EUROPEAN COMMISSION
DIRECTORATE-GENERAL
Joint Research Centre

Privacy in Mobile Agent Systems: Untraceability

Rafał Leszczyna

Institute for the Protection and Security of the Citizen

2007

EUR 22485 EN

The mission of IPSC is to provide research-based, systems-oriented support to EU policies so as to protect the citizen against economic and technological risk. The Institute also continues to maintain and develop its expertise in information, communication, space and engineering technologies in support of its mission. The strong cross-fertilisation between its nuclear and non-nuclear activities strengthens the expertise it can bring to the benefit of customers in both domains.

European Commission
Directorate-General Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information
Address: Via E. Fermi, 1 – 21020 Ispra (VA) - Italy
E-mail: rafal.leszczyna@jrc.it
Tel.: +39.0332.786715
Fax: +39.0332.789576

http://www.jrc.cec.eu.int

EUR 22485 EN

ISSN 1018-5593

Luxembourg: Office for Official Publications of the European Communities

Printed in Italy

Contents

Acknowledgement

I would like to thank Marcelo Masera for valuable remarks and suggestions concerning the content of this report.

Introduction

Agent based Internet environments are an interesting alternative to existing approaches of building software systems. The enabling feature of agents is that they allow software development based on the abstraction (a 'metaphor', see [70]) of elements of the real world. In other words, they allow building software systems, which work as human societies, in which members share products and services, cooperate or compete with each other. Organisational, behavioural and functional models etc applied into the systems can be 'imported' from the real world[1].

Other, most commonly cited encountered advantages of agents are [70]:

- Agents improve system deployment and facilitate system integration and coalition formation.
- Agents allow a broad range of users to access a broad range of services offered by various organisations (some of them may be even in competition with each other).

Moreover agents present a number of technological advantages such as bandwidth conservation, faster task completion, latency reduction, disconnected operation, load balancing and dynamic deployment [46].

The growing interest in agent technologies in the European Union was expressed through the foundation of the Coordination Action for Agent-Based Computing, funded under the European Commission's Sixth Framework Programme (FP6). The action, called AgentLink III is run by the Information Society Technologies (IST) programme. The long-term goal of AgentLink is to put Europe at the leading edge of international competitiveness in this increasingly important area.

According to AgentLink 'Roadmap for Agent Based Computing' [70] agent-based systems are perceived as 'one of the most vibrant and important areas of research and development to have emerged in information technology in recent years, underpinning many aspects of broader information society technologies.'

However, with the emergence of the new paradigm, came also new challenges. One of them is that agent environments, especially those which allow for mobility of agents, are much more difficult to protect from intruders than conventional systems. Agent environments still lack sufficient and effective solutions to assure their security

[1] Some case studies illustrating this software design and development approach are [6–9, 74, 75, 80, 108].

[31,54,70]. The problem which till now has not been addressed sufficiently in agent-based systems is privacy, and particularly the anonymity of agent users.

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 [50]. Ceki Gülcü and Gene Tsudik [49] describe four categories of internet applications where anonymity is required[2]: discussion of sensitive and personal issues, information searches, freedom of speech in intolerant environments and polling/surveying.

Although anonymity was studied extensively for traditional message-based communication for which during the past twenty five years various techniques have been proposed, for agent systems this problem has never been directly addressed. A possible reason for this could be that most techniques designed for traditional networks are not directly applicable to MAS.

The research presented in this report aimed at filling this gap. As the first step of the research an extensive study of agents and in particular of agent security was performed. A summary of this study is presented in Chapter 1.

In the progress of this research, it was discovered that mobile agent systems, though more difficult to protect in general, nevertheless have some characteristics which may facilitate the protection of users' privacy, and primarily that in MAS each container could be set up as an anonymiser (a mix). Based on this observation an untraceability protocol was proposed which has the advantage that it doesn't impose restrictions on agent's autonomy. The protocol was implemented [63], its performance [65] was evaluated and security studies [64,66] were performed. It was also applied to an e-Health anonymous counselling scenario [62].

This report summarises results of studies aiming at the identification of threats to privacy in agent-based systems and the methods of their protection.

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

Chapter 1

Mobile Agent Systems

1.1 Brief Historical Outline

Agents integrate concepts from three research areas: Artificial Intelligence (AI), Robotics, Distributed Computing.

In the fifties, in the relatively young fields of Robotics[1], the use of computers allowed researchers to build robots which behaved *autonomously*. Robots *act on behalf of people* and aim to replace them in performing difficult tasks. Constructions such as Shakey [84] were built. Shakey, developed in Stanford Research Institute's (SRI's) pioneering Artificial Intelligence Center from 1966 to 1972, is called to be the world's first mobile robot able to reason about its surroundings [100].

In parallel, AI researched possibilities of conducting a natural language dialogue with a machine. Significant developments in Natural Language Processing (NLP) were made [14] between the sixties and the seventies. In 1966 Joseph Weizenbaum presented Eliza [116], a program which could engage in a conversation with a user[2]. Some people call this program a first *software agent* [30]. Another influential program, Shrdlu (1968) [122], allowed a person to have a more 'convincing' (than with Eliza) conversation with a simulated robot.

While Eliza just 'pretended' to be intelligent by using several language analysis 'tricks' [116], Shrdlu had more genuine understanding resulting from possessing a knowledge base. Research on Knowledge Representation (KR) has been performed since beginning of AI, and has laid down a basis for planning and reasoning. Many KR methods were tried in the seventies and early eighties, such as heuristic question-answering, neural networks, theorem proving, and expert systems, with varying success [119].

In 1986 Marvin Minsky published 'Society of Mind' [77] which brought in the notion of *Multi-Agent Systems*. His vision was that a complex system such as the

[1] The beginning of Robotics is dated at 1920, when Czechoslovakian writer Karel Čapek introduced the word *robot* in the play 'R.U.R. – Rossum's Universal Robots'. The word comes from the Czech *robota*, which means tedious labor [109].

[2] A Java applet faithfully recreating the original Eliza can be found at `http://jerz.setonhill.edu/if/canon/eliza.htm`

human mind should be understood as a collection of relatively simple agents, each of which was a specialist in certain narrow domains. Through structures called K-lines, agents would activate each other whenever their context became relevant [30].

Minsky's concept was one of the first steps towards *cooperation and interaction* of agents. These issues were studied scrupulously later and as a result, several *agent communication languages* were defined in the nineties. The DARPA Knowledge Sharing Initiative produced mechanisms for sharing knowledge, in particular KIF (Knowledge Interchange Format [38]), KQML (Knowledge Query and Manipulation Language [33]) and DAML [24] for modeling ontologies. These have provided the first frameworks for interoperability of agents in heterogeneous environments [30]. Also other standardisation bodies such as W3C (World Wide Web Consortium) [112] have developed specifications [113].

Also the nineties was the time were the Internet boom occurred. This boom created a strong incentive for the development of *intelligent agents*. The Internet, with its focus on interactive technologies – browsable web sites and email was designed for human and not automated processing. There was a demand for such automation so an automated counterpart to the interactive side of the Web was needed.

In parallel to AI, Distributed Computing worked on new programming paradigms: Code Distribution and Code Mobility. One of the first attempts to make code mobile was made in the sixties with introduction of the Job Control Language (JCL) [16]. JCL enabled minicomputers to submit batch jobs to mainframes using a remote entry system (RES). Another direct approach to code mobility is the `rsh` (remote shell) command that wass introduced by 4.2BSD UNIX in 1984 [60]. However, the real milestone was reached in the nineties with definition of scripting languages [120]. This opened up the way to implementations on a wider scale.

The most well-known script language was General Magic's Telescript [118]. Telescript, a communications-oriented programming language, facilitated creating network-independent intelligent agents and distributed applications. It is said that Telescript aimed to do for cross-platform, network-independent messaging what PostScript did for cross-platform, device-independent documents. It was with Telescript that the term *mobile agent* was used for the first time.

In the second half of the nineties mobile agents reached the peak of their popularity. After the introduction of Telescript many mobile agent systems followed, most implemented in Java, which already supported mobile code, and in other scripting languages, such as Tcl/Tk or Python. Unfortunately, few systems actually deployed them in an industrial setting. Eventually despite the fact that mobile agents raised considerable interest in the research community (Agent Tcl [44], Tacoma [55], and Mole [102], for example) and in industry (Aglets [59], Concordia [123], Jumping Beans [1] etc), agents were not widely deployed. One of reasons for this situation was security issues. It is extremely difficult to protect mobile code [31, 54]. This problem is further described in Sections 1.3 and 1.4.

The nineties was the time when most of the previously described concepts converged. This is apparent from the opening of standardisation processes[3] Agent stan-

[3]Another manifestation was the unification of three major agent conferences into one. AA-

dardisation has progressed simultaneously in three areas: Distributed Computing, AI and as already mentioned in agent communication languages.

