Multi-agent Hierarchical System Based on ElGamal Decryption Algorithm With K+1 Access Levels

Iulia Ştefan, Laura Végh, George Moiş, Stelian Flonta, Szilárd Enyedi, Liviu Miclea
Technical University of Cluj-Napoca, Romania
Iulia.Stefan@aut.utcluj.ro, laura.vegh@gmail.com, George.Mois@aut.utcluj.ro, sflonta@colim.ro,
Szilard.Enyedi@aut.utcluj.ro, Liviu.Miclea@aut.utcluj.ro

Abstract—**In an Internet connected world, security represents a priority. This paper discusses a usage scenario of the ElGamal encryption algorithm with k+1 access levels, by generating a hierarchical tree from the private key algorithm. It underlines the effectiveness in data transfer between different agents as nodes in a hierarchical society. An agent-based system was implemented. This structure was generated using as starting point the algorithm underlined above. Also, the context-awareness authentication is discussed considering the fact that system should provide access to encrypted text.**

Keywords: ElGamal decryption algorithm; hierachical agent model.

I. Introduction

In an expanding virtual world, where information is stored as binary values and therefore can be copied, security becomes a mandatory aspect. Products which ensure data protection are obtained by increasing development costs. These costs are motivated by the amount of personal computers [2] in use that shows the human interest toward online connectivity, storage, shopping, search, communication, mobility; in fact, toward information interchange, all actions needing protection of private personal data and payments details. In May 2009 [3], Computer Industry Almanac Inc. concluded that a quarter of the worldwide population was using the Internet, almost 1.5 billion users. This situation obliges the software providers to develop and search for improved ways to protect privacy and data. This paper presents a method to use in software agents technology for establishing security levels by key generation.

Section 1 presents current situations in electronic threats trying to underline some of the basic methods to impose security of information. Section 2 describes the usage of the ElGamal algorithm for generating the public and private keys, in order to determine hierarchical access to information. Section 3 presents a prototype of a usage scenario in an agent based hierarchical structure. In Section 4, we discuss the possibility of adding context awareness to a structure, where its objectives need restrictive interaction with the environment.

II. Electronic Threats

Mutual authentication between a mobile user and a service provider in [4] represents an attempt to describe a model for such an authentication, key definition, and privacy in access control.

Electronic credential and authentication services are used to verify and link a user to an individual's identity, which will be used within an information system to support the online channel of government service delivery [5].

In GRID applications, the authentication process is based on the Virtual Organization concept. The VO administrator is the one conferring access and establishing some sort of agreements with the Resource (CPU, Network Storage) Providers [10].

In some multimedia applications, authentication establishes the encryption levels [13]. In order to protect the digital media content, the number of trusted parties able to decrypt is reduced to a few.

The need of enhanced security is obvious, when considering the factors representing possible attacks: information the user knows or possesses that can be replicated or stolen. There are different possibilities to reduce the risk of an attack, by implementing a robust authentication process with several credential elements, or using hardware tokens. Several Internet Banking applications, using one-time passwords, impose difficulties to a possible attacker to find the password. Encryption increases the security by adding obstacles in identifying the correct information.

When the situation imposes authentication, there are multiple menaces regarding the authentication event: user, password, sequence of steps exposed to external observers, identity substitution, password guessing, or account modification by intercepting an authenticated session.

There are several ways to reduce the risk exposure: the authentication field's protection against visual interception, reducing the allowed number of login attempts (username, password, biometric credentials), retyping authentication credentials to re-display certain information, no authentication process recording allowed, strict rules when creating credentials to prevent fast guessing or memorizing.

Authentication depends, in fact, on: known information (user, password, number of successive steps in the authentication protocol), hardware devices at hand (token, smart card), and biometric identifiable characteristics (fingerprint, palm print). In the end, the security level relies on the number of security elements implemented.

A way to avoid visual identification of credential granting and the authentication process is automated registrations, key generation and transfer, authentication processes between software agents charged with such functionalities.

A model for such security controls is a hierarchical system, where the upper nodes see all the information designated for the lower ones.

For a software agent involved in message decryption, we considered as contextual information: IP, username, schedule, time and location, all in order to obtain the appropriate private key.

III. ENCRYPTION USAGE

Encryption is a key enabling technology for content security.

With encryption, from an initial document is obtained a new one, with no immediate meaning or readability, by applying a cipher (an algorithm) that usually associates a different symbol to a known linguistic one. To recreate the initial message - to decrypt the message -, the key is necessary, as it represents the information that reveals the connection between the contents.

