A Solution for Anonymous Access of IT Services and its Application to e-Health Counseling

Rafał Leszczyna

European Commission, Joint Research Centre,
Via Enrico Fermi, 21020 Ispra (VA), Italy

5 September 2005

Abstract

E-Health is a domain which refers to the use of modern information and communication technologies to provide citizens with health services and information. In the e-Heath particular attention is paid to users' privacy. Patient data must be protected from a disclosure to unauthorized third parties, especially in the face of the progress in genetic research, where obtaining health data related to one user may lead to deduction about the health of the user's relatives. The paper presents a solution for anonymous access of IT services and its application to an e-Health counseling scenario.

1 Introduction

Recently we have proposed two untraceability protocols (further distinguished as *Protocol I* and *Protocol II*) dedicated to mobile agents environments [1, 2]. After investigation of security and performance of the protocols [1, 3] we implemented a mobile agent based e-Health counseling service which takes advantage of Protocol I. This implementation served us as a feasibility study of applying untraceable agents to a medical application. This article describes results of this work.

1.1 Anonymity

Anonymity is the property that ensures that a user may use a resource or a service (the *items of interest*) without disclosing his/her identity [4]. *Untraceability* is a subclass of communication anonymity [5] assuring that the identity can not be inferred by tracing a message.

Anonymity plays a crucial role for various activities conducted in the Internet. For example users may stay reluctant to engage in web-browsing, message-sending, and file-sharing, unless they can receive guarantees that their anonymity will be protected to

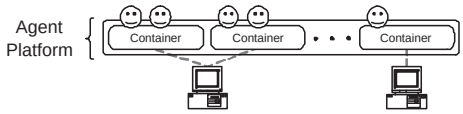


Figure 1: An agent architecture: agents operate on the agent platform deployed over multiple *containers*. Containers are installed on hosts.

some reasonable degree [6]. Gulcu et al [7] describe four categories of internet applications where anonymity is required[1]: discussion of sensitive and personal issues, information searches, freedom of speech in intolerant environments and polling/surveying.

The demand for users anonymity grew even more with the advent of the e-Health domain, where any unauthorized possession of personal data could give one an occasion for conducting misdemeanors (e.g. hindering employment of people with addiction or mental problems occurred in their past; or increasing insurance rates for people with higher morbidity). Thus in many cases providing anonymous access to e-Health services became crucial. One of such services is the e-Health counseling service [8], which is in the scope of this article.

1.2 Software agents

According to the definition based on etymology of the word *agent* (the term *agere* means: to set in motion, drive, lead, conduct, act, do [9, 10]) a *software agent* is a software entity able to act. Then we distinguish between particular types of agents by means of adjectives which describe properties of agents (e.g. *autonomous* – able to exercise control over their own actions or *mobile* – able to roam networks freely). These adjectives were combined from the Franklin's and Graesser's [11]; and Murch's and Johnson's [12] classifications.

Software agent technologies offer a natural extension of component based approaches – where the static components are replaced by the agents. The agents, depending on their complexity, can expose different levels of autonomy and „intelligence" what makes them similar to humans. This gives place to a new approach to software development, in which applications are constructed based on abstractions of elements of the real world [13, 14].

The conventional software applications (desktop programs, applets, servlets etc) are unable to autonomously change their location from one computer to another. To allow this functionality, an additional support must be provided. In case of mobile agents this is done by introducing *agent platforms* – a middleware installed on an operating system. The agent platforms are system independent and can be deployed on various computer architectures (personal computers, mobile devices, diverse operating systems etc). Figure 1 illustrates the fundamental components of agent architecture.

After the evaluation of nine FIPA [15] compliant agent platforms [16] we chose the Java Agent DEvelopment Framework (JADE) [17] as a middleware for our experiments. JADE fully implements the FIPA communication model along with its func-

[1]This is important to note that the authors don't claim this set to be exhaustive.

tional components; and offers various additional features such as a Yellow Pages service or support for agents' mobility. JADE can be flexibly extended by means of JADE add-ons (see the next section) [18].