In Distributed Computing, the Object Management Group (OMG) [85] introduced the first standard, called Mobile Agent Facility (MAF), in 1995. The standard promoted interoperability among agent platforms and in 1988 was converted into Mobile Agent System Interoperability Facility (MASIF) [86] under the influence of a joint submission by IBM, General Magic, The Open Group and GMD FOKUS. The MASIF standard (the latest version was published in 2000) focuses on agent migration and omits any mention of agent communication.

The gap was filled by the Foundation for Intelligent Physical Agents (FIPA) [34], which represents the AI community. FIPA was formed in 1996 to produce software standards for heterogenous and interacting agents and agent-based systems. FIPA have published specifications since 1997 and today there are twenty three standard specifications describing different aspects of agent technology, for example, agent communication, agent management, agent message transport, agent abstract architecture and agent applications. FIPA specifications focus on intelligent agent communication via content languages, adopting the agent communication paradigm. They adopt AI concepts related to knowledge representation, and interchange, aiming at cooperative work of agents. Knowledge representation allows reasoning and is a foundation for autonomy. FIPA 2000 specifications also discuss the aspect of agent mobility.

Work on MASIF finished with the latest release in 2000 and was not continued. On the other hand FIPA is still active and supports and promotes its standards and agent technologies. This and other factors means that emerging agent systems tend to comply to FIPA rather than to MASIF specifications. One such platform is JADE [10, 106]. Nowadays JADE tends to be the most commonly used platform in agent research (see [61]).

After the boom in agents, work on them is performed in a more regular manner.

1.2 Terminology

The fact that software agents were investigated via different research paths, implies that the term *software agents* is defined in different ways. AI research emphasizes intelligence, Distributed Computing research emphasises mobility, etc. The problem escalated to such an extent that it was even reflected in literature (e.g. the famous paper of Franklin and Graesser [36]). There were also attempts to define agents using formal methods [69]. But even today there is no common definition.

Etymologically, the word 'agent', originates from the Latin *agere* and was brought to English in the 1471. *Agere* means *to set in motion*, *drive*, *lead*, *conduct*, *act*, *do* [51] [73]. Thus, an *agent* is someone (something) who (which) acts. And *software agents* are consequently – software entities able to act. Explaining the meaning of a new word by recalling the etymology, has the advantage that it preserves consistency with previous understanding of the word.

MAS (`http://www.aamas-conference.org/`) has become the most important (and the largest) conference in the agent research world.

Research communities use different adjectives such as *autonomous*, *reactive*, *social* etc, in definitions of various kinds of agents, primarily *software agents* but also *intelligent agents* or *mobile agents* and other. These adjectives describe key properties of agents.

After reading these definitions, it comes out that the adjectives often repeat and that their number is limited. This was noticed by Franklin and Graesser [36], and Murch and Johnson [81] who proposed agent classification systems in which combinations of the adjectives describe different agent types. The set of the adjectives is composed of the following[4]:

- *Reactive* (or *sensing and acting*) – responding in a timely fashion to changes in the environment.
- *Autonomous* – able to exercise control over their own actions.
- *Proactive* (or *goal-oriented* or *purposeful*) – goal oriented and able to accomplish goals without prompting from a user, and reacting to changes in an environment.
- *Social* (or *socially able* or *communicative*) – able to communicate both with humans and other agents.
- *Flexible* – having no predetermined behavior (no scripted actions).
- *Reasoning* – able to reason about their actions.
- *Adaptive* (or *learning*) – changing behavior based on previous experience.
- *Mobile* – able to roam networks freely.
- *Persistent* – maintaining a process until the desired result has been achieved.
- *Productive* – capable of achieving desired results.

Franklin and Graesser [36] also add another property, the *character*, which indicates the possession of a believable personality and emotional state.

The incoherence of agent definitions comes from the fact that separate research and development groups took different subsets of this set of adjectives to describe *software agents*. To avoid any misunderstandings, in this report the following convention is used, that the properties are always stated explicitly, by adding proper adjectives to the noun 'agent' (which consequently should be understood in its original meaning based on its etymological root). Since the software context of agents considered in this report is obvious the word *software* can be skipped, and from now on, the phrases *agent* and *software agent* will be treated interchangeably.

Furthermore, this report is mostly focused on *mobile autonomous agents*. This means software which can roam networks freely and make independent (unpredictable) decisions about which agent platform to visit next. In this report, the

[4]The list is made by combining Franklin's and Graesser's [36], and Murch's and Johnson's [81] classifications.

terms: *mobile autonomous agents*, *mobile agents* and *agents* will be used interchangeably.

This was a semiformal approach to define and systematise agents. Its application would allow researchers to more precisely demarcate the scope of their research area. On the other hand it is worth to mention the common sense understanding of the term 'software agent' – a software component, that is *autonomous*, *proactive* and *social* [10]. This perception of agents is supported by FIPA, which despite the fact that it does not directly define the notion of an agent, it gives a clear indication of the intended meaning, through specification of functionalities in an abstract architecture. Specifications of the Agent Communication Language (ACL) and library of Communicative Acts address communication between agents (*sociality*). A set of Interaction Protocols describes autonomous agent behaviours (*proactiveness*, *sociality* and *autonomy*), and support for ontologies opens the way to autonomous reasoning (*autonomy*) [34].

Generally an agent's structure is described in terms of a *state* and a *behaviour*. Formally, state is the set of properties (values, true propositions) that completely describes the agent [88]. In practice, this refers to data (variables and static values) describing the agent and the data held by the agent. Behaviour is a set of actions performed in order to achieve a *goal*. It represents a task that an agent can carry out [11].

To accomplish a goal, mobile agents roam from one network node (called a *container*, see below) to another, starting from a *base* (a.k.a. (*base station*). The sequence of containers to be passed is called a *route*. The agents' goal can be formulated through explicit indication of a node to be visited or in a more abstract way, without indicating nodes. An example of a goal formulated in this way might be: *to provide information about the cheapest vendor of a certain product*. In the former case the node is called a *destination*.

*A*n *agent platform* (AP) is an execution environment for software agents. It supplies agents with various functionalities such as agent intercommunication, agent autonomy, yellow pages, mobility etc.

Agent platforms are deployed horizontally over multiple hardware devices through the containers. On each device at least one container may be set up. Each container is an instance of a virtual machine and it forms a virtual agent network node. Containers make AP independent from underlying operating systems. Mobile agents are able to migrate from one container to another. Consequently, when containers are deployed on different devices, mobile agents can migrate between different devices.

The type of migration, when an agent travels to many containers and may or may not return to its origin is called *multi-hop migration*. Each trip from one container to another is called a *hop*. On the other hand there is *one-hop migration* and a *two-hop boomerang migration*. A one-hop agent travels from its base to a destination and then does not travel further. A two-hop boomerang agent travels from its base to a destination and back again [89].

When agents can migrate only within one AP, they are said to have *intra-platform mobility*. This is also sometimes called *homogeneous migration* since source and destination have the same interfaces and provide the same environment for the agents on both sides [25], in contrast to *inter-platform* or *cross-platform mobility*

where agents are able to migrate across platforms. The latter, called also *heterogenous migration*, [25, 48] though targeted in standardisation processes, is not supported in practice.

Agent platforms are implemented so that they allow either *strong* or *weak mobility*. In case of strong mobility both the code and the execution state of an agent are transferred. Weak mobile agents lose their execution state during migration and thus start their execution from the beginning after arriving at a new container [17, 37].

Agent platforms can be imagined as agent communities where agents are managed and are given the means to interoperate (communicate and exchange services). Many agent communities may coexist at the same time. Depending on the implementation of the platform, agents may be able to leave one community (platform) and join another.

Finally it is important to mention the abstract agent platform which is provided as a reference point by FIPA.

According to FIPA an agent platform must provide at least the following three components [35] [87]:

- Agent Management System (AMS).
- Directory Facilitator (DF).
- Agent Communication Channel (ACC).

The AMS exerts supervisory control over access to and use of the agent platform. It provides white-page and life-cycle services, maintaining a directory of agent identifiers (AID) and agent states. Each agent must register with an AMS in order to get a valid AID. The DF provides the default yellow page service for the platform. The ACC is the software component controlling all the exchange of messages within the platform, including messages to and from remote platforms [11].

In this report the MA-based communication, it means the network communication in which information is carried by mobile agents – active entities – moving across network nodes, is often referred to *traditional message-based communication* (or *traditional communication* or *message-based communication*) (for example, in Chapter 2, when describing techniques for providing untraceability). For the latter, the communication where *messages* – passive portions of data – are the carriers of information is referred. In this broad understanding of the notion of message, a network packet can be a message in particular.

Consequently MAS - Mobile Agent Systems are the systems with MA-based communication, and *traditional networks* are the systems where traditional message-based communication is performed.

