Anonymity Architecture for Mobile Agents

Rafał Leszczyna[1] and Janusz Górski[2]

[1] European Commission, Joint Research Centre,
Via Enrico Fermi, 21020 Ispra (VA), Italy
[2] Gdansk University of Technology,
Narutowicza 11/12, Gdańsk, Poland

13 April 2006

Abstract

We imagine the future with ubiquitous software agents, intelligently assisting people in their daily lives. We think that very important property of these agents is their ability to autonomously move from one device to another (mobility). Mobile agents proved many advantages but the major obstacle to their widespread adoption are problems related to their security. Thus continuous work on improving security of the agents must be conducted. This is where we concentrated our research effort. As the result we can propose an architecture supporting anonymity of agent owners. In this paper we concentrate on its core module – the infrastructure of untraceability protocol, which we implemented as a JADE add-on and shared with JADE community.

1 Introduction

We envisage the future with ambient intelligence implemented by ubiquitous software agents, working on behalf of people, assisting them in their daily lives. The notions of *intelligent*, *adaptive* and *learning*, *reasoning*, *autonomous*, *proactive*, *social* etc. *agents* so far have been reserved for describing humans [1, 2]. One of these characteristics is *autonomous mobility* – the agents' ability to autonomously change their location in a network. *Mobile agents* (MA) can be delegated to various destinations, and there is no longer a need for a fixed connection after their arrival. Mobile agents work straight at local machines, using their resources. The MA paradigm presents many other advantages (bandwidth conservation, latency reduction, etc [3, 4, 5]) and thus it gained high popularity in the second half of the nineties. Unfortunately, when the first difficulties were discovered, related to agents' mobile nature (when moving, agents are fully dependant on the containers at which they arrive), the interest in the area started to fade away. The primary obstacle is the problem of MA security [6, 7]. To open a way to wide development of useful and innovative solutions based on

MA, the security issues must be resolved and adequate security solutions must be provided [7]. This is the motivation of our work, dedicated to protection of MA.

In the progress of this research, we discovered that *Mobile Agent Systems* (MAS) have some characteristics which may facilitate protection of users' privacy [8]. Based on the observation we proposed a security protocol aiming at anonymity (precisely – untraceability) of agent users [9, 8]. We evaluated its performance [10] and performed security studies [8, 11]. We implemented the protocol [12] as an open source package (the package is available at `{http://jade.tilab.com/community-3rdpartysw.htm#Untraceability}`) and deployed it in an e-Health anonymous counselling scenario [13].

In this paper we present an architecture aiming at protecting anonymity of agents' owners. It core part is formed by our untraceability protocol.

1.1 Anonymity in MAS

Anonymity is the property of users enabling them to use resources or services (the *items of interest*) without disclosing their identity [14].

Anonymity plays a crucial role in many activities conducted in the Internet. For example users may be reluctant to engage in web-browsing, message-sending and file-sharing, unless they receive guarantees that their anonymity will be protected to some reasonable degree [15]. [16] 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.

There are two ways to disclose identity of a mobile agent's owner (see Figure 1): through reading the agent's data or through *traffic analysis* (TA).

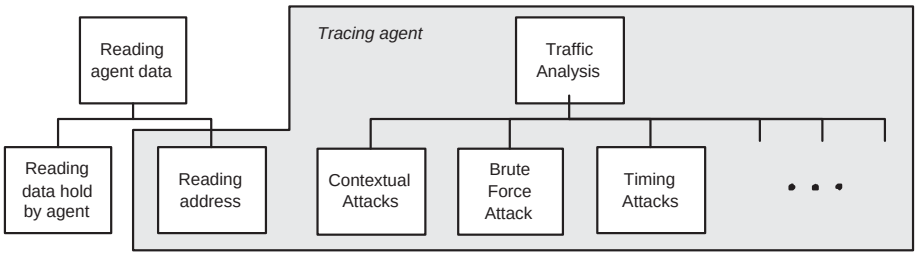

Figure 1: Attacks against anonymity of mobile agents.

In regard to agent's data, the data *of the agent* (describing the agent) and the data *held by agent*, should be distinguished, because they differ significantly in ways of their protection.

