Distributed Agent-based Architecture for Management of Crisis Situations using Trusted Code Execution

L. Hluchý*, Z. Balogh* and E. Gatial*
* Institute of Informatics, Slovak Academy of Sciences/Department of Parallel and Distributed Computing, Bratislava, Slovakia
{Ladislav.Hluchy, Zoltan.Balogh, Emil.Gatial}@savba.sk

***Abstract*—This article proposes a distributed architecture designed for management of crisis situations were multiple actors are involved from various organizations with different competences and communicating over IP-based networks including wireless devices. In such settings requirements exist for secure communication and trusted collection of data from various sources. The role of agents in the proposed architecture is primarily coordinated collection of information. In respect to requirements the overall agent infrastructure must be a secure, robust and fail resistant system. Required level of trust for agents is based on a special hardware module which provides trusted computing functionality. In the article we describe such architecture in terms of detailed requirements, design and decomposition to subsystems. We also provide a sample scenario use case inspired by concrete crisis situation. The architecture herein described is being used in scope of an EU integrated project called Secricom therefore we briefly describe the integration points with other systems involved in the project. We conclude with current state of the architecture implementation and with further plans concerning the development of the described architecture.**

I. INTRODUCTION

One of the challenging demands of the communication infrastructures for nowadays crisis management is to add new smart functions to existing services which would make the communication more effective and helpful for users. Smart functions are aimed to be provided by distributed IT systems which should provide a secure distributed paradigm to achieve confidentiality and access to resources. Such infrastructure should further provide a smart negotiating system for parameterization and independent handling of access requests to achieve rapid reaction. By fulfilling the above stated goals a pervasive and trusted communication infrastructure satisfying the requirements of crisis management authorities and ready for immediate application could be introduced. More concretely in crisis situations requirements exist to collect information from legacy systems of various organizations and from human operators in order to semi-automatically manage the crisis mitigation process or enact decisions on various management levels. This collection of information must be enacted in a secure manner while ensuring trust between both parties – information consumers as well as information providers. In a crisis situation many actors participate where the competences between all parties are explicitly defined in a crisis mitigation plan. The gathering of information is enacted either from legacy systems or from human end-users through mobile devices by guided dialog. Herein we present the requirements analysis, design, and system decomposition of a distributed architecture which would fulfill all the above set goals. Further we suppose that the communication infrastructure is IP-based.

We decided to design and implement such architecture using agent paradigm. The distributed agent-based infrastructure is designed as a collection of software services with agent-like features (such as code mobility) which would execute in a secure and trusted manner. Agent technology was selected due to the ability to fulfill such requirements through support of mobile and dynamically deployable executable code. Other advantages of agent-based systems are that they can help overcoming temporal or longer term communication network failures, save network bandwidth by being executed remotely and deliver only the execution results, provide means to execute code on remote host platforms in a trusted and secure manner or deploy code on host platforms on demand. The role of agents in the architecture is primarily coordinated collection of information. The gathering of information is enacted either from legacy systems or from human end-users through mobile devices by guided dialog. In respect to requirements the overall agent infrastructure must be a secure, robust and fail resistant system. Because validity and authenticity of gathered information is a key factor for decision making in crisis management trust must be set between agents and third party information systems. Also agents must trust the host platform providers - remote sites which provide the computational environment for agents. Required level of trust for agents is based on a special hardware module which provides trusted computing functionality.

This article is written as follows: the next section deals with analysis of requirements and security considerations of the proposed architecture. The third section describes the proposed architecture with decomposition to subsystems and envisaged core agents. The fourths section describes a sample scenario which is used as a reference scenario for the infrastructure implementation. The architecture herein described is being integrated in scope of an EU integrated project called Secricom. Therefore the fifth section introduces the Secricom project and the integration points of the proposed architecture with other systems involved in the project such as the PTT - Push To

Talk system, SDM - Secure Docking Module or MBR-Multi Barrier Router. The last section concludes the article as well as presents our current achievements and plans concerning the implementation of the proposed architecture.