1.3 MAS Overview: Benefits, Applications, Concerns

The agent technology offers a new paradigm of software development, in which even more than in the object oriented approach, the development can be performed through transferring concepts of the real world. The objects have been enriched with an active behavioural part (the behaviours) and their communication skills

were expanded. The behaviours show characteristics of intelligence: the agents infer and make autonomous decisions, spontaneously take the initiative, cooperate and interact with other agents, are goal oriented etc (see Section 1.2).

The agent technology offers a great aid in software design. Applications based on agents can be developed and planned similarly to the initiatives based on people and resources – system actors required to provide a new service are depicted, their behaviours are described and after that the deployment is performed, just like a new service is introduced in the real world. The service providers are moving to various locations, the service and its description is announced and spread across other agents. With common agent platform installed on all devices (which is the direction the technology goes) there will be a society of agents performing in a way similar to a society of human beings.

Additionally the technology offers advantages at technological level. They are mostly related to the decentralisation of computing (which eliminates overloads and single points of failure), uniformisation of communication (which among others enables cooperation between different applications) or agent mobility. The last characteristic is particularly welcomed in the applications were remote processing is needed but at the same time maintaining continuous data transmission links is hindered or uneconomical. An example of possible customer for such technology is cellular telephony where connection is not always available (on the sea, in the mountains, tunnels). Many robots designed for operation in areas of limited access (zones of increased radioactivity at nuclear reactors for example) or for very distant locations (including other planets, which are explored by unmanned space vehicles) are in fact realisations of the concept of autonomous mobile agents.

Drawbacks are primarily related to immaturity of the technology. The solutions are new and unknown, and without sufficiently long time of usage which would prove their reliability. And actually many of them are instable, which discourages from their use. Instead solutions based on older paradigms but tested and trustworthy are preferred.

It is worth to note the invaluable source of information about agents applications and other information about agents – the Internet portal of AgentLink III. AgentLink III is the premier Coordination Action for Agent Based Computing, funded by the European Commission's 6th Framework Program. Launched on 1st January, 2004, it provides support for the network of European researchers and developers with a common interest in the agent technology through events aimed at industry outreach, and standardisation issues, as well as providing support for academic events and providing resources through the portal.

1.3.1 Benefits

The benefits of using the agent technology are as follows:

- Business benefits.
  - Development of systems through metaphor of reality [68].
  - Easier deployment [68].

  - Higher availability [31].
  - Large-scale distributed or decentralized system integration with highly adaptive and dynamic business logic [68].
  - Flexibility and adaptability in business logic [68].
  - Step-by-step system integration [68].
  - Coalition formation [68].

- Technological benefits.
  - Bandwidth conservation [46].
  - Reduction of total completion time [46].
  - Latencies reduction [46].
  - Disconnected operation and mobile computing [46].
  - Load balancing [46].
  - Dynamic deployment [46].
  - Improved querying of various information sources [21, 46].

Business Benefits

Development of Systems through Metaphor of Reality The agent technology makes the next step, after the object oriented programming, towards the application development based on mimicking structures, behaviours and processes existing in the real world. Already in the object oriented approach the entities used in the system could be modelled and described, yet their interactions and behaviours remained artificial. In the real world stakeholders don't communicate with each other through mutual invocation of their methods.

In the agent paradigm instead, while all the advantages of the object approach were kept – as inheritance, encapsulation, polymorphism etc – the dynamic, active part is added. Agents show initiative, and they act spontaneously. They don't invoke methods, but communicate. They don't execute methods, but infer and make autonomous decisions etc (see Section 1.2).

With such developed paradigm it becomes truly realistic to design application on the basis of 'transfer' of solutions and concepts from the real world. If this is the factory system to be built – the instrumentation and the operators are to be modeled as agents and what should they do, what is their goal should be described [6]. If this is about optimisation of logistics, the supplies, suppliers, and the goods are to be represented as agents [75]. If people are the main participants of the system – the human models should be applied, roles and behaviors assigned [74]. Then, after the agents are deployed the system should evolve quite automatically – this is the idea behind the mature agent technology [68].

Easier Deployment In its target deployment the agent platform will be installed on each computing device (PC, cellular phone, PDA etc). One of enablers of this situation is the current development of agent applications – to deploy them users must firstly set up the agent platform on their architectures. This way step by step the platform spreads over the virtual world. With widely distributed agent platforms, the deployment of each new agent application, requires nothing more than the introduction of the new agents (or modification of old agents), which provide functionalities of the application. The agents will disperse in the environment, inform other agents about themselves, publish announcements in yellow pages services. Almost exactly the same as it is performed in the real world.

Availability While on the one hand, companies and organisations can easily deploy their systems, on the other users gain access to a great number of services offered in a consistent and coherent form. Ideally there would be only one agent platform, and the entire society of service providers (and customers) will be deployed on it. Each person having an access to the virtual society (for example through having installed the agent platform on their computer), has thereby the access to the whole mass of services offered by various providers. All they need to do is to submit their requirements to their agents, and in response the offers of many suppliers will be returned. Additionally replication and mobility of agents allows the creation of temporal copies able to accomplish their tasks locally at user devices. At the same time the coherent structure of agent systems prevents from the proliferation of different, often incompatible systems providing particular services from different organisations, where a user must comply with a new protocol each time they want to implement a new service.

Large-Scale Distributed or Decentralized System Integration with Highly Adaptive and Dynamic Business Logic Existing solutions are generally centralised, pulling everything onto one platform, which leads to overloads, delays related to them, up to denials of service. So to avoid these all negative effects, limitations on complexity and flexibility of the solutions are imposed. Centralised structures can also easily become targets of attacks or establish single points of failure.

A decentralised agent approach divides and conquers complexity by pushing a large part of the business logic out onto source systems so that much of the processing can be done on each of them. This distributes workload and increases robustness because the local processing can be performed independently of other systems, resulting in fewer and more relevant interactions with these systems, at a higher level of abstraction [68].

Exemplar benefits resulting from the decentralised computing can be found in the agent based crisis management [74, 107]. The traditional way of performing crisis management is through establishing a central crisis management centre. Such a centre however forms a single point of failure, and causes that all areas not communicated with it are left on their own.

The agent technology enabled organising ad-hoc structures in which agents

communicate with each other and exchange information about the environment in order to achieve the most faithful picture of the critical situation at the local area. For example agents representing individuals having medical skills inform about their presence, and so the people requiring medical attendance do. Then the matching agents assign the former to the latter, taking into consideration the geographical proximity, the seriousness of the injuries and the skills of the physicians etc. With such local ad-hoc structures, crisis management and decision making could be flexibly (and when necessary) shifted to the local areas in case the communication with the global centre was lost or the centre turned inoperative (for example because of terroristic attack).

Flexibility and Adaptability in Business Logic As easy is to deploy agents, that easy is to update the software based on them. It can be performed at system runtime without any interruption in the provision of services, through transition periods during which old and new agents temporarily coexist, and successive step-by-step removals, when the old agents are withdrawn, as they are getting free of their duties. This way of application updating proves a level of dynamism and flexibility that goes far beyond current release policies. Agent communication and behaviour capabilities complete the picture, being very well suited to high-level service-based interactions, the decentralised implementation of business logic, and for adapting and handling change in their environment.

Step-by-Step System Integration The highly component-based structure of agent systems fits integrating approaches of system development. Systems are developed step by step and each step results in added value for the business. For instance in the first stage of development the agent application may be addressed to smaller group of customers and may offer limited functionality. If the application is accepted by the customers, it can be easily made available to a wider public, thanks to the facilitated application deployment. Also the functionality of the application can be fluently extended, through dynamic updating or introduction of new agents. This mode of integration is particularly beneficial in the current economic climate, in which many companies have seen mega-projects fail [68].

Coalition Formation One of the intensively developed agent features is possession of social skills, related to inter-agent communication, negotiation, service exchange etc. These social skills enable agents to cooperate, to form coalitions and consequently to create aggregate entities capable of offering new, different or better services than might otherwise be available. It is similar to making possible cooperation of computer systems which previously operated in isolation – the benefits are numerous: faster service delivery (an overloaded service provider can ask for support their cooperating partners), more complex and comprehensive services (if additional services are required to fulfill a user demand, which are not provided by an application, the application can delegate this part of task to another application) more detailed and more appropriate information (agents cooperate in order to obtain the information of high quality) etc.

Technological Benefits

Bandwidth Conservation Conventional client-server systems are based on exchanging messages between client and server. When more interactions with servers are necessary to accomplish a client task, then the communication takes a significant part of the task and consumes a large amount of network bandwidth. The mobile agents technology saves the bandwidth by sending agent to the server and enabling the intermediate interactions to be performed locally. This was illustrated by the experiments conducted by D'Agents Research Team from Dartmouth's Thayer School of Engineering [46].