2 The Protocols

We proposed an untraceability protocol for mobile agents – *Protocol I* and investigated its security [1, 2]. The study showed that the protocol, while having the advantage of being effective, was prone to the costly, nonetheless possible to perform – *cordoning-off attack* – i.e. the attack in which an attacker compromises all containers surrounding the source one [1, 2]. Then we introduced *Protocol II* which eliminates this vulnerability at the cost of putting more computation workload on the source container and restricting agent autonomy at the beginning of its route. Hereby we gave choice to the users: they can use more effective but slightly less secure Protocol I, or choose Protocol II, to receive stronger protection.

Protocol I was implemented following practices of extending JADE [18]. The recommendations say that the framework may be flexibly (new functionalities can be easily added and removed) and transparently (adding a new feature does not affect the applications which do not use it) extended by means of add-ons [18]. This implementation gave the foundation for the performance tests (described in the next paragraph) and was used to the realization of anonymous e-Health service presented in this paper.

The analysis of performance of the protocols comprised two parts: analytical and empirical [3]. The first part, focusing on time complexity of standard and untraceable agents, resulted in analytical formulae estimating the overhead brought in by untraceability. The empirical part involved proposing and implementing a benchmark architecture and running a series of tests in order to obtain performance characteristics of traceable and untraceable (utilizing Protocol I) agents. The empirical results confirmed the analytical estimates and provided more concrete data related to Java enabled agent platforms. The studies showed that untraceability cost would be twice as much as the cost without untraceability, if all mobile agents in the environment were replaced by untraceable agents. But since real agent applications are composed of both, traceable and untraceable agents, the resulting total overhead would be relatively small (for instance, reaches the level of 1,92% according to the experiments described in [3]).

Comparing to other solutions [19, 20], the advantage of the protocols is that they support agent's autonomy in choosing the migration path [1, 2]. In the sequel we present the description of Protocol II. The specification and the details of Protocol I can be found in [1, 2],

2.1 Assumptions

A.1. It is assumed that an agent platform guarantees that third parties (other agents, users) are not informed about the presence of other agents if the latter do not want to do so. It means that it is impossible to introduce any agents aiming at observing and following other agents.

A.2. It is also assumed that each container is well isolated, so it is impossible to learn its state from outside.

A.3. Each container owns an individual symmetric key and a private asymmetric key.

A.4. A container must also have access to all needed public keys of other containers.[2]

A.5. Each container stores the identifier of the previous container visited by an agent until the agent leaves out.

A.6. This identifier is available to the agent.

2.2 Protocol II

Protocol II was designed as an extension of Protocol I to make it resistant to the cordoning-off attack, i.e. the attack against the anonymity of an agent source container which during the security analysis (described in [1, 2]) was recognized as effective. In the protocol, the source container arbitrarily chooses a particular number of platforms which the agent has to visit initially and creates the list of their encrypted identifiers. This serves as the initial route letting the agent to obfuscate its source address. After leaving this initial route, the agent is free to make decisions about which platforms to visit next. It autonomously roams the network to achieve its goal and after succeeding returns to the last container of the initial route. Then, to come back to its source container, it must follow the initial route in the reverse order. In Protocol II, only the last container of the route knows the destination address but it is not able to recognize the source address. The first container knows the source address but without the knowledge of being the first in the route (it could be just another container on the route).

The protocol is presented in pseudocode (Listing 1). The variables m and i are used only to illustrate subsequent steps of the algorithm and are not stored explicitly in the agent state.

Listing 1: The Protocol II pseudo-code.