The ElGamal decryption algorithm with k+1 differential degrees of access rights [16] suggests a tree structure model - the entities claiming the decryption are located in nodes of a tree. Thus, every post can be encrypted by X_i with public key and decrypted by Y_i with private key. There is also the possibility that a group of all posts or messages can be decrypted by other users, using a special private key. A grade 0 user can decrypt all the messages, a grade one user can decrypt a subset of messages that can be decrypted by grade 0 key and so on; a k user can decrypt a single message.

We will now describe the ElGamal decryption algorithm with k+1 differential degrees of access rights EG(k+1)GA.

The algorithm uses the information as a multitude of reduced size messages $M = \{m_{ijk,}\ m_{ijk} \epsilon \{0,1\}, \forall i, j, k \epsilon \mathbb{N}\}$. Let us consider the set I of k pairs of natural values indices. The following notations will be necessary: i… represents $(i_{1,} i_{2,} \ldots, i_k)$ pairs of indices obtained by concatenation and i-k... describes the pairs of indices as $(i_{1,} i_{2,} \ldots, i_{k-j})$ where $j \epsilon \{1,\ldots,k\}$. The information $\{m_{i\ldots} \mid i \epsilon I\}$ is encrypted using a public key by entity X_i and decrypted by Y_i entities using private keys. The intention is that Y^{k-j} k-j decrypts with a private key only the messages included in a partition of the set m_i. Y^0 will be the only one capable to decrypt the entire message.

The algorithm for messages is divided into several steps: generating the tree and creating the node indices, generating the keys, encrypting and decrypting the messages. If necessary, an access level could be eliminated or added.

Defining the tree structure means establishing the users, access levels and the hierarchy: nodes, levels, arcs. The first step is to generate the tree by establishing the indices associated to every node. Every node and the user, associated to a leaf or node, will possess a private key; every leaf has a public and a private one. The indices in a node are generated by pre-ordered tree traversal; each value is given by the child node that was visited. This stage establishes also the level access for every private key.

It is possible to decrypt a message by a private key if and only if there is a chain formed by descendent nodes from the analyzed key to leaf z_i where the message to be decrypted is situated. To generate the private keys, a cyclic group of order q will be chosen, q being a prime number, for which the discrete logarithm problem is difficult; g is its generator.

In case of longer messages, the algorithm could be improved with a symmetric system.

The difficulty in solving the discrete logarithm problem and calculating all the possibilities is given by the number of group G elements. The chosen prime number q is big enough to be represented using 256 bits, and G is cyclic group of order q. Nowadays, the dimension of current keys is equal to 512 and 1024 bits; sometimes, even a larger number is necessary. Another possibility is to use Sophie - Germain prime numbers to prevent external attacks.

From the set $\{1,\ldots,q-1\}$, k distinctive elements will be chosen: $x_1, x_2, \ldots, x_k$. The following functions are chosen $\theta_i : \mathbb{N}^* \rightarrow \mathbb{N}^*$, i=1,...,k, function generated by irreducible polynomials.

From mathematical point of view, a private key is a function defined as:

- for the zero level of the tree, the notation for private keys is SK(k+1)G0 and the formula is:

$$f^0 : \underbrace{N^* \text{ x } N^* \text{ x} \,..\text{x } N^*}_{k} \;\text{->}\; Z^*_{q}. \tag{1}$$

$$f^0(n_1, n_2, \ldots, n_k) = x_1^{\Theta 1(n1)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \tag{2}$$

- for the first level of the tree , SK(k+1)G1

$$f_1^{\,1} : \underbrace{N^* \text{ x } N^* \text{ x} \,..\text{x } N^*}_{k-1} \;\text{->}\; Z^*_{q}. \tag{3}$$

$$f_1^{\,1}\,(n_2, n_3, \ldots, n_k) = x_1^{\Theta 1(n1)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \tag{4}$$

$$f_2^{\,1} : \underbrace{N^* \text{ x } N^* \text{ x} \ldots\text{x } N^*}_{k-1} \;\text{->}\; Z^*_{q}. \tag{5}$$

$$f_2^{\,1}\,(n_2, n_3, \ldots, n_k) = x_1^{\Theta 1(2)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \tag{6}$$

...