Reduction of Total Completion Time Since the agent technology significantly reduces the number of intermediate interactions through the network, it also speeds up the applications. At the same time, the agents complete the tasks faster only if the interactions involve enough operations for the transmission time-savings to make up for the migration overhead. The experiments of D'Agent Team [46] showed that agent performance relative to Remote Procedure Calls (RPC) improves with decreasing speed of network, but in relatively fast networks agents perform worse than the corresponding client-server solution works.

Latencies Reduction The substitution of remote processing with local processing caused by moving the computing unit from remote location to the local one obviously eliminates all delays related to communication. Additionally, mobile agents can dynamically choose the container on which to perform their task, to optimally minimise latencies. For example sometimes it may be sensible for an agent to migrate to a distant location (so its operation will require additional communication with original container) of higher computational power (so the received computation speedup will compensate the time spent on communication). The Sumatra chat server, written as a mobile Java agent, minimized the maximum latency two to four times comparing to the fixed-location server and its clients [46].

Disconnected Operation and Mobile Computing Traditional client-server network applications require a continuous (i.e. if connection is lost, the execution is interrupted and it must be restarted after the connection is reestablished) or in optimistic version – semi continuous (the execution is suspended when the components are disconnected, and continued from the last executed command after the connection is reestablished) connection between the client and server. For example in traditional systems when the connection gets broken during processing of a query, the user client must repeat the query when the connection is established again. This makes the client-server model harder to adapt to mobile computing since mobile computers often disconnect from the network, and when they reconnect, they often find themselves with a different network connection in terms of bandwidth, latency or reliability. A mobile agent, on the other hand, has the ability to operate disconnected from the client machine. It moves to a server or proxy machine and conducts its task independently to the status of connection. The connection is necessary only

for sending back the results. Moreover, agent based applications can easily adapt to network links of different bandwidths and latencies [46].

Load Balancing Load-balancing problems arise in a distributed computing system when there is an unequal work allocation at the different processing elements in the system. Three features of mobile agents support dynamic work allocation: agents can move from one platform to another, they can move across heterogeneous platforms, and they carry all application-specific code with them, rather than requiring pre-installation of that code on the destination machine. Most existing load-balancing systems are not as flexible as mobile agents, since they lack one of the three key features. For these reasons, load balancing is a significant growth area for the mobile-agent technology [46].

Improved Querying of Various Information Sources The typical Internet usage schema, which is usually related to information searching (e.g. searching for the most attractive price offer for a product or a service, such as the cheapest flight) relies on connecting to the website of one product/service provider, analysing and memorising (notes) the offer, and going to the next provider. Users usually perform a couple of such interactions (making a compromise between the time consumption of the task and the expected profit), to finally analyse (compare) the obtained results. Agents can improve this situation because these all described tasks can be delegated to them [21]. With agents it is sufficient that users describe their search criteria and launche the agent (see also Section 1.3.2). While the agent performs its task, users can spend their time on other activities, because no additional user-system interactions are required. Usually though there is no time to spend, because the agent (thanks to proliferation, cooperation, effective computing) finishes the task quickly.

Dynamic Deployment As with load balancing, mobile agents support the most general form of dynamic deployment, where an application can distribute its components dynamically to arbitrary network sites and those components can move at will from one site to another as conditions change. In fact, although the true strength of mobile agents is their combination of individual strengths, it can be argued that flexible dynamic deployment is what makes mobile agents such an effective choice for distributed applications. This is the dynamic deployment which allows mobile agents to conserve bandwidth, reduce latencies and completion times, handle disconnected operation, and balance load [46].

1.3.2 Applications

Four broad categories of applications are *particularly* suitable for agents:

- Assistant agents, such as agents engaged in gathering information (information agents) or executing transactions on behalf of their human principals on the Internet.

- Multi-agent distributed decision-making systems, where the agents participating in the system must together make some joint decisions [68].
- Multi-agent simulation systems, where the multi-agent system is used as a model to simulate some real-world domain.
- Multi-agent based integration.

Assistant Agents

Assistant agents per definition aim at assisting users and performing tasks for them. Assistant agents can moreover *represent* users and perform tasks *on behalf of* users.

The most often implemented function realised by assistant agents is information searching. The user defines their criteria and launches the agent which will autonomously crawl through the Internet resources. This searching is supported by knowledge representations (e.g. ontologies), inferring (for example an agent searching for a price offer may autonomously decide to look for offers presented in different currencies, after it infers the relationship between the prices), and communication and social skills (the agent may delegate the searching task (or its part) to other agents).

Another very popular (or even more popular) 'incarnation' of assistant agents are shopping agents (a.k.a. personal shoppers, shopping robots, comparison shopping services, or shop bots). Despite the fact that the Internet has made 'effective' (leading to the lowest price) shopping much easier for consumers (since moving from one online shop to another is just a matter of a few mouse clicks, comparing to time consuming physical moving or phone conversations), it still can be time-consuming, as users have to know the Web address of each online shop, and spend time searching for products in the site. Thus the idea of shopping agents to which all these tasks can be delegated. Currently numerous comparison shopping agents exist over the Web, which not only autonomously crawl the Web for the best offers, but also help users in specifying their preferences. They also guide them about choosing products if the users have only a general idea of what they want to buy [27].

Multi-Agent Decision-Making Systems

Applying agents to building decision systems became popular for two reasons: firstly because the agent approach is very suitable for making simulations (see Chapter 1.3.2), based on which, the decisions about the original processes and behaviours can be made. Secondly, because the highly componential structure of agent systems facilitates modelling of problem situations, which is especially true in case of the situations in which many influential factors exist.

The DaimlerChrysler's research team in Berlin has implemented a materials flow control system at one of the company's factories, entirely based on autonomous intelligent software agents. The system is used in the live production of cylinder heads from blocks of raw metal. Each metal block has to go through many processes with different machine tools to get shaped according to high specification. There is considerable flexibility in the choice of routes through the production line and the

order in which jobs are done [90]. Thus the aim of the agent system is to make optimal the choice of routes, and to most effectively schedule the jobs.

The decentralised approach obtained with the agent technology, the decomposition of the problem into autonomous units represented by goal-oriented agents, able to communicate and negotiate with each other resulted in high flexibility of the system, which in contrary to traditional approaches based on central control, can respond very quickly to sudden changes in the environment (related for example to machine disruption) [90]. With this dynamic agent coordination production of cylinder heads at DaimlerChrysler achieved savings of 10% [68].

Another example comes from the area of telecommunications and is related to the management of mobile telecommunications networks. Mobile networks typically have several types of network components: base stations, base-station controllers, mobile switching centres, etc [68], for which the most appropriate locations must be selected to obtain the most efficient work of the network. The problem lies in the fact that the 'appropriateness' of the locations is strongly dependant on the pattern of network traffic, which changes dynamically and very often – by time of day or by day of the week. Thus the optimal system would be the one which is able to dynamically reconfigure itself and to move its components according to the changes in the network traffic. With agents it has become possible to build such a system. Although moving network components is not usually feasible, the functionality of the components (encapsulated in agent) can be moved [68].

Other examples include decision making during a crisis situation [74], optimisation of a corrugated-box manufacturing plant [6], or scheduling of cargo fleets [75].

Multi-Agent Simulation

Agents especially fit simulation of systems and processes in which multiple interacting entities participate. One example is agent based battlefield simulation [7]. In this simulation agents represent soldiers, military outfit and outward features, and agents are grouped in teams with associated team roles (e.g. commander, observer, etc.) [7]. Another example is simulation of manufacturing process, where agents are used to model the factory instrumentation and its operators [6].

Another example is the RoboCupRescue – Simulation League [107] – an international testbed for the simulation of software agents and robots performing Urban Search And Rescue (USAR) missions. The main purpose of the RoboCupRescue Simulation Project is to provide emergency decision support through the integration of disaster information, prediction, planning, and human interface. A generic urban disaster simulation environment is constructed on network computers. Heterogeneous intelligent agents such as fire fighters, commanders, victims, volunteers, etc conduct search and rescue activities in this virtual disaster world. Real-world interfaces such as helicopter image synchronizes the virtuality and the reality by sensing data. Mission-critical human interfaces (for instance based on PDA) support disaster managers, disaster relief brigades, residents and volunteers to decide their action to minimize the disaster damage.

Agent Based System Integration

The agent technology facilitates the integration of heterogeneous systems. One scenario of such integration is to define interface agents which are able to interact with integrated systems. The interface agents must be able to transfer (forward) the data and the services originated in the systems, connecting them with the agent environment, where the data and the services will be further processed by other agents supported with the mechanisms for agent communication and interaction.

For instance such integration solutions are offered by Global IDs Inc. [39], whose data integration products are capable of simultaneously monitoring many hundreds of enterprise systems for relevant changes in data or metadata, by deploying mobile agents onto those systems [40]. The agents monitor local databases or applications, keep track of changes, can pre-process data and only forward relevant events or structured derived data to centralized collectors – in real time if required. The mobility of the agents allows highly customized functionality, which can be dynamically updated. Thus, the business user can change the business rules that are being executed at any point in time, while only relevant drivers and adapters are transferred to a source system. Agents can asses the impact of changes in the business rules and handle that impact throughout the integration process [68].