II. Requirements and Security Considerations

In order to define concrete security requirements for our architecture we sketch the basic infrastructure in which agents will operate (Figure 1):

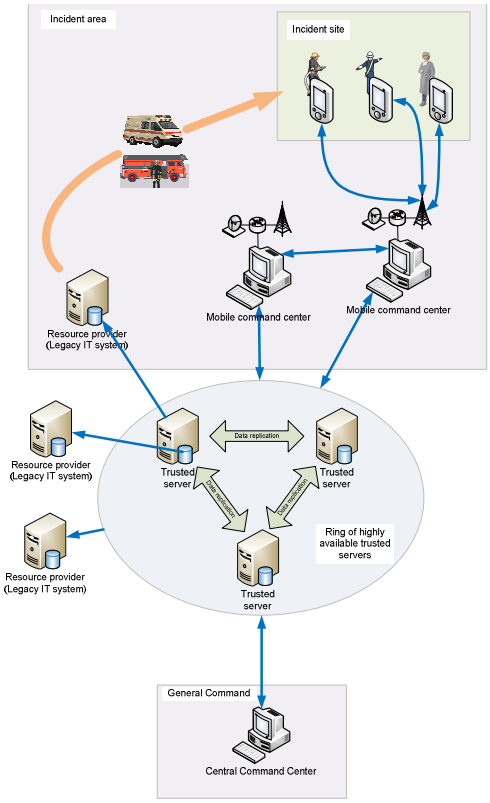

Figure 1. Infrastructure and Host Platform Providers in the Distributed Agent-based Architecture.

The home platform for agents is a network of Trusted Servers (TS). According to [1] the platform from which an agent originates is referred to as the home platform, and normally is the most trusted environment for an agent. This is also true for our agents – the network of TS is a managed set of systems with defined security policies and possibly managed by a central authority. From here agents are delegated to host platforms to gather data and information. Agents are mainly executed on remote sites which provide the computational environment in which agents operate. We will refer to these sites as to host platforms (or agent platforms).

In general, any party which wishes to join the implemented architecture and to provide information from their legacy systems or users must introduce a host platform for agents. We refer to such parties as Host Platform Providers (HPP). From end-user requirements, the following HPPs were identified (Figure 1): Resource Providers – hospitals, fire brigade, police, warehouses or any other entities which can play a role in the mitigation of crisis situation; Command Centers – mobile (nomadic) centers which coordinate locally the incident site; and General Command Center and Operators – usually located in one place or at least tightly interconnected.

The features of agents encompass several chosen attributes: code mobility (without execution state) – ability to move code to different platforms and execute there, within the project we do not plan to support execution state mobility (as there is no such requirement); autonomy – ability to autonomously deliver gathered data to one or several optional destinations; reactivity – in some cases agents will perceive the context in which they operate and react to it appropriately (e.g. agents can monitor availability of some resource and notify the requestor).

Since agents collect information which is often of high sensitivity, confidentiality and security, while at the same time requirements for action or decision traceability exist, agents must be provided with a secure, trusted and attested execution environment. In the following we identify main agent-related security threats. A detailed explanation of generic mobile agent security aspects is discussed in [1]. Generally, four threat categories are identified: Agent platform attacking an agent, Agent attacking an agent platform, Agent attacking another agent on the agent platform and other entities attacking the agent system. The last category covers the cases of an agent attacking an agent on another agent platform, and of an agent platform attacking another platform, since these attacks are primarily focused on the communications capability of the platform to exploit potential vulnerabilities. The last category also includes more conventional attacks against the underlying operating system of the agent platform.

A. *The Host Platform Attacking the Agent*

The main threat for agents in foreign execution environment of host platforms is the “malicious host problem”. This is one of the main problems in the class of “an agent platform attacking an agent“. Simple explanation of “malicious host problem” is provided in [2]: “Once an agent has arrived at a host, little can be done to stop the host from treating the agent as it likes”. Therefore, the main requirements from the agent-side are laid out in respect to the “malicious host problem”. Concrete security requirements of agents in respect to the host platform are as follows: isolated execution environment for agent execution – not only virtual isolated execution environment but dedicated isolated hardware preferred; means to attest the platform required in order to detect if the host platform is in trusted state; and protected storage for credential data (such as PKI’s secret key).

B. *The Agent Attacking the Host Platform*

There are also threats stemming from an agent attacking an agent host platform. Therefore reversely a host platform has also requirements in respect to agents. These requirements are more evident when provided in context of HPPs security requirements:

1. HPPs do not want to install and execute any external application on their systems in line with their strategic legacy applications.

2. HPPs prefer to have a dedicated and isolated system for agent system which would connect to their legacy system in a secure predefined way.

3. HPPs want to be able to control what (data), when and by who (traceability) is provided to agents.

4. HPPs want to be able to configure set of applications executable on their side. Agents must be therefore audited and verified thus mediate trust to executable agent code.