// Preparations

1. The agent's base container defines the initial route of an arbitrary length i: AC_2, AC_3, ... AC_{i+1}
2. The container encrypts the route into the string – $K_2(ID_3, H(ID_1, ID_2, ID_3), N_2)$, $K_3(ID_4, H(ID_2, ID_3, ID_4), N_3)$, ..., $K_i(ID_{i+1}, H(ID_{i-1}, ID_i, ID_{i+1}), N_i)$ using the public keys K_2, K_3 , ..., K_i of the chosen platforms, and stores it into the agent's state
3. The container produces the return route string $K_{i+1}(ID_i, H(ID_{i+1}, ID_i), N'_{i+1})$, $K_i(ID_{i-1}, H(ID_{i+1}, ID_i, ID_{i-1}), N'_i)$, ..., $K_2(ID_1, H(ID_3, ID_2, ID_1), N'_2)$ using the public keys K_2, K_3, ..., K_{i+1}, and stores it into the agent's state
4. The container encrypts the agent's goal G using the public key K_{i+1} of the last container AC_{i+1} on the initial route, obtaining $K_{i+1}(G)$, and stores it into the agent's state

// Following the initial route

5. m=2
6. The agent moves to the container AC_m
7. If the string of encrypted container identifiers is empty go to 14 // at the same time it means that the agent reached the last container on its initial route (and m=i+1 at the moment)

[2] However, the actual implementation of the key management is out of scope of this article.

8. The container AC_m decrypts the dedicated part of the string of encrypted container identifiers $K_m(ID_{m+1}, H(ID_{m-1}, ID_m, ID_{m+1}), N_m)$ obtaining the identifier of the succeeding container ID_{m+1}, the nonce and the hash value $H(ID_{m-1}, ID_m, ID_{m+1})$
9. The container verifies the hash value $H(ID_{m-1}, ID_m, ID_{m+1})$
10. If the verification fails go to 31
11. The container removes the $K_m(ID_{m+1}, H(ID_{m-1}, ID_m, ID_{m+1}),N_m)$ entry from the string of encrypted container identifiers
12. m=m+1
13. Go to 6
14. The container AC_m, where m=i+1 at the moment, decrypts $K_m(G)$ to unhide the agent's goal
15. The identifier ID_{i+1} of the container AC_{i+1} is stated explicitly in the agent's state

// Autonomous migration to achieve the goal

16. The agent migrates autonomously from one to another agent container to achieve its goal G
17. The agent decides to return to its base container

// Returning to the base container

18. The agent moves back to AC_{i+1}
19. The identifier ID_{i+1} of the container AC_{i+1} is removed from the agent's state
20. The container AC_{i+1} decrypts the dedicated part of the string of encrypted container identifiers $K_{i+1}(ID_i, H(ID_{i+1}, ID_i), N'_{i+1})$ obtaining the identifier of the proceeding container ID_i, the hash value $H(ID_{i+1}, ID_i)$ and the nonce
21. The container verifies the hash value $H(ID_{i+1}, ID_i)$
22. If the verification fails go to 31
23. m=m-1
24. The agent moves to the container AC_m
25. If the string of encrypted container identifiers is empty then finish() // at the same time it means that the agent reached its base container (and m=1 at the moment)
26. The container AC_m decrypts the dedicated part of the string of encrypted container identifiers $K_m(ID_{m-1}, H(ID_{m+1}, ID_m, ID_{m-1}), N'_m)$ obtaining the identifier of the proceeding container ID_{m-1}, the nonce and the hash value $H(ID_{m+1}, ID_m, ID_{m-1})$
27. The container verifies the hash value $H(ID_{m+1}, ID_m, ID_{m-1})$
28. If the verification fails go to 31
29. The container removes the $K_m(ID_{m-1}, H(ID_{m+1}, ID_m, ID_{m-1}), N'_m)$ entry from the string of encrypted container identifiers
30. Go to 23

// The verification of has value has failed

31. Perform the emergency scenario

---

3 Applying the untraceability to e-Health service

After we had studied the security and the performance of the protocols [1] we wanted to investigate their applicability to real IT services. Since e-Health is one of the IT areas where user anonymity is in strong demand, we decided to develop an anonymous medical service. Finally our choice was the anonymous e-Health counseling service, which as belonging to the large group of knowledge management based services, has this advantage that the solutions designed for it can be simply adapted to other services belonging to the group.