Application Domains

According to [68] the application domains where agent technologies will play a crucial role include:

- Ambient Intelligence – with agents enabling the seamless delivery of ubiquitous computing, continuous communications and intelligent user interfaces to consumer and industrial devices.
- Grid Computing, where multi-agent system approaches will enable efficient use of the resources of high-performance computing infrastructure in science, engineering, medical and commercial applications.
- Electronic Business, where agent-based approaches are already supporting the automation and semi-automation of information gathering activities and purchase transactions over the Internet.
- Bioinformatics and Computational Biology, where intelligent agents may support the coherent exploitation of the data revolution occurring in biology, and others including monitoring and control, resource management, and space, military and manufacturing applications, for example.

Ambient Intelligence The Ambient Intelligence vision [53] of people being surrounded by intelligent interfaces which form the environment capable to seamlessly and unobtrusively recognise and respond to the presence and to the needs of people, in fact describes an environment of numerous embedded and mobile devices working and interacting to support user-centred goals and activities. Realisation of such environment requires a componential approach, with components being able to

interact with each other while being widely distributed. This plus the required characteristics of intelligence, autonomy, often mobility etc make the agent technology practically dedicated for implementation of this environment [68].

Grid Computing Grid Computing is a form of distributed computing that involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations [47]. Grid applications, in which typically many services are involved, spread over a geographically distributed environment, which new services join and existing ones leave, thus very strongly suggest the use of agent-based computing. In this view, agents act on behalf of service owners, managing access to services, and ensuring that contracts are fulfilled. They also act on behalf of service consumers, locating services, agreeing contracts, and receiving and presenting results. Just as in the Ambient Intelligence vision, agents will be required to engage in interactions, to negotiate, and to make pro-active runtime decisions while responding to changing circumstances. In particular, agents will need to collaborate and to form coalitions of agents with different capabilities in support of new virtual organisations [68].

Biological Sciences One of the applications of agent systems in the Biological Sciences is simulation modeling of biological systems. Another area of application in biology is in Bioinformatics. With the arising magnitude of biological information, there is a great need for automated information-gathering and information-inference tools. Information-gathering agents may provide assistance to human researchers in finding appropriate research literature or in conducting automated or semi-automated testing of data. In addition, data mining agents may present human researchers with a set of potential hypotheses that can be induced from the data sources. In particular, the kinds of resources available in the bioinformatics domain, with numerous databases and analysis tools independently administered in geographically distinct locations, lend themselves almost ideally to adoption of a multi-agent approach [68].

Electronic Business To date agents have been used in the first stages of eCommerce, product and merchant discovery and brokering. The next step will involve moving into real trading – negotiating deals and making purchases. According to [68] in the very near future a boom in agent-mediated auctions is expected. The auction is a long-established and well-understood trading mechanism, and the available agent technology can support such agent mediated auction houses. It is anticipated that electronic commerce will dramatically change the supply chain since consumer will be able to contact directly the producer and avoid resellers. In the short term, travel agencies and retailing will be the primary business-to-consumer application domains using agent technology in eCommerce. On the other hand, it can be foreseen that agent technology in this market will enable small and medium enterprises to collaborate and form coalitions in much more flexible ways, almost regardless of geographic location [68].

1.3.3 Concerns

New Security Threats

The security of the currently predominating static distributed systems have been studied extensively and resulted in the design and implementation of numerous countermeasures. The mobile agent technology introduces significantly new threats [31] which relate to agents' mobility. Although these threats are the concern of various scientific groups and a number of solutions has been proposed, there are still problems not completely resolved. The most important difficulty is the agent exposure to the visited host. When an agent migrates to a remote host it is fully dependent on the good intentions of the host owners. If the host was maliciously altered, it is hard, if not impossible, [31] to recognize it.

The agent technology researchers agreed that resolving the security issues and developing a collection of security mechanisms to counter the associated risks, would allow users to freely develop useful and innovative solutions to existing problems and find a wide array of application areas that would benefit from this technology [54].

The issues of MAS security are further described in Section 1.4.

Lack of Mature Software Development Methodologies

The lack of mature software development methodologies for agent-based systems is another of the most fundamental obstacles to the take-up of agent technology. Basic principles of software and knowledge engineering need to be applied to the development and deployment of multi-agent systems and they also need to be augmented to suit the differing demands of this new paradigm [68]. The other question is related to agents creation frameworks. Currently the frameworks are fit for experienced system developers. To allow widespread popularity of agents, the agents construction environments for novice users should be introduced [76].

Small Number of Mature, Development-Oriented Platforms

Although there are several relatively mature, development-oriented platforms (with fewer agent-specific capabilities) and some richer, more research-oriented agent platforms, most platforms are still too immature for operational environments [68]. A larger number of mature platforms might enable a competition between the platforms providers and finally result in the improvement of their quality.

Lack of Awareness of the Potential Applications of Agent Systems

Agent technology is a relatively recent technology and there is a lack of awareness of the potential applications of agent systems [68]. This prevents system developers from applying the agent technology to their solutions, since the developers are unaware of the features of the technology and the paths of the development. AgentLink [2] propose a number of steps to be taken to facilitate greater awareness. These steps mainly rely on developing various case studies [68].

Cost of System Development and Implementation Due to Recentness of the Technology

Another concern is the cost of system development and implementation, both in direct financial terms and in terms of required skills and timescales. High deployment costs are a feature of any new technology. As agent design tools and standard methodologies are developed, and as development teams gain greater experience, these costs should fall [68].

Worse Performance in Particular Solutions

A mobile agent will not always perform better than a client-server solution. If the agent code is larger than the total intermediate data, the mobile agent must perform worse, since it will transfer more bytes across the network than the client-server solution. Similarly, if the network is fast enough, the agent might do worse even if the code is smaller, since mobile agents are typically written in an interpreted language for portability and security reasons. With a fast and reliable network, interpreting the agent on the server might be slower than transmitting the intermediate data to the client. As network speed and reliability drops, however, or data sizes increase, the situation changes considerably [45].

1.4 MAS Protection

The problem of the protection of agent systems involves the two questions:

- The issue of the protection of the agent platform from malicious agents.
- The issue of the protection of agents from malicious platforms.

Protection of agent platforms is relevant to the general problem of protection of execution environments from untrusted code, which has been investigated for many years and resulted in proposal of many efficient solutions. One of the recent instances (before agents) of the problem is related to client-server applications, where part of the code is executed at the client machine (e.g. aplets or AcitveX controls).

The techniques aiming at protecting agent platforms are based on the four main concepts:

- Detection of the maliciousness of the code prior to its execution (antivirus software works in this manner).
- Separation of the executed code in the execution environment from other processes and thorough control of its access to the system resources.
- Providing guaranties of safety of untrusted code (to make it trusted).
- Assuring safe execution of an untrusted code.

For detection of maliciousness of the code prior to its execution, apart from the basic technique massively used in antivirus systems, namely detecting through code signatures of malicious software, a method dedicated particularly to agents was proposed, called Path Histories [20, 89]. The method relies on agents securely creating lists of all the hosts they interacted with, so a new container can determine whether to execute the agent or not. Another method specific to agents is State Appraisal Functions [31]. The state appraisal functions aim at allowing an agent container to asses the level of alteration of the agent's state which occurred during the agent's migration. Based on the assessment the container can decide what privileges to grant an agent.

Software Based Fault Isolation [54] belongs to the second group of techniques (based on code separation). In this method, each untrusted code is stored into a distinct virtual address space called a *fault domain*. Then it is assured that all memory calls are performed only within the domain. This technique, known as *sandboxing* gained high popularity and is used, for example, in Java applications. Another approach based on code separation allow memory calls out of the address space dedicated for the untrusted code but these calls are controlled by reference monitors [54].

The basic method for providing guarantees of safety of untrusted code is through signing the code [54]. Another approach relies on providing a formal proof of safety of agent code [83].

An example of assuring safe execution of an untrusted code is Safe Code Interpretation [54]. In this method agents are run in interpreted environments and they can only run commands of the interpreter. Then the interpreters can either detect harmful commands and disallow their execution, or assure correct execution of commands [54].

Most of the described methods proved to be sufficient and it wouldn't be an exaggeration to say that the protection of agent platforms reached an acceptable level. On the contrary, the problem of agents' protection from malicious platforms establishes a serious challenge for researchers. This is related to the fact, that execution environments, namely agent containers on which agents are run, are out of the control of the agents and their owners.