The agent platform has the following security requirements in respect to agents: isolated execution environment for agent execution - agents must be executed in isolated environment (isolated hardware preferred), so an agent can not harm legacy systems; means to monitor and trace agents activity; and means to configure the set of agents executable on the host platform. In order to track agents, any agent in the platform must be cryptographically signed. Only agents signed with trusted authority and assigned to selected category will be trusted by a host system. Agents need to send signed messages to Trusted Servers.

C. *The Agent Attacking another Agent*

It is required that any agent which will be used in the system will need to be audited and certified by a central authority. In turn every host platform will be configured to execute only agents which are certified. These two security policies should ensure that malicious agents will not be deployed into the infrastructure. Only a breach of the set security policies might lead to potential agent-to-agent security risk.

Moreover, each agent should be executed in a relatively isolated virtual environment with limited access to data of other parallel executed agents on the same host platform.

D. *Other Entities Attacking the Agent System*

Agents will also connect to legacy systems (third party software). Therefore a risk of an agent being attacked by a legacy system but also vice versa – the risk of attacking legacy system by an agent also exists. The host platforms will need to provide some kind of connection to legacy systems. We explicitly presume that this will be a network connection. On any network connection there is an eavesdropping risk. Therefore another requirement which arises from agents to the host platform is secure protected connection to legacy systems. Physical security of network connection can be achieved either by direct cable connection of the host platform with legacy system or by managed network security (managed switch with well defined security policies). The data transport security will be achieved primarily through encryption.

III. ARCHITECTURE

In this section we present a distributed architecture designed for management of crisis situations were multiple actors are involved from various organizations with different competences and communicating over IP-based networks including wireless.

The architecture (Figure 2) is designed for mobile services with agent-like features (mobility, pro-activity) which would execute on secure devices. In general consists of interconnected trusted (TS) and untrusted servers (US). TS carry out the following tasks: registry of services, users and modules, public encryption keys, the agent base (base of mobile code) or generic security politics. Each agent has features and “abilities”, which are used for the enactment of certain processes.

The enactment of processes is inspired by the domain of management of crisis situations in which collection of information from multiple systems is required. The whole

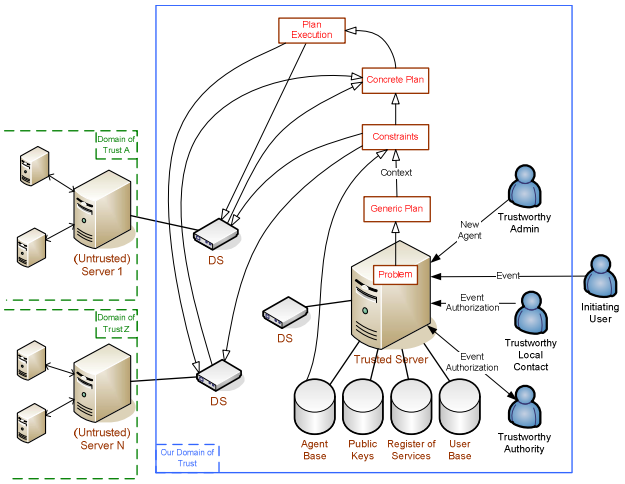

Figure 2. Distributed architecture designed for management of crisis situations.

process starts with the specification of a problem in the form of dialog. Further an agent specifies the most serious problems which were rendered by the crises situation. Based on the type of crisis situation and on the region where the crisis has occurred appropriate actions are initiated for each crisis situation type. The system will semi-automatically generate plausible generic plans of possible solutions (mitigation plans) of identified problems. In the next step the specification of context will be enacted in order to be able to generate the constraints of the crisis situation. Relevant resource providers will be identified in the central database based on constraints generated in the previous step. Agents which are able to query selected servers will be selected from the agent base. Information about available capacities of resource providers will be retrieved and sent back to central trusted server base. The system will then generate a concrete plan of crisis situation resolution based on the retrieved disposable resource capacities. The last step is execution of prepared plan for the concrete crisis situation.

In order to fulfill the architectural requirements set the infrastructure was decomposed into subsystems. These subsystems are related and will cooperate together through defined interfaces. The list of all subsystems is in the Table I:

TABLE I.
SUBSYSTEMS OF THE ARCHITECTURE

Subsystem	***Basic description and functionality***
Distributed Secure Agent Platform (DSAP)	The core agent platform. Will provide means for agent deployment, execution, migration and communication.
Process Management Subsystem (PMS)	Based on the plan collected from users will generate a plan of activities. Executes the plan.
Agent Repository (AR)	Database of system users, agents and their certificates. Process of accreditation of agents.
Public Key Infrastructure (PKI)	Certification and verification of agents, users and resources.
Resource Inquire System (RIS)	Will provide information which system to query for specific information.