$$f_r^{\,1} : \underbrace{N^* \text{ x } N^* \text{ x} \ldots\text{x } N^*}_{k-1} \;\text{->}\; Z^*_{q}. \tag{7}$$

$$f_r^{\,1}\,(n_2, n_3, \ldots, n_k) = x_1^{\Theta 1(r)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \tag{8}$$

…

- for the second level of the tree , SK(k+1)G2:

$$f_{11}^{\,2} : \underbrace{N^* \text{ x } N^* \text{ x} \ldots\text{x } N^*}_{k-2} \;\text{->}\; Z^*_{q}. \tag{9}$$

$$f_{11}^{\,2}(n_3, \ldots, n_k) = x_1^{\Theta 1(1)} x_2^{\Theta 2(2)} \ldots x_k^{\Theta k(nk)} \tag{10}$$

$$f_{12}^{\,2} : \underbrace{N^* \text{ x } N^* \text{ x} \ldots\text{x } N^*}_{k-2} \;\text{->}\; Z^*_{q}. \tag{11}$$

$$f_{12}^{\,2}\,(n_3, \ldots, n_k) = x_1^{\Theta 1(2)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)}$$

…

$$f_{1r}^{\,1} : \underbrace{N^* \text{ x } N^* \text{ x} \ldots\text{x } N^*}_{k-2} \;\text{->}\; Z^*_{q}. \tag{12}$$

$$f_{11}^{\ 2} : \underbrace{N^* \times N^* \times \ldots \times N^*}_{k-2} \ -> Z^*_q. \qquad (9)$$

$$f_{1r}^{\ 1} (n_3, \ldots, n_k)= x_1^{\ \Theta 1(r)} x_2^{\ \Theta 2(r)} \ldots\ x_k^{\ \Theta k(nk)} \qquad (13)$$

…

In the end, by the reunion of the **SK(k+1)G$_l$** private keys, the complete set of private keys can be obtained.

$$\bigcup_{l=0}^{k} \mathbf{SK(k+1)Gl = SK(k+1)GA.} \qquad (14)$$

A public key to encrypt the message is given by a set of three values tuples as:

P(k+1)K ={(q, g, $h_{i\ldots}$) | i... ϵI }, where $h_i\ldots = g^{zi\ldots}$
and the set of private keys SK(k+1)GA becomes the set {$z_{i\ldots}$, f^j, j=0,…, k-1}. Every $Y_{i\ldots}$ user (situated in a node) receives a $z_{i\ldots}$ key and the users situated in lower access levels get keys contained in the set{f^j, j=0,…,k-1}; f^0 is able to decrypt all messages.

To encrypt the message m_i , the node having the public key chooses the elements $X_{i\ldots}$ ϵ{1,…,q-1} and the following equations are obtained:

$$c^1_{i\ldots} = g^{\ y_i\ldots},\ c^2_{i\ldots} = m_i.\ h^{yi\ldots}_{\ \ i\ldots}. \qquad (15)$$

The encrypted message is: {$c^1_{i\ldots}$, $c^2_{i\ldots}$}.

The node Y_i uses q and the private key $z_i\ldots$ to decrypt a k level message {$c^1_{i\ldots}$, $c^2_{i\ldots}$}:

$$\frac{c^2_{i\ldots}}{{c^1_{i\ldots}}^{z_{i\ldots}}} = \frac{m_{i\ldots} h^{y_{i\ldots}}_{i\ldots}}{(g^{y_{i\ldots}})^{z_{i\ldots}}} = \frac{m_{i\ldots} h^{y_{i\ldots}}_{i\ldots}}{(g^{y_{i\ldots}})^{z_{i\ldots}}} = \frac{m_{i\ldots} g^{z_{i\ldots} y_{i\ldots}}}{g^{\ y_{i\ldots} z_{i\ldots}}} = m_{i\ldots} \qquad (16)$$

To obtain a decryption on a k-j level, one needs a set of descendent nodes from the key to the leaf $z_{\ i\ldots}$ of the $m_{i\ldots}$ message.

Starting from the idea of the algorithm presented above, a multi-agent system can be developed where each node in the tree represents an agent.

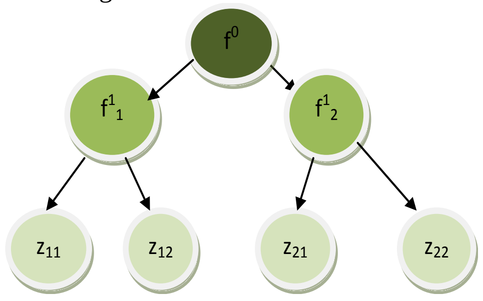

Figure 1. Hierarchical structure

The algorithm assumes the existence of a management entity to generate and share the keys. This entity is also modeled as an agent, without being included in the tree structure. This improves the security, because this entity manages all the keys, both private and public ones.