Thus, in practice most methods focus on detection of malicious alterations, rather than on their prevention. Mostly it is done through the application of data integrity checking mechanisms, such as message digests or message authentication codes, to agents. If the agent code was maliciously altered, this alteration is detected at one of the next visited trustworthy containers after the harmful one. Then all results produced by the agent during its migration can be discarded or a subset of them. This group of methods include:

- Simple MAC based PRAC [125].
- Hash Chains [56].
- Chained MAC [56].
- Digital signatures [125].

- Chained digital signatures [56].
- Partial Result Authentication Codes (PRAC) [125].
- PRAC with one way functions [125].
- Verifiable PRACs [125].
- Multiple-hops Protocol [22].
- Execution Tracing [111].

In the progress of the research on agent systems some methods aiming at secure agent computation were proposed:

1. Multi-agent cryptographic protocol of Tate and Xu [105] is based on threshold cryptography and oblivious transfer. The computation security is achieved through involvement of multiple agents into computation of secure function.

2. A similar involvement of multiple entities into computation of one function takes place in fault-tolerant approaches such as Server Replication [78] and Agent Replication [125]. In the methods computing entities (servers and agents respectively) are replicated to simultaneously calculate the result of the function, which is later selected through voting [125].

3. Environmental Key Generation [97] is a scheme for allowing an agent to take a predefined action when some environmental condition is true. When the agent encounters an environmental condition, such as a particular string in search, a key is generated, which allows executing some agent's code. The environmental condition is hidden through either a one-way hash or public key encryption of the environmental trigger. The technique ensures that a platform or an observer of the agent cannot uncover the triggering message or response action by directly reading the agent's code.

4. Hohl [52] describes a technique called Time Limited Blackboxes. Standard non time limited blackboxes are implemented through scrambling the code of agents, so the code, though performing the same work as the original, is difficult to understand for external observers. Because there is no known algorithm for providing Blackbox protection, the time limited variant was introduced which allows achieving the blackbox security through changes of agent structure.

5. Sander and Tschudin proposed use of encrypted functions [98]. The user encrypts a function s, which is then executed by the host, without the host having access to s. Because Sander's and Tschudin's Undetachable Signatures [98] were not proven to be safe, Kotzanikolaou (et al) extended the research and introduced an undetachable signature scheme, based on exponential computing with encrypted functions [58] which filled the gap.

6. A provably secure protection is provided through introducing independent third party, such as a Secure Computation Service of Algesheimer (et al) [3] but this approach is in opposition to the whole idea of agent paradigm, where one of the key objectives is that computations were performed by agents locally at the containers where the agents arrived to. The Algesheimer's approach instead mixes the agent paradigm with the client-server paradigm.

Unfortunately the above methods are:

- either too theoretical (4),
- or too resource consuming and computationally costly (1, 2),
- or their security was not proven (5),
- or they address very narrow problems (5),
- or they impose too high restrictions on agents execution (6).

Thus the methods show rather limited applicability.

As a matter of fact, it is very difficult to propose a practical approach for assuring safe execution of agents on remote containers unless safety of the containers was guaranteed. Providing such a guarantee is extremely difficult (it can, for example, require application of beyond-the-software measures such as physical separation of computers, which hinders use of the computers [125]). However certain hopes are raised with relation to the concept of security enforced with trusted hardware [121]. Examples of hardware-supported security methods include:

- Trusted Computing Modules [110].
- Tamper-Proof Environments (TPE) [121].
- Secure coprocessors and trusted environments [124, 125].

In the domain of mobile agents security, it is believed that without the hardware support it would be impossible to protect the proper execution of mobile agents [20, 111, 121, 126]. The opinion of Chess (et al) from IBM T. J. Watson Research Center manifested in [20] that:

> 'It is impossible to prevent agent tampering unless trusted (and tamper-resistant) hardware is available in AMPs[5]. Without such hardware, a malicious AMP can always modify/manipulate the agent.'

was agreed by most of agent security researchers.

Finally it must be noted that in the research on agents security, insufficient attention has been paid to the issues of privacy in MAS, including anonymity of agent users and untraceability of agents.

An interested reader may refer to [15, 54, 125] for alternative overviews of protection methods for MAS.

[5] AMP – Agent Meeting Point – in Chess' (et al) nomenclature it denotes agent execution environment (such as container).

1.5 Chapter Summary

The chapter aimed at introducing the area of software agents. It started with a description of the history of software agents, where it was depicted that the development of agents was performed in three different research areas, thus some concepts (for example the concept of agent itself) may have different meaning depending on the area from which they originated. Then the concepts were introduced. Section 1.3 gave an overview of agents domain and showed that agent based Internet environments are an interesting alternative to the existing approaches of building software systems and expose many enabling features. However, protection of their security forms a great challenge, mainly because the agents are fully dependant on agent platforms. Finally the issues of agents security together with already proposed countermeasures were recalled in Section 1.4.

Chapter 2

Untraceability

2.1 Untraceability Introduction

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 [50]. Ceki Gülcü and Gene Tsudik [49] 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 by attacking an agent: through reading the agent's data or through *traffic analysis* (TA). If the agent's data are obscured (for example through encryption) then an attacker can still learn about the agent's origins through reading the agent address data (its base address, the destination address) which must be explicit (since the agent uses them for its operation) or by performing traffic analysis – for instance following the agent, observing its migration paths.

The aim of *untraceability protocols* is to hide address data. They may also, to a certain level, prevent from TA, but it is out of their power to achieve this goal completely. This is because there are various, and mostly potent TA attacks which can be executed to this aim. These attacks require more systemic defensive solutions.

2.2 Untraceability Definitions

Anonymity is the property of users enabling them to use resources or services (the *items of interest*) without disclosing their identity [82].

An attacker who observes an agent in order to disclose the identity of the agent's owner can use two methods (see Figure 2.1):

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

- Reading the agent's data.
- *Traffic analysis* (TA).

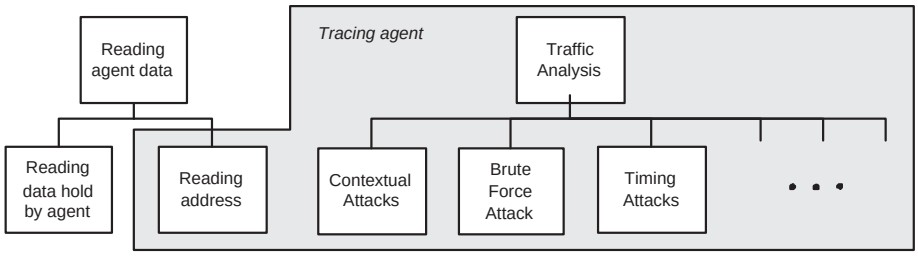

Figure 2.1: Attacks against anonymity of mobile agents.

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 from the point of view of their protection.

The data of the agent are: the agent's execution code, the agent's state, and *operational data*. Operational data are the data required for basic performance of 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 the base station needed by the agent for its return. Since the operational data must be of high availability for 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 tasks. 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*.

It is assumed 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 based on agent distinguishing features, which are described in Chapter 2.3. *Traffic analysis* (TA), is a process of intercepting and examining agents in order to deduce information from their patterns of communication (see Chapter 2.3 for more details).

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 agent's destination or both. Using the anonymity terminology dedicated for message based communication [91] they aim at violating subsequently: *sender anonymity*, *receiver anonymity* and *relationship anonymity* (also *unlinkability*)[2].

The state of agent's protection against tracing is called *untraceability* [91].

[2] *Sender anonymity* assures that it is not possible to identify the sender of a message. *Receiver*

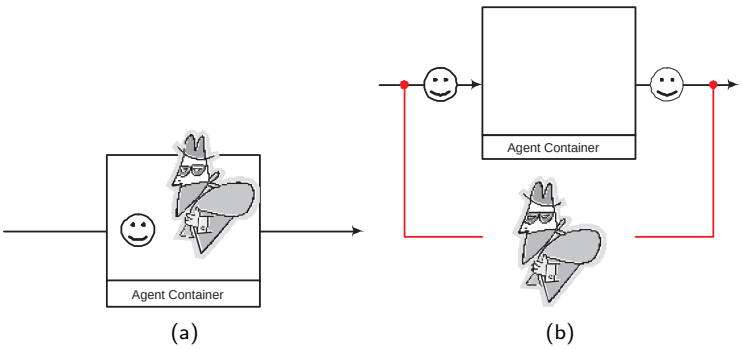

Figure 2.2: Internal (a) and external (b) adversaries.

2.3 Traffic Analysis

The term *traffic analysis* (TA) comes from military intelligence, and it describes a process of intercepting and examining messages in order to deduce information from their patterns of communication. The first significant uses of TA date the Word War I. With the advent of Internet TA was soon applied to Information Technology (IT).

In IT, TA is defined as the analysis of network traffic flow for the purpose of deducing information that is useful to an adversary. Examples of such information are frequency of transmission, the identities of the conversing parties, sizes of packets, flow identifiers used, etc [5, 32, 99]. Another definition is brought in by Chaum, who describes *the traffic analysis problem* as the problem of keeping confidential who converses with whom, and when they converse [18].