The data of the agent are: the agent's execution code, its state, and *operational data*. Operational data are the data required for basic performance of

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

mobile agent and they are at the agent's full disposition through its whole (or the major part of) lifecycle. An example of such data is the address of *base station* (or *base* – the container from which the agent started its migration) needed by the agent while returning to the base station. Since the operational data must be of high availability of the agent, they are practically continuously exposed: they can not be protected using any direct encryption schema.

The data held by an agent are the data necessary to realise its concrete task. These data have to be accessible at the agent's destination. If the agent carries data which identify the agent's owner (e.g. the unique identifier of the owner or its indicative description), the owner's identity can be easily disclosed. To obscure the data of this type it is sufficient to encrypt them with the destination's public key.

The data targets of an attacker aiming at anonymity of the agent's owner (anonymity attacker) are *the address data*, and *the data held by the agent*.

We assume that all agents are based on the same code and that they can be represented by the same state machine. In other words, neither the code nor the state are characteristic for a particular agent. If this assumption is not satisfied, it opens a way to traffic analysis attacks, which we described in [11]. *Traffic analysis* (TA), is a process of intercepting and examining agents in order to deduce information from their patterns of communication.

Reading the agent's address data or performing TA are the activities of *tracing* the agent. They may lead an attacker to infer the identity of the agent's owner. The state of protection against this type of inference is called *untraceability* [17].

1.2 Solutions for Untraceability

Most solutions to message untraceability are founded on the idea of *mix* introduced by Chaum in 1981 [18]. Mix is a proxy, located between message sender and receiver to obfuscate their addresses using cryptographic methods [18]. Chaum considered application of one multiple mixes (so called *cascades*). Since mixes can form different constellations, such as networks of cascades, multiple-duplicated cascades, tree structures, networks with restricted choice of patches and others [19] and they can offer different functionalities, many different solutions have been proposed. Examples include: Non Disclosure Method (NDM) [20], BABEL [16], Onion Routing [21], Crowds [22], raw remailers, Cypherpunks (Type I remailers), Mixmasters (Type II remailers) and others [23]. These solutions differ in complexity and provide varying levels of protection.

Another stream of research is related to Chaum's DC (Dining Cryptographers) network [24]. However, due to the complexity of proposed solutions it still remains in the experimental phase [25, 26]. Involvement of all partners in transmission of every message (by forecasting and receiving packets to/from the others) and the need to share some secret information between the partners make the proposed protocols very resource consuming.

The untraceability solutions dedicated to message based communication can not be directly applied to MAS because they do not respect agents' autonomy. The route has to be determined before launching an agent and the spontaneous route selection during migration is severely restricted, so the advantages agents which are very useful in some applications (targeting dynamic deployment, load balancing etc) can not be exploited [3]. As far as we know very few solutions for MAS untraceability have been proposed so far [27, 28]. They are based on the conception of onion routing [21]. In [9, 8] we have proposed an untraceability protocol which does not impose such limitations.

2 Anonymity Architecture

As it was already stated in Section 1.1, anonymity of the agent's owner can be compromised in two ways: by reading the data held by the agent and by tracing the agent. The proposed architecture aims at *preventing agents from being followed without imposing additional constraints on the agents' autonomy* .

This objective is formulated in terms of two subobjectives:

1. The architecture should allow agent owners to hide (make unreadable to unauthorised parties) the address of the agent's base station. This obfuscation should not constrain autonomy of the agent in planning and following its route. Despite the obfuscation the agent should be able to come back to the base station.

2. The architecture should facilitate protection of agents from traffic analysis attacks performed by various types of attackers.

2.1 Model of Adversary

The following types of attackers are considered:

Internal / external – the adversary can compromise communication media (*external*) and/or mix nodes, recipients and senders (*internal*) [29].

Omnipresent / k-listening – the adversary may succeed in attacking all nodes (*omnipresent*), or `k` of them (*k-listening* [30]). In particular only one node may be compromised *single adversary* [31].

Active / passive – an active adversary can arbitrarily modify the computations and messages (by adding and deleting) whereas a passive adversary can only listen [29].

Static / adaptive – static adversaries choose the resources they compromise before the protocol starts and are not able to change them once the protocol has started. Adaptive adversaries are allowed to change the resources they control while the protocol is being executed [29, 32]. They can, for example, 'follow' messages [29].