For example, considering k degrees of access, the tree structure will be as shown in figure 2.

Generating keys is actually creating functions. An initial cyclic group of order q is chosen, q prime number, for which the discrete logarithm problem is difficult; g is its generator. From the set {1, ..., q-1}, k elements and k distinct functions are chosen to be used for private keys calculation. These functions represent irreducible polynomials. To implement these calculations, the Java programming language was chosen.

A function of degree *i* is obtained from an *i-1* grade function, by assigning values to variables in a polynomial component generated in the previous step. For example, for a system with *k+1* degrees of access, the function f^0 has *k* variables, the functions f^1 (on level 1) have *k-1* variables, functions f^2 have *k-2* variables. In the end, the private keys are obtained by assigning values to all variables.

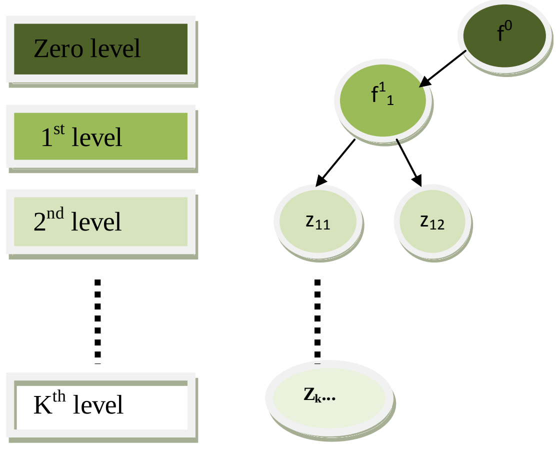

Figure 2. Correlation between level and tree structure

By the method of defining a key, the keys can be constructed one from the other, from level 0 towards level k+1. The reverse is impossible. The Java BigInteger data type is used to represent the key values; it allows storing very large values - an essential requirement for ensuring a high degree of security of the algorithm.

The main objective is to construct a hierarchical structured tree using the private key generation algorithm [17]. This represents one of many ways to build such a system, having as starting point a discrete algorithm problem.

Because the work now is focused on the tree generation, the discussion regarding the decisional Diffie-Hellman assumption and why it holds for chosen group G will be the object of future work, when the system will be improved with the encryption/decryption capabilities. To exemplify, a model for hierarchy implementation is described next, a model implemented in Java Agent DEvelopment Framework(JADE).

The application is based on two packages: one is algPack containing the algorithm implementation classes and the other, the agent implementation classes.

“algPack” is composed of KeyGeneration.java for key generation, Encryption.java for message encryption and decryption.java for decryption.

The second package describes two types of agents. The first is KeyManager; without it, the application does not run. It is in charge of defining the tree structure, starting the agents, generating the keys and transmitting them towards the other agents, the node agents representing the second type of agents..

Initially, online key transfer was considered, but, due to the lack of security in that solution, we consider the possibility to transfer the private keys by offline means, in person.

The tree structure is based on a zero level agent and the other agents placed on inferior hierarchical branches. The zero level agent is characterized by several children, but no parent; the lower levels and the leaf nodes do have parent nodes and children nodes. There is no structural differentiation between leaf node and lower level node because it is important that the tree structure could expand, based on contextual needs.

The KeyManager Agent extracts the necessary data for the tree structure: the number of levels, number of agents and an array describing the number of subordinate agents for every agent. The next phase concerns starting agents and index generation, based on the already mentioned array.
The zero level agent has the index zero. We are assuming it has two children placed on level one, each being described by a single index, 1 and 2. Next, we consider the agent indexed with 1 and we are assuming it has two subordinates (children). Being on the second level as children of first node from level 1, they have two indexes and these are 11 and 12. For every node, its indices are created by copying the parent index and adding at the end its ordinal number until all indices are generated. These indices are necessary for public and private keys and for agent identification in the tree structure. After index generation, the corresponding agents are started. The next phase is represented by key generation and key transfer.

The KeyGeneration class from the algPack package is responsible for key generation. Its constructor receives, as parameters, the array with indices and the number of levels and then generates the public and private keys. These keys are then transferred towards the KeyManager of every agent. Sending the key involves sending a message as ACLMessage.INFORM that a private key will be sent. Because the private key never travels, a trustee will manually install every private key.