3.1 The e-Health counseling scenario

The e-Health counseling service embraces all functionalities and activities necessary to provide a user with a comprehensive medical, pharmaceutical, and dietetical knowledge related to the user's health problem. The sample scenario of accessing the service looks as the following:

John would like to obtain a comprehensive information about embarrassing symptoms which he thinks are related to some disease. Since the subject is subtle, he would like to obtain this information anonymously so nobody could trace the author of the question.

He would like to have the description of the disease given by different doctors, from different medical environments. Moreover the knowledge must be supported by direct citations from biomedical journals. He would like to be presented with adequate medicines and their generics with an account of prices. He wants this information to be accompanied with some diets helping with faster removal of the symptoms. With food products and their prices and delivery information.

John accesses the anonymous e-Health counseling service using his PC; and he is asked for a description of the symptoms. He fills up the form and is informed that soon, depending on availability of health professionals, he will receive the information. System informs that he will be notified about availability of the results by the message sent to his mobile phone.

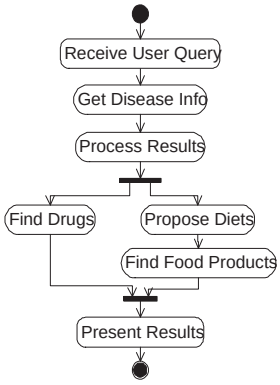


Figure 2: The UML activity diagram of health counseling scenario

We have identified the primary actions which will occur in the realization of the scenario (see Fig. 2):

- *Receive User Query* – obtaining the description of symptoms of the health problem from a user;
- *Get Disease Info* – communication with different knowledge sources in order to accumulate complete information about the health problem (e.g. a detailed description of the disease, a list of cures for the disease etc); this information may contain redundancies and inconsistencies;
- *Process Results* – removing the redundancies and the inconsistencies from the information to make the information substantial;
- *Propose Diets* – preparing (or retrieving – if they are already in the system) the most appropriate diets supporting the treatment for the disease;

- *Find Food Products* – searching for comprehensive information about food products related to the proposed diets;
- *Find Drugs* – retrieving data about the medicines presented during the Get Disease Info process;
- *Present Results* – notifying the user about the availability of the results and presenting them to the user.

All of these actions must be performed in the manner endorsing the privacy policy, so nobody would be able to infer the identity of the author of the query.

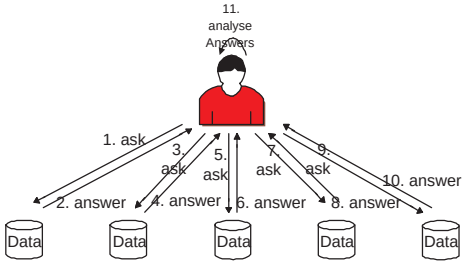
3.2 The design

The e-Health counseling service represents an ample group of IT services where in order to retrieve a complete information many distributed information sources must be accessed. The unique characteristics of mobile agents, related to their mobility, make them an ideal tool for realization of such services.

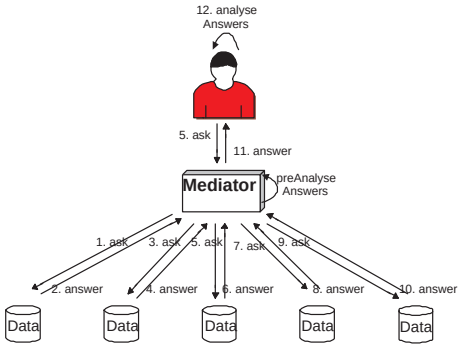
Typically, without an application support the instantiation of the scenario would be as the presented in Figure 3a. A user would have to access each site containing any useful knowledge and when visiting a site, to make notes for later comparison. The whole process would be time-consuming and requiring active (and patient) participation of the user. Employing a client-server application would improve this situation through delegating the data-mining tasks to the IT system (see Fig. 3b). But not until mobile agents are introduced, the case is resolved optimally. The agents allow to decentralize the execution of data retrieval and processing; and to distribute it between all participants (Fig. 3c), so an overload of particular nodes, characteristic for most centralized schemas, will be avoided.

