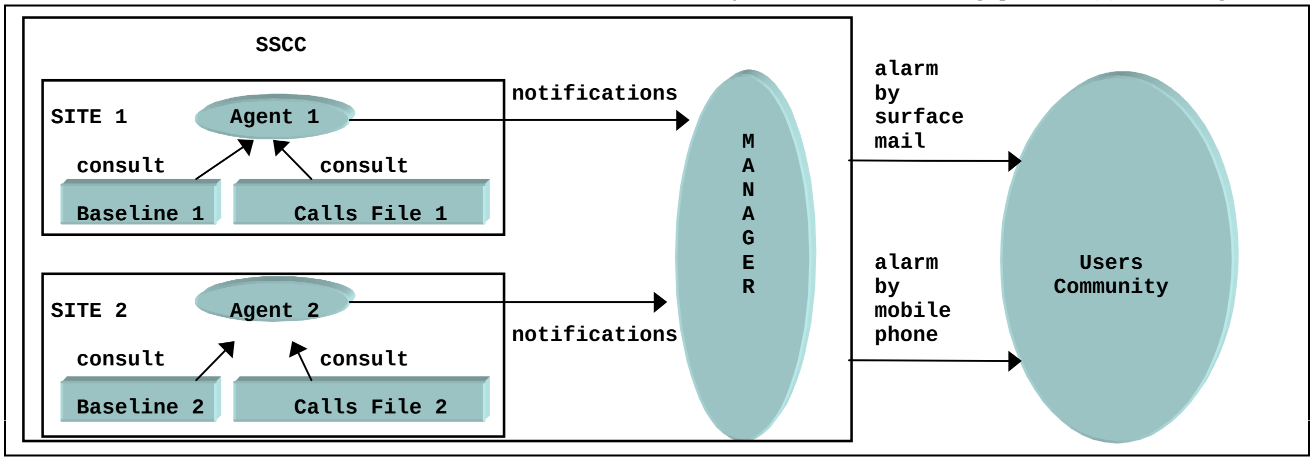


Figure 2.1 - SSCC representation with two managed sites.

Presently, this specification is being validated and translated from LOTOS to C code, using the Eucalyptus Toolbox (see Section 2.6 hereafter). This C code will be helpful for the C and Java implementation presented in the Section 4.

2.6 Results of the Validation Using Eucalyptus Toolbox

The Eucalyptus toolbox [6] is a graphical user-interface (GUI) based on X-Windows. Although in the Eucalyptus toolbox the tools were developed by different partners, extensive efforts have been made to achieve a smooth integration, by making all the tools compatible with each other, by developing gateways that allow different tools to interoperate, and by providing a unified user-interface. The procedure used to obtain the correction proofs between refinements, i.e., specification from SSCC to SSCC_DET uses the CADP tool (Caesar Aldébaran Development Package) [6] included in the Eucalyptus toolbox. This procedure generates two automata: SSCC.AUT and SSCC_DET.AUT. Using the observational equivalence option, the ´TRUE´ result is achieved.

The ´TRUE´ result states that the processes under consideration are observational equivalent, i.e. the refinement is proved correct, proving that each refinement made is equivalent to the previous specification. We consider the validation process composed by simulations, test and verifications. Both, simulations (were made exhaustive and interactive simulations) and testing (find events sequences, for instance) are able to found and identify errors – but they do not prove correctness. On the other hand, verifications provides us with the correctness proof.

3. USING PATTERN RECOGNITION

This section, we discuss the outline of the algorithm, a description of its implementation., and the results obtained.

3.1 Outline of the Algorithm

Neural network (when seen as an adaptive machine) can be defined as a processor distributed massively in parallel and which has the natural propensity to store experimental knowledge and make it available for use. It is similar to the mind in two aspects: (1) Knowledge is acquired by the network by means of the learning process. (2) The weights of the connections between neurons, known as synapses, are used to store the knowledge. The procedure used to represent the learning process, commonly called learning algorithm, has the function of modifying the weights of the connections of the network in seeking to reach an initial designed objective. Neural networks are also referred to in literature as neuro-computers, artificial neural networks and parallel distributed processors, for instance.