Hybrid – hybrids and alliances of attackers may occur, such as external-active or colluding internal and external. Syverson et al [31], for example, distinguish between *multiple adversary* and *roving adversary*, which are subsequently: k-listening static and k-listening adaptive adversary.

2.2 Network Model

The architecture is dedicated for any agent platform complying to the FIPA specification [33]. In this environment, due to its complexity, various attack scenarios aiming at compromising anonymity of agent owners can be imagined. In particular the attacks which target message communication between platform users. The architecture focuses on aspects of agent mobility (migration among containers) and aims at protects against following the agent. The question of tracking agents through observing messages sent by and to them (for instance during communication with other agents, service queries and requests etc) are not considered. To ensure message untraceability one of the methods mentioned in Section 1.1 could be applied.

We assume that in normal situation (when neither the whole platform nor its part are being compromised) the following conditions are satisfied:

A.1. Third parties (other agents, users) are not informed about the presence of other agents if the latter do not want to do so. It is impossible to introduce any agents aiming at observing and following other agents.

A.2. Each container is well isolated, so it is impossible to learn its state from outside.

A.3. Each container participating in untraceable migration owns an individual symmetric key.

A.4. Each container is able to provide an agent with the identifier of the previous container visited by the agent. The identifier is destroyed after the agent leaves out.

A.5. This identifier is available to the agent.

2.3 The Architecture

The anonymity architecture is composed of two modules (see Figure 2): Module I: Untraceability Protocol Infrastructure and Module II: Additional Anti Traffic-Analysis Support.

The Untraceability Protocol Infrastructure module is responsible for obfuscating the address of agent base. It forms a core of the anonymity architecture, which is embraced by the elements of the second module. This core module implements the untraceability protocol described in the next section.

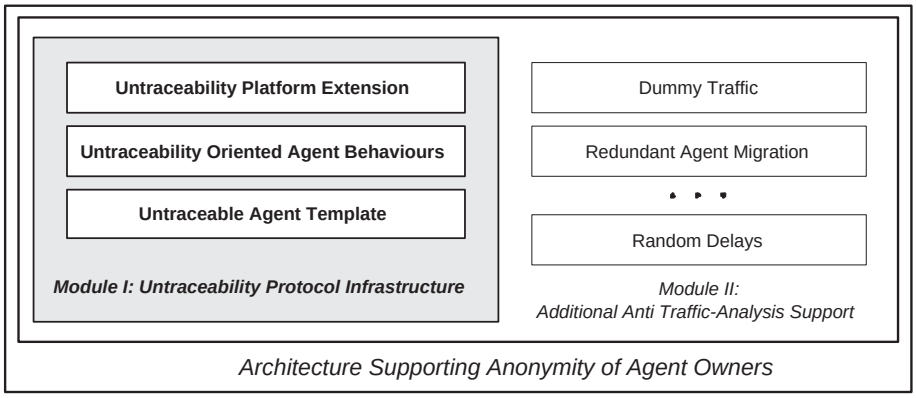

Figure 2: The anonymity architecture.

The Additional Anti Traffic-Analysis Support module aims at protecting agents from TA attacks. The module components can be the methods of protection against TA, which we proposed in [11]. The selection of a particular components is delegated to agent platform administrators or designers who should take into account that each component (dummy traffic, redundant migration of agents etc) introduces some additional overhead to the platform [11].

2.4 Protocol Description

In common scenarios, an agent covers its route and come back to its base. To provide for this, the identifier of the base is explicitly stored in agent's state. The proposed protocol aims at obscuring this identifier.

In brief, the idea of the protocol is that at each visited container an agent asks the container to encrypt the identifier of the previous container using the current container's secret key. Then the encrypted identifier is put into the LIFO queue stored in the agent's data (see Figure 3). After achieving its goal, when the agent wishes to come back to its base container, it uses the queue to find its way back. Down the route back, the identifiers are subsequently decrypted by the containers using their secret keys. More information about the protocol can be found in [9, 8, 13, 10, 11].