The purpose of the Distributed Secure Agent Platform (DSAP) is to provide an execution environment for different types of agents. The main aim of Process Management Subsystem (PMS) is execution of processes and coordination of involved agents in the emerged crisis situation. The plan scenario for each type of crisis situation will need to be pre-prepared in form of an abstract process. The exact execution of such plan in a concrete situation will depend on the context of the crisis situation. The agents available within the herein proposed infrastructure have to be stored on Trusted Servers, from which they can be requested for deployment on the side of HPPs – this functionality is encompassed within the Agent Registry (AR). Public Key Infrastructure (PKI) will allow certifying, and subsequently verifying, all the objects deployed in the infrastructure. Agents will require having information about the information sources which can be queried in order to retrieve information about resource availabilities. Resource Inquire System (RIS) will provide an interface which will provide such capability.

Additionally there is a set of core agents which are required in order to ensure functionalities of the architecture. List of agents including their brief functionality description is in the Table 2:

TABLE II.
CORE AGENTS USED IN THE ARCHITECTURE

Agent	***Functionality***
Information Delivery Agents (IDA)	IDA agents will need to connect to legacy information systems of third parties to retrieve information about available resource capacities
User Communication Agent (UCA)	Will communicate with users in a form of guided dialog through electronic device. Will include authentication and interface to authorization of the user.
IP Agent (IPA)	An agent able to configure IP devices such as routers.

IV. SAMPLE SCENARIO

Herein we present a sample scenario in which coordinated information collection using agents takes place. The presented scenario is not a typical crisis scenario were emergency responders are involved but demonstrates all the important and useful abilities of agents in such distributed settings.

The schema on the Figure 3 depicts an imaginary epidemic crisis scenario: A country has a sudden rise in number of people sick from an epidemic flu. There are many infected people and others are suspected to be sick soon. The organization responsible for mitigation of epidemic is UVZ. Personally a Chief Officer (CO) at UVZ is responsible for such situations. CO decides to set warning level to 5. As part of this warning level UVZ needs to make sure that there are sufficient supplies of vaccines in regional UVZ branches (RUVZ). Such information must be retrieved from legacy systems of each RUVZ. CO must delegate this information collection to an officer at another organization called SHR (O2). After the officer at SHR finds out about the supplies at individual RUVZ he needs to delegate the task of distributing additional sufficient amount of vaccines to an Officer at SHR Warehouse. Information about complement shipments of vaccines to RUVZ is sent by SHR Warehouse Officer directly to SHR Officer which

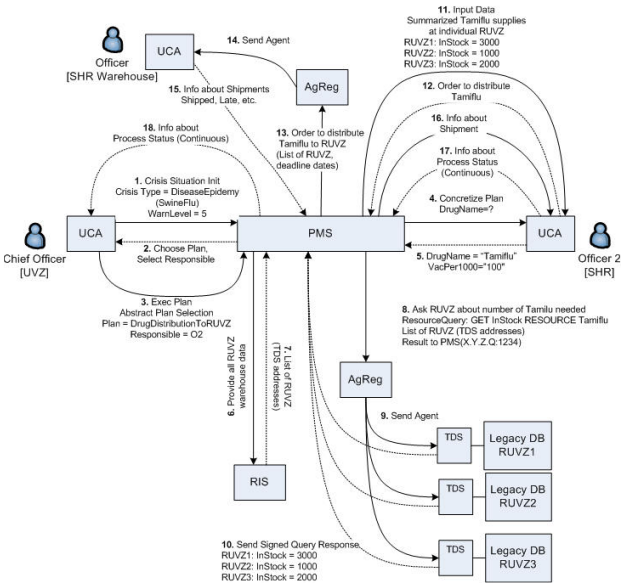

Figure 3. Schema of a sample crisis scenario.

redirects this information to CO at UVZ. Concrete steps of the scenario are the following:

1. CO initiates a new Crisis Situation in the UCA user interface, where CO opens the UCA and selects "Initiate Crisis Scenario" of type "DiseaseEpidemic" and sets "Level" to value 5.