The construction of an RBF (Radial Basis Function) in its more basic form involves three layers, whose output nodes form a linear combination of the Radial Basis Function (kernel) calculated by the nodes of the hidden layer. The Radial Basis Function in the hidden layer produces a response for the input stimulus pattern), that is, it produces response different from zero only when the input pattern is within a small region located in the input space. The input is made from the source nodes (sensorial units). Each activation function requires a center and a numeric parameter. A function which can be used as activation is the Gauss function, while this network can be used to make decisions of maximum hood, determining which of the various centers is most similar to the input vector.

Another common variation in the Radial Basis Function is to increase its functionality using the Mahalanobis distance in the Gaussian function. The previous equation becomes:

$$f = (\mathbf{x} - \mathbf{c}) = \frac{1}{(2\pi)^{n/2}|K|^{1/2}} \exp\left\{-\tfrac{1}{2}(x-c)^T K^{-1}(x-c)\right\}$$

where K^{-1} is the inverse of the X co-variance matrix, associated with the node of the hidden C layer. Given n-vectors (input data) of p-samples, representing p-classes, the network may be initiated with the knowledge of the centers (locations of the samples). If the J-th vector sample is represented, then the weight matrix C can be defined as: $C = [c_1\ c_2\ ...c_3]^T$ so that the weights of the hidden layer in the j node are composed of the center vector. The output layer is a weight sum of the outputs of the hidden layer. When presenting an input vector for the network, the network implements

$$\mathrm{y} = \mathrm{W} \cdot f(\|\mathrm{x} - \mathrm{c}\|)$$

where *f* represents the functional output vector of the hidden layer, and C the corresponding center vector. After supplying some data with desired results, the weights W can be determined using LMS training algorithm interactively and non-interactively, as techniques of the descendant and pseudo inverse gradient, respectively. The learning in the intermediate (hidden) layer is executed using the non-supervised method, typically a cluster algorithm, a heuristic cluster algorithm, or an algorithm supervised to find the centers (C nodes in the hidden layer). The most common algorithm employed to determine the centers (which are the connections between the input layer and the intermediate layer) is the Lloyd or K-means algorithm. Some studies also have employed supervised learning to find the centers, and self-organised learning of the centers or the minimum Orthogonal Least Squares algorithm. A simple way of determining the σ^2 variation parameter for the Gaussian functions is to make them equal to the median distance between all the training data

$$\sigma_j^2 = \frac{1}{M_j} \sum_{x \in \Theta_j} (x-c)^T (x-c)$$

where Θ_j is the group of training patterns grouped in the center of the cluster Cj, and Mj is the number of patterns in Θ. Another way of choosing the parameter σ^2 is to calculate the distances between the centers in each dimension and use some percentage of this distance for the scale factor. In this manner, the p-nearest neighbour algorithm has been used. The reasons that have led us to study the application and use of the RNA approach for classification are: (1) an RNA has the intrinsic capacity of learning input data and to generalise; (2) the network is non-parametric and makes more delicate suppositions regarding the distribution of input data than the static traditional methods (Baysian); and (3) an RNA is capable of creating decision boundaries which are highly non-linear in the space of characteristics. Beyond this, these attributes are not unique for the RNAs used for classification.

In summary, in order to conduct the classification from the existing data base, an artificial neural network was used, built from a radial base function (Gaussian), known in literature as RBF with use of the clustering algorithm (k-means) that, for this work, was shown to be very efficient. The architecture of the radial basis function network consists of an entry layer, a hidden layer and an output layer. We believe that the algorithms used are efficient, though we are currently researching the implementation of possibly more efficient algorithms to improve the system.