2.5 Protocol Implementation: Untraceability Protocol Infrastructure

The infrastructure implementing our untraceability protocol is composed of the following elements (see Figure 4):

1. An extension to the agent platform providing untraceability service.
2. A set of untraceability-oriented agent behaviours.

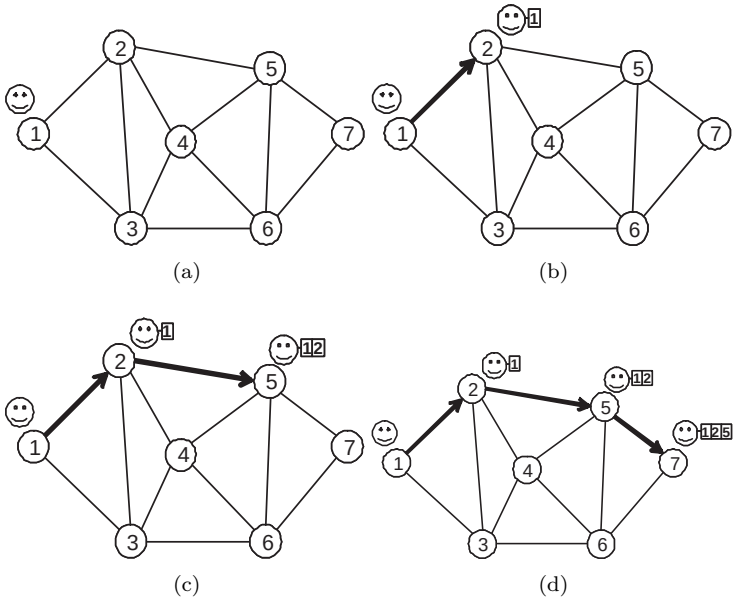

Figure 3: Untraceability Protocol: When the agent migrates, on each subsequent container the identifier of previous container is encrypted and stored into the agent's data. This list of encrypted identifiers will allow the agent to come back to the base container.

3. A template of an untraceable agent.

We defined basic behaviours[2] and a template of untraceable agent which allow to realise full scenario of untraceable migration which most commonly incorporates the following steps:

1. The agent determines which containers are available in the agent platform and based on this it determines the route which it will cover.
2. The agent leaves the base container and follows subsequent containers on the route up to the destination.
3. The agent realises its task at the destination.
4. The agent returns to the base container.

CreateRouteToAgentBehaviour and CreateRouteToDestinationBehaviour – are behaviours of creating a migration path to a predestined container or to another

[2]Just to remind – an agent's *behaviour* is a set of actions performed in order to achieve the goal. It represents a task that an agent can perform [34].

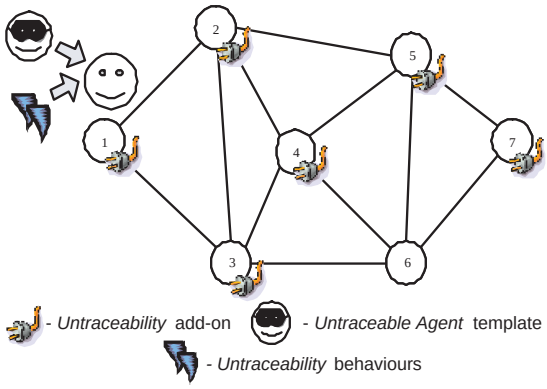

Figure 4: The anonymity architecture.

agent. GetAvailableLocationsBehaviour – is a behaviour of finding which containers are available. And finally the behaviours GoAheadBehaviour and ComeBackBehaviour allow the agent to migrate untraceably towards and from the destination.

The agent platform extension is designed as a plug-in (add-on) for agent platform containers. This form of implementation gives freedom in choosing which containers should provide the untraceability function (Figure 4). When agent arrives at a container, it can check whether the service is available and if so, then the agent may take advantage of it. The agent calls the service and provides as an input the identifier of the container which it has chosen as a next on its route. Based on this, the plug-in compiles the identifier into a string of the three identifiers: the previous–, the current– and the next– container. Then it computes a MD5 message digest of the string, and accompanies with a generated nonce and the identifier of the previous container. Finally it is all encrypted using AES algorithm and returned to the agent - the details can be found in [8, 9]. To provide this functionality each installed plug-in has to maintain an unique secret key for the AES algorithm.