2. PMS is informed about new crisis. PMS checks if request is signed and whether CO is trusted and has rights to initiate the mitigation. After confirmation is send back to CO's UCA, possible (pre-prepared) mitigation plans and list of qualified responsible persons (officers O1-O3) is generated.

3. CO selects the right mitigation plan and decides to ask O2 to supervise this process. The process is in this stage in an abstract format, i.e. details are not concretized.

4. PMS informs O2's UCA that he is responsible for supervising the process. He is also asked to concretize the process, in this case by specifying "DrugName" and "VacPer1000" properties.

5. O2 accepts to supervise the process and specifies required properties.

6. PMS is informed about "DrugName". PMS needs to find out who is able to supply "DrugName" resource. PMS contacts RIS with a query to provide all suppliers of "DrugName" resource.

7. RIS replies with a list of RUVZ.

8. PMS now can query all RUVZ for availability of resource called "Tamiflu". PMS formulates the query and sends List of RUVZ and where to send the result. Query is send to AR (AgentRepository). Also deadline for result delivery is specified.

9. AR must select an appropriate agent (of IDA type) for each RUVZ because each RUVZ might have different legacy systems. AR sends out agents to collect relevant data. Agents are deployed to each resource provider (RUVZ in this case).

10. Agents send back their response to query.

11. Data are collected by PMS and after deadline sent in consolidated form to O2. O2 reviews the data where he can see current stock amounts at each RUVZ warehouse.

12. O2 creates order to distribute missing drugs to RUVZ. This will be a request for resources to be ordered/delivered. Request is send through PMS to Officer at SHR Warehouse. O2 is able to specify that each region should be equipped with 100 vaccines per 1000 people. Based on information about population of regions vaccine numbers are computed order is created.

13. PMS requests AR to send OrderAgents to the officer. At SHR Warehouse.

14. ShipmentAgent is sent out to each RUVZ.

15. ShipmentAgent informs PMS about the status of deliveries.

16. PMS informs O2 about status of deliveries.

17. O2 informs PMS about process status.

18. PMS informs CO about process status.

Please note that in this scenario communication between users is proposed to be done using UCA – User Communication Agents. UCA is able to communicate with users either through computer or through a mobile device. UCA collects information from a user through a sequence of simple forms. UCA summarizes the form results and sends it to PMS for further processing. The IDA – Information Delivery Agent is used for retrieving information from legacy systems. There might be different types of IDA suitable for different legacy systems of various resource providers. There are also other agents used in the scenario such as OrderAgent or ShipmentAgent – which are specific purpose agents. The only agent not mentioned in this scenario is the IP Agent – this agent is intended to configure routers or other active configurable IP devices. IPA can semi-automatically configure the network according to current need of the crisis responders. For example in our sample scenario we could use IPA to prioritise the communication between the officers at UVZ and SHR.

V. Integration With Other Systems

The architecture herein described is being used also in scope of an EU integrated project called Secricom [9]. The implementation of the architecture in the project is called Secure Agent Infrastructure (SAI). SAI solves the timely delivery of relevant information, obtains the information about available resources (material or human) and helps the authorities manage the distribution of such resources. SAI also communicates with legacy information systems operated by agencies and institutions involved in the crisis resolution. There are several systems to which SAI gets connected. Concretely we describe integration with SDM - Secure Docking Module, PTT - Push To Talk system and MBR-Multi Barrier Router systems.

In order to overcome threats described in section II, agents require safe secured place to store cryptographic credentials (PKI secret keys) and provide interfaces to retrieve these keys, ways to attested platform (execute on a host platform which is in a trusted state) and provide interface to safely communicate with legacy systems. All these functionalities are provided by a hardware module called Secure Docking Module (SDM) [5, 6]: SDM is a key storage device with local attestation and verification capabilities. SDM establishes trust on the host platform where agents are being executed – called Trusted Docking Station (TDS). A trusted state is a specific software configuration. This software configuration is measured by using a Trusted Platform Module (TPM). A TPM is a special security chip which provides amongst other functionalities the protected capability of measuring the software configuration of its host device. A TPM must be present in the TDS. The combination of a SDM and a TDS is called a Secure Docking Station (SDS) as shown on the Figure 4.

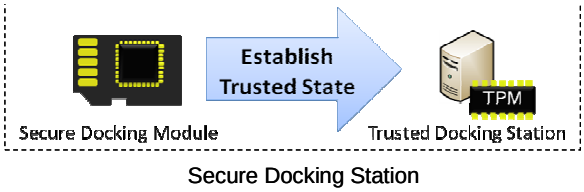

Figure 4. Schema of how SDM, TPM, TDS and SDS relate.