The mobile agent technology helped us also in addressing another issue facing the designers of knowledge access-based services – the problem of incoherence of internal data representation among knowledge sources integrated in the system. For users participating in the first schema it does not raise a problem, they visit all sites, one after another, and simply read the data – no matter what is the actual format of data, whether it is a raw HTML, dynamic scripts or binaries or document scans. But for the computer system it is much more difficult. The additional functionality related to the data transformation must be included which arises new questions: where to include this functionality? – put it all to the mediator who will be responsible for all transformations – or to distribute parts of the functionality to each participant of the scenario?; what common data representation to choose for the metalanguage of all transformations?

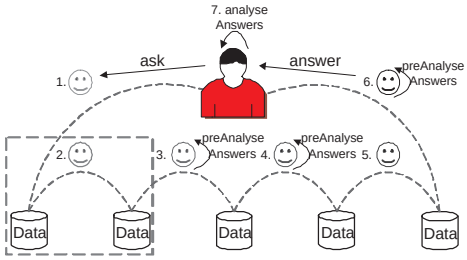
With mobile agents technology we could apply the (inherent in agents) decentralized deployment of transformation functionality (Fig. 3c), so each participant takes a part of transformation work and is responsible for providing the proper knowledge (Figure 4). (Practically this functionality could be implemented as intermediate agent able to communicate in two languages: the common agent communication language [such as FIPA ACL [15]]; and the language [data representation] intrinsic to knowledge source.) The mobile agents provided also the common knowledge representation through the ontologies and content definition languages [15].



(a) basic access schema



(b) mediator based access schema



(c) mobile agent based access schema

Figure 3: Knowledge access schemas: a) basic: a user individually accesses all knowledge sources, b) mediator based: the (3a) task was delegated to a software mediator, c) mobile agent based: a mobile agent visits all subsequent locations.

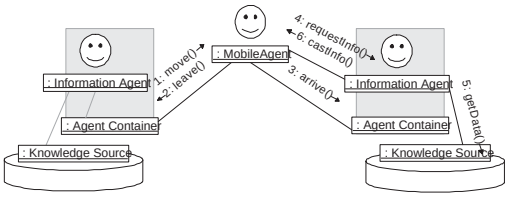


Figure 4: Mobile agent based schema: an agent arrives at a new container and accesses the data from knowledge source with an aid of an information agent. This diagram refers to the part of mobile agent based schema which in Figure 3c is surrounded by a dashed line rectangle.

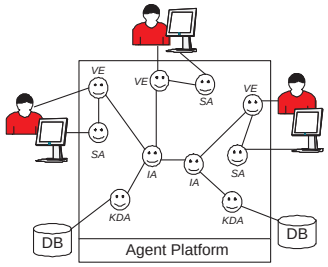


Figure 5: Design of the e-Health counseling service: each user is represented by a Virtual Ego (VE) and can contact it using a Support Agent (SA). For obtaining and processing the knowledge the Knowledge Discovery Agents (KDA) and the Information Agents (IA) are responsible.

Input:

<query> – the query of the user, it may be for example a health problem description, a medicine name or a dietetic question

<service> – the name of a particular service responsible for providing context-specific response for the <query>

Output:

<answer> – the consistent and essential answer for the user's <query>

Steps:

1. User contacts its Virtual Ego through Support Agent
2. Virtual Ego arrives to the User's container and receives the <**query**> from the User
3. Virtual Ego queries the Directory Facilitator for Virtual Egos of <**service**> providers
4. User selects the <**service**> providers according to his/her preferences
5. Virtual Ego leaves the container and using Untraceability Protocol II migrates to the containers where Virtual Egos of <**service**> providers are located
6. On each of the containers user's Virtual Ego contacts <**service**> provider's Virtual Ego of to obtain <answer> for the <**query**>
7. After gathering all <answers> Virtual Ego contacts Knowledge Discovery Agent to process the <answers> [3] to obtain one consistent <**answer**>

Figure 6: The QSA template. To receive an output value in the <answer> variable, the input variables (the <query> and the <service>) should be assigned with particular query and service instances.