The protocol has been implemented o as a service add-on for JADE [12]. The process of developing untraceable agents involves employing the UntraceableAgent class combined with the predetermined migration behaviours – the GoAheadBehaviour and ComeBackBehaviour – available in the behaviours subpackage). It is important to have the agent route stored in the myUnprotectedRoute field of the agent's class. To obtain the route the CreateRouteToAgentBehaviour or CreateRouteToDestinationBehaviour may be used. To allow the Untraceability Service must be registered at each container participating in agent's untraceable migration.

3 Case Study: Untraceable Agents in e-Health

To validate the proposed architecture we developed an e-Health counseling service which 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. We have chosen e-Health application because here user's anonymity has high priority. The service itself belongs to the class of knowledge based services and thus the anonymity solutions designed for it can easily extended to other services belonging to this class. The service can be exemplified by the following scenario:

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 should 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.

In the implementation all users (both the service users and providers) are represented by their Virtual Egos which exist permanently in the agent platform. The Virtual Egos of patients are implemented based on our UntraceableAgent template (as classes extending the UntraceableAgent class, see 2.5). Users contact them by 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). Patients describe their cases and the descriptions are passed to the untraceable Virtual Egos which arrived at the users' devices. Then the Virtual Egos untraceably migrate to containers of doctors, taking advantage of untraceability-oriented behaviours. There they present the health problem descriptions to the doctors' Virtual Egos. The doctors give their counsels, and the patients' Virtual Egos move further to contact Knowledge Agents in order to accompany the advices with domain-specific information (dietetic and pharmaceutical). After completing the data, the Virtual Egos return to the patients' containers to present the full counsel. The Figure 5 illustrates this schema. On all containers likely to be visited by patients' Virtual Egos, the Untraceability add-on is activated.

The case study demonstrated the applicability of the proposed architecture and experimentally confirmed the theoretical result [10] that for a route of the length n leading to a destination, an untraceable agent must cover the route of the length 2n. We could also track some discrepancies between the theoretical

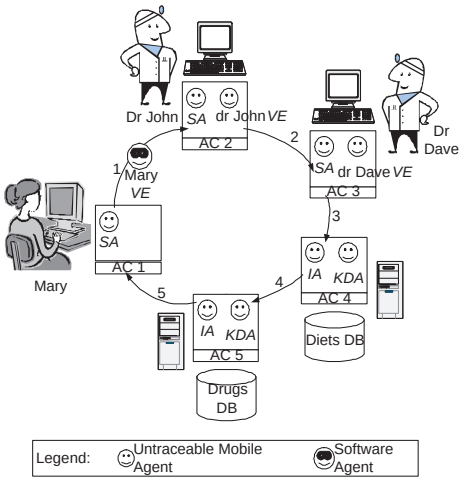

Figure 5: The implemented e-Health counseling service based on the Untraceability Protocol Infrastructure. The patient presents the description of his/her health problem and the untraceable Virtual Ego migrates through subsequent agent containers in order to gather the relevant knowledge.

specification of the protocol and its actual realization in the real environment. More details can be found in [13].

4 Summary

We proposed an anonymity architecture for mobile agent systems. The architecture is composed of two modules, one responsible for obscuring agents' base addresses and the second, protecting agents from traffic analysis. The first – the Untraceability Protocol Infrastructure – the core module of the architecture, implements specification of our untraceability protocol. Security and performance analysis of this protocol was presented elsewhere [8, 11, 10].

The infrastructure consists of three components: an extension to agent platform providing untraceability service, a set of untraceability-oriented agent behaviours and a template of an untraceable agent. In this form the infrastructure has been implemented and published [12]. It has been also successfully applied to an eHealth case study. The case study demonstrated the applicability and usefulness of the proposed architecture.

The architecture offers a solution for protecting mobile agents and we believe that it is a can contribute to wider popularisation of mobile agents. Although we have implemented the infrastructure for JADE, its design can be easily adapted to other agent platforms.

References

[1] 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. Available at `citeseer.ist.psu.edu/franklin96is.html`.

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

[3] Robert S. Gray, David Kotz, George Cybenko, and Daniela Rus. Mobile agents: Motivations and state-of-the-art systems. Technical Report TR2000-365, Dartmouth College, Hanover, NH, 2000.