Due to the 2:1 expansion of ciphered text over plain text, the consequences are discernible in the necessary volume to store the data and/or the time needed to transfer the encrypted information.

Several BigInteger variables for key storage and manipulation will be set and these variables will be seen as objects in application by using setContentObject method.

Every agent must receive a key. The zero level private key has the highest access level. To receive a message that a key is available from the KeyManager agent, the method "receive" is applied combined with the getContentObject method. If the returned object is not null, a message was transferred and a validation of the message generated by KeyGeneration class is initiated.

In our vision, the agents execute tasks inside a hierarchy. Such tree structures could be identified, for example, in hospitals where the access to private patient data depends on job position or, in software development companies where secrecy represents a management priority in order to maintain the competitors at a safe distance.

In our proposed model, the authority that defines the hierarchy is a supervisor agent. It establishes the private key to send along with the encrypted required information.

The Supervisor Agent will create a hierarchical structure based on stored information about staff and their credentials.

In order to obtain information, field agents running on every network device will send a request message to the supervisor. The supervising authority will send the encrypted information.

The field agent will be able to perceive request information as user/password, date/time, IP, location (inside, outside the buildings) and send it over to the supervisor.

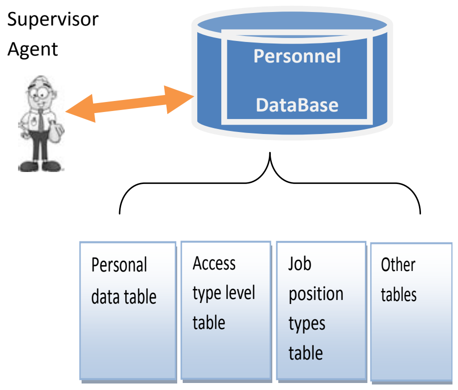

Figure 3. Information resource for Supervisor

Some information, such as a work program or specified scheduled tasks, could be declared inaccessible. For example, why does a lab nurse need to see private data of a patient if he is not scheduled for analysis? The access should be limited to certain time intervals.

IV. CONTEXT AWARENESS

Next, we discuss the possibility to add context-awareness capabilities to the application. This ability begins to become a must in the development of various competitive products, electronic or otherwise, to improve life standards for their owners.

In pervasive computing environments, the humans are continuously surrounded by interconnected devices. Some of these devices are able to access contextual information that can be used as a significant factor in authentication processes, for granting/denying access. User location, date/time, mobile/fixed device used are several examples.

When related to the information content, authentication represents the process to determine, by a pre-established protocol, the identity of the user, user/passwords, user/one-time passwords, using sensors to determine the identity by fingerprint, face characteristics.

„Context includes, but is not limited to the user's location, nearby people and objects, accessible devices, and changes to these objects over time. It may also include lighting, noise level, network connectivity, communication costs, communication bandwidth, and even social situations." [7]. The definition is emphasizing the importance of connectivity and communication.

The user context regarding a computer, mobile device, software driven hardware, "is any information that can be used to characterize the situation of an entity, the user and applications themselves" [11]. In [12], context-aware computing has a major goal: to acquire and fruitfully exploit the information „about context of a device in order to provide services that are appropriate".

But context awareness involves a certain degree of indeterminacy due to its imprecise or incomplete character. For example, not all the sensors send data and, in this case, the retrieved information is not representative for the described situation.

In the case of an agent based structure, designed to offer encryption/decryption capabilities, the service provided should be, at first hand, the authentication. The major problem is the method accurately identifying the user/other service to access the encrypted content. The discussion should be directed toward available means for uniquely identifying a user in our days. Some methods are based on RFID tags, fingerprint sensors, image processing capabilities (retina scan). Other methods could be added to verify different confirmation variables and restrict user access.

If context awareness capabilities are configured for software agents in user identification and authentication processes, the accuracy of contextual information is a binding condition.

In [8], five methods for context reasoning are emphasized starting from their driving factors: the case of past situations, the logic of inference starting from a predicate definition, the ontology as determinations and structure of objective reality, the probability of uncertain contexts and the pre-established system of rules.

By [9], there are four context aware applications based on two criteria: one regarding the presence of automation and the other involving the application objective: to inform or to command something.

In our model, the software agents are responsible for encrypting/decrypting a certain message, based on context data.

To ensure security, multi agent systems use digital signature, proxy servers' certification and hash tables [14].