Finally we brought in the following conventions (Fig. 5): each user (a patient, a physician, a pharmacist etc) is represented in the e-Health Agent Platform by a Virtual Ego – the software agent permanently present in the system. Information Agents are responsible for providing a requested information from knowledge sources in the unified manner. Support Agents intermediate between users and the system, providing the user interface. The Knowledge Discovery Agents are responsible for extracting any essential knowledge from databases; and eliminating redundancies of information.

3.3 The implementation

The implementation of the service was based on the reusable *Query-Service-Answer (QSA)* schema of information retrieval, which we designed specially for this purpose. The schema was notated into a template (Fig. 6) which facilitates tracking the information flows from the input to the output, and the actors involved in the information retrieval (indicated by underlined nouns). The simplified activity diagram of full anonymous health counseling scenario based on QSA schema is presented on Figure 7.

Based on the schema we implemented the medical service in which the user (later called: the patient) can consult his health problem with doctors supported by pharmaceutical and dietetic knowledge. All users (both the service users and providers) are represented by their Virtual Egos which exist permanently in the agent platform. Users

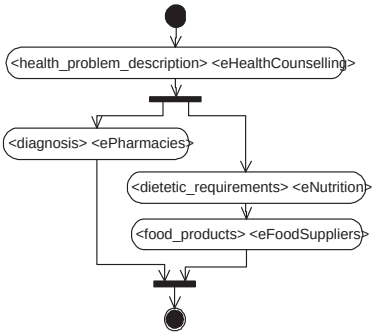


Figure 7: The QSA schema based actions in implementation of the health counseling scenario. Each action represents the use of QSA schema. Instead of action names, the schema input values (subsequently: a <query> and a <service>) are used.

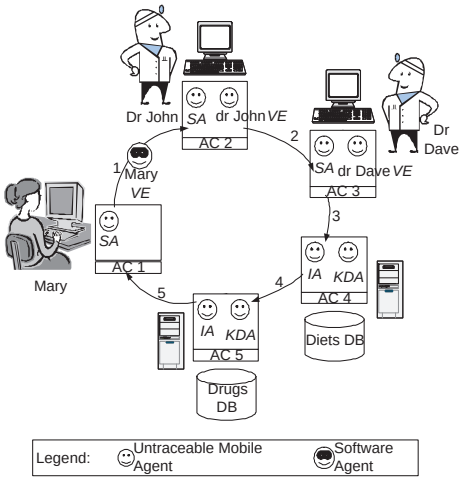


Figure 8: The implemented e-Health counseling service utilizing mobile agents. The patient presents the description of his/her health problem and the untraceable Virtual Ego migrates through subsequent agent containers in order to gather the relevant knowledge.

contact the Virtual Egos with the aid of Support Agents. The Support Agents can be activated from any agent enabled architecture (currently a PC, but since lightweight JADE platform – JADE Leap – is available, in the future this might be done from a mobile phone, a hand-held device etc). The patient describe his/her case and the description is passed to the Virtual Ego which arrived at the user device. Then the Virtual Ego visits containers of doctors and presents the description to the doctors' Virtual Egos. After the doctors gave their counsels the patient's Virtual Ego moves further to contact Knowledge Agents in order to accompany the advices with domain-specific information (dietetic and pharmaceutical). After completing the data, the Virtual Ego returns to the patient's container to present the full counsel. The Figure 8 illustrates this schema.

To protect anonymity of the user, the Virtual Ego was implemented as an untraceable agent utilizing Protocol I. For this we had to simply extend the previously written UntraceableAgent class, which encompasses all untraceability functions for agent; and to activate our untraceability service (a JADE add-on – see Section 2) on all the containers likely to be visited by Virtual Egos. An interested reader can also look to [3] – for further information.