SAI uses SDS deployed in a physical proximity of the legacy information system, preferably in the same room and acts as a secured and trusted extension of the Secricom infrastructure. SAI executed in SDS eliminates the exposure of the legacy IS to the outside world and allows the operator of the legacy IS to have increased trust in the information consuming party. SAI can process the information received from the legacy IS while conserving the network bandwidth, limiting possible exposure of sensitive data – sending back the results only and continuing data processing even if the connection to the outside world is intermittent. More information about these technologies can be found in [5, 6].

Secricom PTT (Push To Talk) is a client-server communication system using IP protocol and is developed by a Slovak company Ardaco [7]. PTT optimizes and protects the way teams of people communicate without being concerned about misuse of information. Regardless of communication endpoint (mobile, laptop or handheld) the communication is secure and safe. SAI connects to PTT servers in order to communicate with users. Concretely UCA agent is being integrated with PTT through implementation of simple forms. PTT takes care of delivering and displaying the forms on the user side, while UCA is responsible for the form processing. Forms are being automatically generated by PMS during run-time in accordance to overall process status and process configuration. Integration of UCA with PTT adds a more flexible way of data collection and user communication to Secricom infrastructure.

The Secricom Multi Bearer Router (MBR) is a modular router development platform and is developed by a UK-based company QinetiQ [8]. MBR provides one of the core Secricom platforms and delivers the IPv6 network enabling overlay. It provides seamless, ad-hoc end-to-end connectivity between various legacy and emerging next generation, static and mobile bearers, networks and user access devices. SAI integrates with MBR using the IPA agent. MBR must be equipped with SDS in order to provide trusted and attested execution environment for agents. IPA agent is able to configure different properties of network such as communication prioritization or bandwidth control between different bearers.

VI. Conclusion

In this article we have analyzed the requirements for agent-based systems and have proposed a distributed architecture designed for management of crisis situations. We have decomposed the proposed agent architecture to subsystems and identified several core agents to be used in an architecture implementation. A sample scenario was described, which demonstrates the possible use of individual agents in case of a crisis in a distributed IP-based communication infrastructure. Lastly we described the use of the proposed architecture in scope of an EU integrated project called Secricom.

Currently the proposed architecture is being implemented in Java [10] using a Jini [11] services technology framework. All the subsystems identified in Table I are implemented and are in pre-prototype version. There are also core agents (Table II) implemented and deployed in the system. Currently integration work is in progress with SDM, PTT and MBR systems as described in section V.

Our overall goal is to provide full prototype implementation of the proposed framework. We believe that besides crisis management there are many other application domains where trusted code execution using agents is appropriate to use and where the proposed distributed agent-based architecture would suit well. In the future we plan to identify other suitable problem domains for our architecture and customize the system for use in other challenging distributed infrastructures.

Acknowledgment

This work is supported by projects SECRICOM FP7-218123, APVV DO7RP-0007-08, SEMCO-WS APVV-0391-06, VEGA No. 2/0211/09, VEGA 2/0184/10.

References

[1] Wayne Jansen, Tom Karygiannis: Mobile Agent Security – NIST Special Publication 800-19. National Institute of Standards and Technology, Computer Security Division, Gaithersburg, MD 20899.

[2] Niklas Borselius: Mobile agent security, Electronics & Communication Engineering Journal, October 2002, Volume 14, no 5, IEE, London, UK, pp 211-218.

[3] Kocis et al: SECRICOM – Analysis of external and internal system requirements. Deliverable report D2.1, the SECRICOM project, February 2009.

[4] O'Neill et al: SECRICOM – Analysis of Crisis Management System Requirements. Deliverable report D2.2, the SECRICOM project, February 2009.

[5] Šimo et al.: SECRICOM - Security requirements and specification for docking station module. Deliverble report D4.1, the SECRICOM project, April 2009, URL: http://www.secricom.eu/public-deliverables

[6] Hein et al.: Functional specification of the Secure Docking Module. Deliverable report D5.1, the SECRICOM project, May 2009, URL: http://www.secricom.eu/public-deliverables

[7] Ardaco homepage. URL: http://www.ardaco.com/

[8] QinetiQ homepage. URL: http://www.qinetiq.com/global.html

[9] Secricom Project Homepage. URL: http://www.secricom.eu/

[10] JAVA Home. URL: http://java.sun.com/

[11] Jini Homepage. URL: http://www.jini.org/