In [15], the mobile agents can cooperate based on a framework for authentication, using standards already verified in GRID and WEB services.

Further work will be conducted toward an easier way of establishing hierarchy by context awareness capabilities of the application. Based on several pre-established criteria, the application could allow self-validation.

This paper presents an encryption algorithm that allows data access only for an authorized group within a distributed system. This way, sensitive data can be read / modified only by trusted entities in the system. In the decryption algorithm, every node involved in the process is an agent. The decryption key is generated by a higher authority and transferred manually to increase security. The advantage of using such an algorithm to generate the key is the possibility to add nodes without changing all the private keys in the structure.

V. Conclusion

The paper represents a starting point in future work related to the architecture of hierarchical structures and possible usage of encryption/decryption algorithms toward security enhancement in information interchange. It discusses the usage of context awareness capabilities in an encryption/decryption hierarchical system. Sensors could provide enough information to uniquely identify a user, but the main issue still remains: the determination if data provided are accurately obtained because the delivered information allows user access to encrypted content and private key.

References

[1] http://www.nationmaster.com/graph/med_per_com-media-personal-computers&date=2005, April 29th 2011,

[2] http://www.c-i-a.com/pr012010.htm, April 1th 2011,

[3] http://www.c-i-a.com/pr0509.htm, April 3th 2011,

[4] K. Ren, W. Lou, K. Kim, and R. Deng, A novel privacy preserving authentication and access control scheme for pervasive environments, IEEE Transactions on Vehicular Technology 55 (4) (2006), pp. 1373-1384,

[5] http://www.cio.gov.bc.ca/local/cio/standards/documents/standards/electronic_credential_authentication_standard.pdf, April 28th 2011, pp. 15,

[6] L. Chun-Ta, H. Min-Shiang, and C. Yen-Ping, Further Improvement on a novel privacy preserving authentication and access control scheme for pervasive computing environments, Computer Communications 31(2008), www.elsevier.com/locate/comcom, April 3th 2011, pp. 4255-4258,

[7] K. Ohbyung , S.M. Jong, and K. Seong, Context-aware multi-agent approach to pervasive negotiation support systems, Expert Systems with Applications 31 (2006), pp. 275–285,

[8] Z. Daqiang, G. Minyi, Z. Jingyu, K. Dazhou, and C. Jiannong, Context reasoning using extended evidence theory in pervasive computing environments, Future Generation Computer Systems 26(2010), pp. 207-216,

[9] B. Schilit, N. Adams, and R. Want, Context-Aware Computing Applications, Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications, ISBN:978-0-7695-3451, pp. 85-90,

[10] R. Alfieria, R. Cecchinib, V. Ciaschinic, L. dell'Agnellod, A. Frohnere, K. Lo Renteyf , and F. Spatarog , From gridmap-file to VOMS: managing authorization in a Grid environment, Future Generation Computer Systems 21 (2005), pp. 549–558,

[11] A.K. Dey and G.D. Abowd, Toward a better understanding of context and context awareness, 1999, GVU technical report GIT-GVU-99-22, pp. 3-4,

[12] K. Ohbyung, S. Jong Min, and K.W. Seong, Context-aware multi-agent approach to pervasive negotiation support systems, Expert Systems with Applications 31 (2006), pp. 275–285,

[13] H. Kodikara Arachchi, X. Perramon, S. Dogan, and A.M. Kondoz, Adaptation–aware encryption of scalable H.264/ AVCvideo for content security, Signal Processing: Image Communication 24 (2009), link: www.elsevier.com, pp. 468–483,

[14] C. Ou, C.R. Ou, and Y.T. Wang, Security of Mobile Agent-based Web Applications, 2008 IEEE Asia-Pacific Services Computing Conference, 2008, pp. 107-112,

[15] G. Navarro-Arribas and J. Borrell. An XML Standards Based Authorization Framework for Mobile Agents. In Secure Mobile Ad-hoc Networks and Sensors. Springer Verlag, vol. 4075 of Lecture Notes in Computer Science, 2006, pp. 54-66,

[16] S. Flonta, Contributions to the development of models for accessibility and security of information systems, PHD Thesys, 2010, pp. 78-86,

[17] S. Flonta, L. Miclea, and I. Paun, ElGamal with differentiated decryption on K+1 access levels, Applied Computational Intelligence and Informatics, 2009. SACI '09. 5th International Symposium on, Timisoara 2009, pp. 375-380.