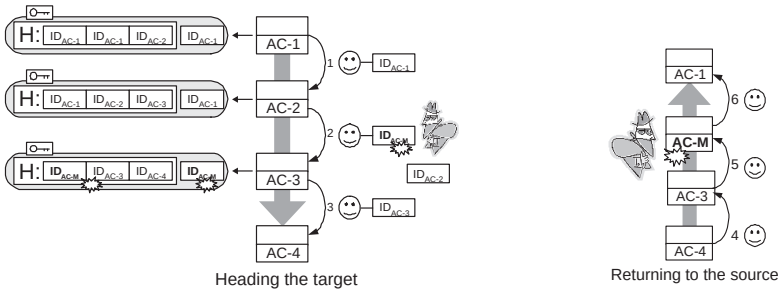
3.4 Results

During the implementation we encountered two major issues. The first is related to the technical limitations of current realization of the JADE agent platform which disallowed us to set up the environments with high numbers of agents[4]. We could launch one container with no more than around 1300 agents before receiving an error related to the memory outage. This number yet decreased when installing more containers. For the environment with 60 containers the error occurred with the initialization of the 212^{th} agent. To make sure that the problem is not related to operational system we investigated it on Windows XP and the Debian distribution of Linux – with the same results on both platforms.

The second issue was related to the fact the JADE Agent Management service does not provide the ID of the previous container when an agent arrives to the new one. This information is crucial for proper implementation of Protocol I (the assumptions A.5 and A.6, see Section 2.1, are not satisfied). Thus we had to incorporate the necessary functionality into agents. This means that an untraceable agent has to carry the identifier of the previous container itself. Unfortunately, this induces a possibility of a new attack. An attacker able to intercept the untraceable agent can replace the container identifier, so the agent will visit the attacker's malicious container on its way back (Fig. 9). This could be avoided if we introduced a secure (inaccessible for the attacker) inter-container communication channel. We think to implement such channel in the future. We plan to underpin the secrecy of the channel with the aid of hardware Trusted Computing Module [21, 22].

Although the impossibility of obtaining the ID of a preceding container in JADE brought in a vulnerability to the implementation of Protocol I, this is not the obstacle

[4]We performed our tests on the Intel®Pentium®4, 3 GHz, 1.5 GB RAM station equipped with Java™2 Runtime Environment, Standard Edition (build 1.4.2_01-b06), Java HotSpot™Client VM (build 1.4.2_01-b06, mixed mode) – which were run with default settings.



(a) The attacker replaces the container ID with the ID of its malicious container

(b) thus the agent visits the malicious platform on its return.

Figure 9: The Agent Container ID replacement attack causing that the agent has to visit the attacker's malicious platform on its way back.

preventing a realization of the anonymous counseling service. Especially because we have already proposed a solution to deal with this limitation.

However the problem with deploying larger agent environments appears to be serious. Real e-Health systems will be much more complex than the study which we performed. They will serve hundreds of thousands if not millions of users, which will require presence of thousands, millions of agents. And after the experience with the feasibility study we have to admit that at the current stage of JADE development [5] the platform is not ready for bearing such large environments. Thus we look forward to the new release of JADE which will deal with the limitation.

On the other hand the experiment showed that our agent based approach is adequate for implementation of complex knowledge processing related scenarios (since the majority of medical services is based on the same same schema – in order to acquire a complete and essential information, various information sources must be accessed – the proposal we made for the counseling service might be applied to other medical services). The FIPA-compliant JADE middleware provides a homogeneous infrastructure for knowledge management with the common knowledge representation (agent content languages and ontologies) and the knowledge interchange protocols. Moreover JADE as a service-focused environment supports flexibility and scalability of applications built on its base – an introduction of a new functionality does not require any code modification, instead a new agent is added to the environment and its functionality is announced in the services directory.

Furthermore the decentralized knowledge access schema based on mobile agents exposes several advantages over the traditional mediator-centric approaches. Implementing the schema will prevent overloading arbitrary nodes because the knowledge processing workload described in Section 3.2 will be equally deployed between all system functional components. It will also reduce the knowledge translation related complexity of nodes since each node will have to implement only the functionality of translation between the internal language and the common language. Finally, the re-

[5] This is worth to remind that despite the fact that the first fully-functional release of JADE was published few years ago, the development is still being conducted and new, improved versions are released.

duction of nodes' complexity and the balanced workload deployment should result in better performance of the whole platform; and the decentralization will prevent from several Denial of Service attacks, at least making them more difficult to perform.