3.2 Algorithm Implementation

The following, steps to reach a solution to the proposed problem are described.
Step 1: at the first level of connections of a radial base network, one must first of all identify the number of neurons of the hidden layer; Step 2: next the centers c_j are found (j=1,…..,M) which make-up the base of an M-dimensional space. Step 3: for each presented input pattern for, the $\| x_i - c_j \|$ is sent as a parameter to the radial basis (Gaussian) function which describes the level of classification of the input patterns. The output of the hidden layer makes-up a G matrix, which serves as a basis for the calculation of the weight ***W*** (connections for an output layer), following the formula: ***W***= $G^{□}$ ***t*** ***w***here $G^{□}$ is a pseudo-inverse of the G matrix given by: $G^{□}=(G^{7}$□□$G + \lambda\ G_0)^{□□}.G^{7}$ and ***t*** is the matrix which contains the group of training data. Step 4: the output is calculated as a sum of the activated neurons (excited neurons) of a hidden layer.

The K-means and P-nearest neighbour algorithms were used to obtain the centers and radiuses of each cluster and variance between the centers, respectively. The Gauss function was used for obtaining the output of a hidden layer (centers data, input patterns and radii) and a linear function (denoted *purelin*) contained in the neural Toolbox. This Toolbox is an addendum to the Matlab software [11], where various implemented functions are available for use in the neural networks project. Next the source code (`.m` files) is presented which executes the classification of users through the K-means, P-nearest neighbour and Gauss algorithms [5, 10].

3.3. Results of the Classification

The best classification using this pattern recognition technique was obtained using 110 neurons in the hidden layer, giving an error rate equal to 4,2027% (see Table I).

Table I - Number of neurons in the hidden layer and respective error rate.

Neurons of the hidden layer	error rate
50	5.0185
100	4.4252
110	4.2027
127	4.3511

This error rate is quite satisfactory. Previous algorithms showed a higher rate, e.g., using the same data, and Back Propagation algorithm, an error rate of 5,4 was obtained.

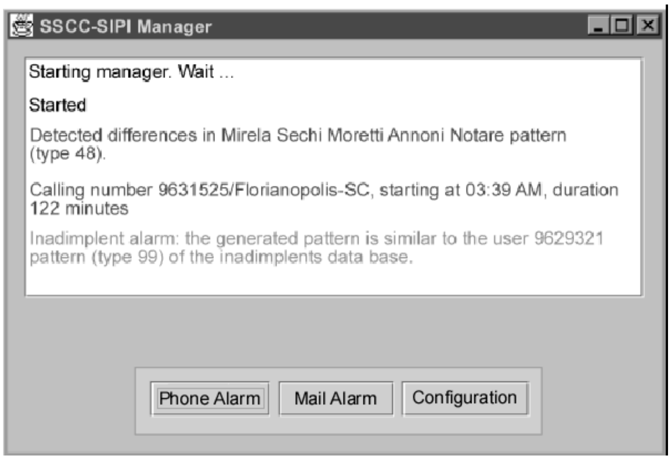
This classified data makes-up the database used in the implementation of the Mobile Phone Security Management System presented in Section 4. In the implementation, every call is compared with this database in order to identify a possible fraud, i.e., a call which does not match with that the pattern of the client.

4. IMPLEMENTATION OF THE SSCC

In our implementation, we have used Java Development Kit (JDK), and Visibroker 3.1. JDK includes the necessary tools to compile, refine and execute applets and applications written in

Java language. Visibroker 3.1 for Java is a software that integrates CORBA and Java technologies, allowing the implementation of client-server applications, written in Java. Java was used because it is simple object oriented language, architecture independent, portable, multi-task, dynamic, robust, secure and offers high-performance [3, 8, 9, 12, 13].

See in Figure 4.1 the system (manager) running.



Starting manager. Wait ...

Started

Detected differences in Mirela Sechi Moretti Annoni Notare pattern (type 48).

Calling number 9631525/Florianopolis-SC, starting at 03:39 AM, duration 122 minutes

Inadimplent alarm: the generated pattern is similar to the user 9629321 pattern (type 99) of the inadimplents data base.

Figure 4.1 – SSCC/SIPI running.