[4] Davis Chess, Benjamin Grosof, Colin Harrison, David Levine, Colin Parris, and Gene Tsudik. Itinerant Agents for Mobile Computing. *IEEE Personal Communications*, 2(5):34–49, 1995. Available at `citeseer.ist.psu.edu/article/chess95itinerant.html`.

[5] Constantinos Spyrou, George Samaras, Evaggelia Pitoura, and Paraskevas Evripidou. Mobile agents for wireless computing: The convergence of wireless computational models with mobile-agent technologies. *Mob. Netw. Appl.*, 9(5):517–528, October 2004. Available at `http://doi.acm.org/10.1145/1027347.1027354`.

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

[7] W. Jansen and T. Karygiannis. Nist special publication 800-19 - mobile agent security, 2000. Available at `citeseer.ist.psu.edu/jansen00nist.html`.

[8] 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.

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

[10] Rafał Leszczyna and Janusz Górski. Performance analysis of untraceability protocols for mobile agents using an adaptable framework. Accepted for 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06), Future University-Hakodate, 8 – 12 May 2006, October 2005.

[11] Rafał Leszczyna and Janusz Górski. An untraceability protocol for mobile agents and its enhanced security study. Accepted for 15th EICAR Annual Conference, Hamburg-Germany, 29 April – 2 May 2006, March 2006.

[12] Rafał Leszczyna. *Untraceability I Add-on for JADE.* European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Via Enrico Fermi 1, Ispra, Italy, 1 edition, September 2005. The add-on is available to download at `http://jade.tilab.com/community-3rdpartysw.htm#Untraceability` (last access: December 16, 2005).

[13] Rafał Leszczyna. The solution for anonymous access of it services and its application to e-health counselling. In *Proceedings of the 1st 2005 IEEE International Conference on Technologies for Homeland Security and Safety (TEHOSS '05)*, volume 1, pages 161–170, Gdańsk, Poland, September 2005. Gdańsk University of Technology.

[14] 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.

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

[16] Ceki Gülcü 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.

[17] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, unobservability, pseudonymity, and identity management a consolidated proposal for terminology (version v0.25). Website, December 2005. Available at `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml` (last access: February 15, 2006).

[18] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[19] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000. Available at `http://www.tik.ee.ethz.ch/~weiler/lehre/netsec/Unterlagen/anon/disadvan%tages_berthold.pdf`.

[20] A. Fasbender, D. Kesdogan, and O. Kubitz. Variable and scalable security: Protection of location information in mobile ip, 1996. Available at `citeseer.ist.psu.edu/fasbender96variable.html`.

[21] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.

[22] Michael K. Reiter and Aviel D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–48, 1999.

[23] Anonymity and privacy on the internet. Website. Available at `http://www.stack.nl/~galactus/remailers/`.

[24] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[25] Experimental crypto software: Dcchat. Website. `http://www.geocities.com/cryptosw/dcchat.html`.

[26] Heiko Stamer. Dining cryptographers networks. Master's thesis, The Institute of Computer Science, Leipzig University, May 2003.

[27] 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.

[28] 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.

[29] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 10–29, New York, NY, USA, 2001. Springer-Verlag New York, Inc. Available at `citeseer.ist.psu.edu/454354.html`.

[30] Shlomi Dolev and Rafail Ostrobsky. Xor-trees for efficient anonymous multicast and reception. *ACM Trans. Inf. Syst. Secur.*, 3(2):63–84, 2000. Available at `www.cs.ucla.edu/~rafail/PUBLIC/31.ps`.

[31] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 96–114, New York, NY, USA, July 2000. Springer-Verlag New York, Inc.

[32] Yehuda Lindell. Foundations of cryptography 89-856. Electronic document, April 2006. First draft of lecture notes written for a graduate course in the foundations of cryptography at Bar-Ilan University, Israel. Available at `http://www.cs.biu.ac.il/~lindell/89-856/complete-89-856.pdf` (last access: April 3, 2006).

[33] Foundation for Intelligent Physical Agents (FIPA). Fipa abstract architecture specification, 2002.

[34] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *Jade programmers guide*, February 2003.