4 Conclusions

We have presented our solution for anonymous access of IT services and the anonymous e-Health counseling service which takes advantage of the solution.

The results of the experiment showed that for the current state of the JADE development, an agent based implementation of complex medical system providing services such as the anonymous e-Health counseling to large societies of users, appears to be infeasible. For the moment, JADE (which anyway, according to our study [see [16]], seems to be the best of open source agent environments) allows one only to set up environments with the limited number of agents.

On the other hand, despite that we are not able to implement a complex agent based e-Health platform because of the JADE limitations, the experiment proved that the agent based approach fits well implementation of knowledge processing services, offering great flexibility, scalability and reach support for knowledge processing and representation.

We have also showed feasibility of the realization of anonymous e-Health counseling scenario based on our untraceability protocol. Moreover with the help of the study we could track discrepancies between the theoretical specification of the protocol and its actual realization in the real environment. These discrepancies stem from the fact that in reality not all assumptions stated for the protocols are satisfied; and (as in our case) in consequence may result in reduction of security of the protocol. During the investigation we found such a bottleneck entailing vulnerability to insertion attacks. We have proposed a solution to avoid these attacks, which we are going to implement in the future.

References

[1] Rafał Leszczyna and Janusz Górski. Untraceability of mobile agents. *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*, 3:1233–1234, July 2005.

[2] Rafał Leszczyna and Janusz Górski. Untraceability of mobile agents. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, December 2004.

[3] Rafał Leszczyna and Janusz Górski. Performance analysis of untraceability protocols for mobile agents. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, March 2005.

[4] National Institute of Standards and Technology (NIST). *Common Criteria for Information Technology Security Evaluation - Part 2: Security Funtional Requirements*. U.S. Government Printing Office, 1998.

[5] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. draft v0.21. 2004.

[6] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 2004. http://www.cs.cornell.edu/people/oneill/papers/jcs_halpern_oneill.pdf.

[7] Ceki Gulcu and Gene Tsudik. Mixing email with Babel. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 2. IEEE Computer Society, 1996.

[8] EU IST-2002-507591 PRIME. Requirements version 0 part 3: Application requirements.

[9] Douglas Harper. Online etymology dictionary. Website. http://www.etymonline.com/ (last access: May 4, 2005).

[10] Merriam-webster online. Website. http://www.m-w.com/ (last access: May 4, 2005).

[11] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96)*, volume 1193, Berlin, Germany, 1996. Springer-Verlag.

[12] Richard Murch and Tony Johnson. *Intelligent software agents*. Prentice Hall PTR, 1999.

[13] M. Luck, P. McBurney, and C. Preist. *Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing)*. AgentLink, January 2003.

[14] W. Farmer, J. Guttmann, and V. Swarup. Security for mobile agents: Issues and requirements, 1996. gunther.smeal.psu.edu/farmer96security.html.

[15] Foundation for Intelligent Physical Agents (FIPA). Website. http://www.fipa.org/.

[16] Rafał Leszczyna. Evaluation of agent platforms. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Ispra, Italy, June 2004.

[17] Jade-board. Website. http://jade.talab.com/.

[18] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *JADE - A White Paper*, September 2003.

[19] Dirk Westhoff, II Markus Schneider, Claus Unger, and Firoz Kaderali. Protecting a mobile agent's route against collusions. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 215–225. Springer-Verlag, 2000.

[20] Matthias Enzmann, Thomas Kunz, and Markus Schneider. Using mobile agents for privacy amplification in the trade with tangible goods. In *6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*, volume IV, Orlando, Florida, USA, July 2002.

[21] Trusted Computing Group. Website. `https://www.trustedcomputinggroup.org`.

[22] Trusted Computing Group. TCG specification architecture overview. specification revision 1.2, April 2004. Available at `https://www.trustedcomputinggroup.org`.