Traffic Analysis has been already studied for over twenty years and various TA attacks were recognised and described. Of the prominent studies (e.g. [23, 26, 41, 49, 57, 71, 93, 95]), Raymond's appears to be the most comprehensive (it actually summarises all the others).

In this work the known attacks were referred to MAS. The description of the attacks on the ground of message based communication are discussed in the already mentioned works [26, 41, 57, 71, 93, 95]).

It must be underlined that TA concentrates on patterns of communication, obtaining direct data indicating the base and/or destination is out of scope of TA and as it was stated in the previous section (Section 2.2), it belongs to attacks based on reading agent data.

2.3.1 Adversaries

The following types of attackers are considered:

Internal / external – The *internal* adversary is the adversary who succeeded in compromising a container, and is able to observe agents within it (*passive*

anonymity means that the intended recipient cannot be inferred from the message. *Relationship anonymity* means that it is untraceable who communicates with whom [91].

– see below) or even may obtain control over the container (*active* – see below). The *external* adversary is the one who aims at particular container but was able to compromise (for example to brak security of an encrypted link) only the communication channels leading to and from it (see Figure 2.2). Internal attackers are viewed as more potent than external attackers since they have access to more resources and in particular they may observe the agent's behaviour at the node to distinguish the agent from others. Considering a particular node, an internal attacker who managed to compromise the node has at least as good view as external attackers who managed to compromise all channels entering and leaving the node. On the other side, the external attackers are able to observe agents coming from / to other nodes if only they pass through the channel the adversary observes. Note that the distinction between the two types of attackers must be made in the context of a particular container. Thus an attacker who compromised a number of containers but is aiming at another one (which they were not able to compromise) should be viewed as external from the point of view of the targeted container.

Omnipresent / k-listening – The adversary may succeed in attacking all nodes (then the adversary is called the *omnipresent* adversary), or k of them (the adversary is called the *k-listening* adversary [26]). Or, in particular, only one node may be compromised (the *single* adversary) [104]. It must be noted that although in practice it often occurs for the attackers to become omnipresent adversaries in local area networks (the attacker manages to obtain the access to network administrator account), becoming omnipresent adversary in WANs, is commonly considered as low probable and in the Internet – as infeasible[3] – due to obvious reasons (resources required and technical difficulty in breaking into various differently protected systems, complex administration of distributed agents of the adversary, complex information processing and much more). In broadly distributed networks the attackers, in the most optimistic scenarios, manage to observe k selected nodes (which they chose as the nodes of particular interest).

Active / passive – An *active* adversary can arbitrarily modify the computations and messages (by adding and deleting) whereas a *passive* adversary can only listen [93].

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 [67, 93]. They can, for example, 'follow' messages [93].

Hybrids and alliances – *Hybrids* and *alliances* of attackers may occur, such as external-active or colluding internal and external. Syverson (et al) [104], for

[3] At least as long as the adversary doesn't gain a highly institutional or governmental control of organisations controlling or attempting to govern areas of the Internet – see `http://www.wgig.org/`, `www.icann.org`.

example, distinguish between *multiple adversary* and *roving adversary*, which are subsequently: k-listening static and k-listening adaptive adversary.

For the matter of convenience the k-listening, roving and omnipresent adversaries will be altogether called as *k-present adversaries*[4].

2.3.2 Time Correlation

Description

Time correlation, together with content correlation (see the next Section) are the most basic traffic analysis attacks. Time correlation is, as the name indicates, activity of linking incoming and outgoing agents through time relations. In traditional networks it is easy to trace messages passing intermediate nodes, since the first-in-first-out rule is often satisfied. Also for agents, if they only pass an intermediate container, where they don't perform any tasks, it is very probable that this feature will occur.

Time correlation may be performed by k-listening or omnipresent adversaries, or multiple colluding roving adversaries (spying agents) – k-present adversaries (see the previous section). Either external or internal.

Countermeasures

Countermeasuring this attack is done through breaking the obvious time relations between incoming and outgoing agents. This can be done through stopping agents at ech container until a particular number of agents arrives to the container, forming a *batch* of agents. Obviously, it must be assured that the messages in the batch are not located in the same order as they arrived to the mix. This requirement is satisfied through *reordering* messages – changing their location in the batch[5]. This method was already utilised by Chaum in the first mix [18].

Another method relies on delaying agents for a random time at each container, where the time of delay is assigned according to probabilistic calculations based on statistical characteristics of network traffic, to make sure that this will lead to expected reordering of agents [57].

2.3.3 Correlation Based on Distinguishing Features

Description

The attack is based on the fact that it is easy to recognise and trace agents through subsequent containers, as long as agents are easy distinguishable from each other. Such distinguishing features may be, for example, varying, non-uniform size (in terms of number of packets) of agents or characteristic content of data held by the

[4]Where k is equal to number of nodes in the network for omnipresent adversaries

[5]Note that the actual strength of n-batchs doesn't lie in the fact of *releasing* messages in n-sized batchs but in assuring that there are n messages at the same time in the mix (and they are reordered). In practice, the messages collected in the batch don't leave the mix concurrently but in a sequence.

agent (attacks targeting these particular features are also called *content correlation* attacks [49]) [49, 93].

In this attack the goal of the attacker is to follow the agent, thus the more nodes the attacker is able to observe, the more likely they will discover the agent base and destination. The attack may be performed by k-present adversaries. Either external or internal.

There is also active version of the attack (see Agent Tagging 2.3.12).

Countermeasures

To prevent from this attack the agents must be made undistinguishable. First, a standard size of agents in MAS must be induced (*agent size uniformising*). Afterwards, all the data which can make agents different to others, must be obfuscated (*data obfuscation* or *content uniformising*). This obfuscation may be done through one-to-one hashing (permutations), through encryption or any other method which assures that the obfuscated data are different before and after traversing each container (an interesting method of agent data obfuscation can be found in [52]).

Unfortunately, *size uniformising* of all agents leads either to costly redundancies if the size is large, or to inconvenient boundaries, if the size is too small, imposing the distribution of data between more than one agent. Thus at least groups of agents of the same size should be introduced to MAS.

2.3.4 Brute Force Attack

Description

Suppose the deserted MAS in which just few agents roam. Then an attacker may learn much about agents bases and destinations just through tracing them. This is, through observing each agent entering a container and then following all outgoing agents. This is called *brute force attack* [93].

This attack may be very successful in deserted MAS'es, especially when the agents follow separate paths and they don't meet at the same container at the same time[6].

To the contrary, if the network is very crowded, and agents normally come to the same containers and stay there at the same time, it is difficult to learn something about agents since it is difficult to correlate agents coming in and leaving the same containers (this is actually the observation which stays behind the idea of the mix [18]).

Even though, one may try to apply the brute force attack since situation in the network changes and it may happen that in certain network areas some separate paths will be observed. In this case it is important to perform the attack continuously for a long time.

The attacker performing a brute force attack must be able to trace multiple agents if it occurs that the particular agent visits one container and then many

[6] Needless to say, then there is always one agent entering a container, and one leaving it, and moreover – this is the same agent.

agents leave it. Thus the attack is attributed to k-present adversaries (see the previous section). Either external or internal.

In practice, apart of the deserted MAS case, it is relatively difficult to perform the brute force attack. First of all it is rather unlikely to obtain access to k-locations (or all of them) in MAS, or to introduce a high number of spying agents. Second, the observation must be made during a long time. Finally, correlations in between the received observations must be found.

Countermeasures

Guaranteeing that containers are always visited by a certain number of agents and more than one leaves a container at the same moment, decreases the probability of linking a base and a destination exponentially with the length of the route. However it may happen that the network has low traffic and at a particular instant, the agent was the only one which traversed a given route. Then it is totally exposed to an adversary.

In traditional networks, to avoid this, *dummy traffic* is usually introduced, which means redundant messages sent from one places to other to make the network always occupied. The more dummy messages are sent, the less probable is to correlate certain messages. This is a costly option, overloading the network, narrowing its bandwidth, introducing redundant computations.

In MAS, the situation appears to be better, since all nodes are equal in the sense that there is no distinction between mixes and ordinary nodes (yet all nodes are mixes). So the attacker cannot easily detect which node is the destination one. They might guess that it is the last one before returning to the base, but it is enough to force the agent to traverse additional containers, a so called *redundant agent migration*, on its way back to prevent such inferences. Thus if there is only one destination for a particular agent, it is very difficult to deduce which one is it.

To prevent from discovering the agent's base station, one option is to introduce *redundant agents* which start their lifecycle on random containers and then roam around the network waiting for a task. Thus when a user needs an agent they pick it up rather than launch. They assign the agent a task and let the agent migrate in order to fulfill it. After the agent comes back with the results, the results are read, and the agent is left free to finish ts lifecycle. Because the agent started its lifecycle earlier on a container different to the one where it is being picked up, this previous container is the base container and not the one where the agent was employed. Note that these redundant agents differ from the 'dummy' ones, since they finally are being used. As well as their number can be controlled and adjusted according to agents usage statistics.