In the Figure 4.1 you can se the Manager receiving alarms from the Agents about possible frauds. Then the Manager will be able to detect if this altered pattern matches with an existent impostor pattern, and also, to send immediately a warning to the user, by phone and by mail.

5. CONCLUSIONS

At this time, we can say that the techniques employed were very useful. The LOTOS ISO standard helps with a rigorous validation process. The automata generation was especially important in revealing all possible execution paths. The C code generated from LOTOS code is being investigated, in order to catalyse the development. The CORBA support for the distribution of codes (both C and Java) have been very satisfactory. In other works, various techniques were employed for the development of this security system for telecommunication networks subject of this paper. The current work seeks to employ a classification algorithm of high reliability. The method used for the classification of the carrier clients, which includes the K-means, P-Nearest Neighbour and Gauss algorithms and the *purelin* function, proved to be efficient and reliable with the use of the MatLab software.

As a continuation of this study, the intention is to reduce the obtained error rates, by employing the Orthogonal Least Squares algorithm. This is a Gram-Schmidt orthogonalization process, which guarantees that each new column added to the design matrix of a growing subset is orthogonal to all previous columns. This simplifies the process of obtaining the sum-squared-error, which makes the algorithm more efficient. In addition, similarity investigations between the C code generated with CADP tool and C/Java code generated under a CORBA distribution support will be made.

As future works, we are developing and integrating the SSCC system with two additional systems: (1) SETWeb – Phone Bill by Web; and (2) SIPI – System to Identify Probable Impostors. Using SETWeb, the clients are able to detect a call made by a clone immediately. The SIPI system can identify a probable future bad-payer knowing your calling patterns.

6. REFERENCES

[1] D.S. Alexander; W.A. Arbaugh; A,D. Keromytis; J.M. Smith. “Safety and Security of Programmable Networks Infrastructures”. IEEE Communications Magazine. Vol. 36. N10, Oct98. Pp. 84-92.

[2] E. Brinksma. ISO 8807 - *LOTOS - Language of Temporal Ordering Specifications*, 1988.

[3] G. Chen, J Rixon, Q. KONG. *Integration CORBA and Java for ATM Connection Management.* DSOM’97. pp. 104-117, 1997.

[4] P. Dowd; J.T. McHenry. “Network Security: It’s time to take it Seriously”. IEEE Computer Magazine, Vol31, N9, t98, pp.24-28.

[5] R. O. Duda, P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[6] H. Garavel. *CADP Manual*. INRIA, France, 1996.

[7] D. N. Gonçalves. *Eyes on the Bill - The growing number of victims of the cloned mobile phone*. (in Portuguese) Veja Magazine, p. 86, 08/10/1997.

[8] L. H. Hauw, Z. Canela, F. Voyer. *A CORBA-Based TMN Prototype With Web Access*. DSOM´97, Sydney, Australia, pp. 81-93, 1997.

[9] ITU-T Recommendation *Q.821 - International Telecommunication Union – Telecommunication Standardization Sector of ITU*. Specifications of Signalling System No. 7. Stage 2 and Stage 3 Description for the Q3 Interface - Alarm Surveillance, 1993.

[10] R. A. Johnson, D. Wichern. *Applied Multivariate Statistical Analysis*, Prentice-Hall, Inc., New Jersey. 1982.

[11] *The Student Edition Of Matlab* – For Ms-DOS Personal Computers: The Problem-Solving Tool for Engineers, Mathematicians, and Scientists. The Math Works Inc., Prentice-Hall, Englewood Cliffs, NJ 08632, 1992.

[12] G. McGraw, E. Felten. *Java Security*. Ed. Wiley, 1997.

[13] G. Pavlou. *From Protocol-based to Distributed Object-based Management Architectures*. DSOM 97. Australia, pp. 25-40, 1997.

[14] E. Simon, Distributed Information Systems – From Client/Server to Distributed Multimedia, McGraw-Hill, 1996, pp. 359-377.

[15] W. Stallings. *Network and Internetwork Security – Principles and Practice*. IEEE Press. Prentice-Hall. IEEE, pp. 462, 1995.