This means of protection is not as strong as the introduction of dummy traffic, because the probability decreases linearly with the route length, but it is much less costly and may be sufficient, especially if the network is naturally loaded with normal agent traffic.

2.3.5 Replay Attacks

See Agent Tagging – tagging through shadowing – Section 2.3.12.

2.3.6 Timing Attacks

Description

If traversing a route of a particular length always takes an agent a specific time, then an adversary may correlate times taken by observed agents. Raymond, for example [93] assumes two routes to cover, for which subsequently 1 second and 3 seconds are always required. Then if two messages (as it was already said, the Raymond's work is dedicated to message based communication) are observed arriving in the network at 0:00 and 0:01 and leaving it at 0:03 and 0:02, discovering which message passed which route is trivial.

Regarding the types of adversaries able to perform this attack, two attack stages must be distinguished. In the first stage, the attacker measures times needed for covering different routes. To obtain comprehensive time information, the attacker must compromise many network locations (to observe which route is being passed when the time is measured). Thus this part of the attack is performed by k-present adversaries. Either external or internal.

Once the attacker owns the necessary data, then it is enough to locate at particular container which the attacker assumes to be a base container for a particular agent. Then the adversary measures time between the agent leaves and returns to the container, learning by this which route the agent passed, and possibly what was its destination. At this stage, it is enough for the adversary to be a single, passive adversary – internal or external.

It is important to note, that the attacker is already in possession of significant information if they are able to establish that a container is the base for a particular agent.

Countermeasures

In MAS, timing attacks are less successful than in traditional networks due to the difficulty in identifying the destination (see section 2.3.4). What the attacker receives when performing this attack in MAS is the information about the route the agent passed, which may indicate some *potential* destinations.

To increase this already high level of anonymity it is worth considering the introduction of random delays of agents' visits to containers or batch processing (see Sections 2.3.2 and 2.3.7). Note, that in practical agent environments, in which agents roam and perform tasks, random delays may naturally result from the tasks.

Furthermore, if redundant migration or redundant agents (see Section 2.3.4) are applied, then attacker is not able to learn neither about the base nor the destination, which results in agent's unlinkability (see Chapter 2.2).

2.3.7 The Node Flushing Attack (a.k.a. Spam Attack, Flooding Attack, n-1 Attack, Isolate & Identify Attack)

Description

As it was already mentioned when describing the brute force attack (Section 2.3.4), the key thing to make attackers' work more difficult is to cause that many agents are at the same container at the same time so it would be complicated to correlate the incoming agents and the outgoing ones. As it was described (see Section 2.3.2), the method which aims at imposing this feature in traditional message communication is *batch processing*. The method relies on deliberately collecting *n* messages before they are released (*flushed* – see [93]) from the mix. This causes that messages leave mixes in *n* sized batches. The crucial role in this method plays *reordering* of messages, so the time correlations between incoming and outgoing messages are broken. The method hinders performing brute force attacks, as well as timing and time correlation attacks.

The cost of this functionality are message delays and uneven network traffic. There are 'waves' of messages in the network instead.

The attacker response to this protection is the *n-1 attack* (other popular names are: *node flushing attack*, *flooding attack* and *spam attack*). An adversary 'fills' the mix with *n-1* of their messages, allowing only one foreign message to join the batch. Then when the batch of *n* messages leaves the mix, it is trivial for the adversary to separate the message which they observed.

This attack is primarily performed by external adversaries. As in previous cases, to trace the agent the adversary must observe multiple nodes, thus they must be k-present.

Countermeasures

To prevent from the n-1 attack, it must be assured that either the adversary is not able to recognise their `n-1` agents after they leave the container or to disallow the attacker to deliver the agents to the container.

When discussing the countermeasures against this attack, Raymond concludes that by encrypting the traffic between containers, the attacker looses the ability to easily recognize their agents. But this is not true. If the attacker assures that all their flooding agents go to a particular container, after leaving the observed one, then they are distinguishable unless the observed agent also goes to the container (which actually doesn't interfere with the main aim of the adversary, which is to recognise where the agent goes).

This encryption disallows the adversary to recognise their agents through examination of their data. Raymond discusses the encryption but this can be also one of other data obfuscation methods discussed in Section 2.3.3.

Another solution (used in Stop-and-Go mixes [57]) is to force agents to wait at the container for a random amount of time. Then even if the adversary flooded the container with their `n-1` agents causes, they will not leave the container in the same batch. Instead, *k* agents not belonging to the adversary will be in the batch, which

hinders correlating incoming and outgoing agents. This countermeasure appears to be effective, for the cost of delays in agent communication.

Summarising, in MAS, instead of n-batchs, the agent data obfuscation described in Section 2.3.3 together with agent delaying is the most effective method of protection against brute force attacks accompanied with n-1 attacks.

2.3.8 Contextual Attacks

Description

These attacks refer to the communication habits of users. They are performed by k-present adversaries.

There are three types of attacks in this group: *communication pattern attacks*, *packet counting attacks* and *intersection attacks* [93].

Communication pattern attacks aim at observing users' habits in using network services. A good example of situation making it easy to perform this type of attack is a company worker who prefers to work late evenings. If the person is the only one using the company MAS at the particular time then it is not difficult to connect agents active in the MAS with the person.

In case of packet counting attacks, a user makes the task of tracing them easier by launching an agent of a characteristic, distinctive size (in terms of number of packets). This can be done for example through attaching a file to the agent.

When performing intersection attacks, an adversary observes network traffic and stepwise narrows the range of possible interlocutors (as described in [13]). This attack actually undermines the defence of using *different* routes each time an agent goes to the same destination. Imagine an agent travelling twice from a base `A` to a destination `B`, each time passing through completely different containers. If an adversary observes these two trips, they notice that the only containers in common are `A` and the `B`, which makes them good candidates for interlocutors.

Countermeasures

A method for avoiding packet counting attacks, called *size uniformising* was already described in Section 2.3.3.

In general, contextual attacks are difficult to protect from, because they rely on factors which are out of the control of system designers and administrators, namely the unpredictable users behaviours.

The only way to prevent from the attacks is to avoid performing characteristic activities, for example not to have distinguishing habits.

Thus they are the users who are responsible for protecting themselves: they should be conscious of the fact that by performing identifiable activities, they become more distinct and thus easier to trace.

2.3.9 Denial of Service Attacks

Description

First, it must be underlined that the following description focuses on *denial of service* (DoS) attacks which *aim at compromising untraceability* – not on all denial of service attacks.

An attacker compromises some mix-nodes (making them inoperative), counting on the fact that this will force users sending their messages through those nodes to change their behaviour. Though this attack always works in the case of conventional networks, in many MAS it appears to be ineffective, since agents arbitrarily choose a different route each time they roam. In case some containers are nonfunctional, they simply omit them and choose other. This feature of MAS is further described in the Section 2.3.12.

The attackers must be active (static or adaptive, external or internal) and k-present adversaries.

Countermeasures

It must be assured that an agent will not behave abnormally when the destination container is compromised. It means that the agent should pass the whole route as if nothing had happened (not quickly return to its origin to report the damage). However it is not required from the agent to continue the completion of its task. The agent may stop its goal-oriented activities. The important thing is that an *outside* observer could not notice the difference in the agent's behaviour.

For this property, the name *failure-neutral behaviour* is proposed. It designates the characteristic of an active entity that it is impossible to distinguish between its system-normally-operating and system-failure-occurred behaviours, through *external observation*. Failure-neutral behaviour is a requirement weaker that *fault-tolerance*, however it can be perceived as a part of it. Fault-tolerance refers to a more restricted property of systems, requiring them to continue their proper operation if the failure occurrs. For failure-neutral behaviour it's not required from the entity to remain properly operating, internal functionalities may get lost.

2.3.10 Active Attacks Exploiting User Reactions

Description

An example of active attack exploiting user reactions is agent interception, cloning it and sending the clones to all possible recipients. The idea is that the original recipient will behave differently from others.

As in case of denial of service attacks, these attacks are performed by active (static or adaptive, external or internal) and k-present adversaries.

Countermeasures

One solution for preventing this kind of attacks, is to assure that agents go further than to their base containers when returning (*redundant agent migration*). This

idea was already described in Section 2.3.4.

Also the feature of completing the whole route even in unusual situations (this time it is that the declared destination appears not to be the real one) – the failure-neutral behaviour – as in case of DoS attacks (see section 2.3.9), should be included.

It is very difficult to imagine all the attacks aiming at provoking characteristic user reactions, and so it's very difficult if not impossible to propose a one effective prevention method. It seems that the only realistic approach is through *learning from experience* – known from the area of malware protection, where countermeasures are developed as soon as new attack is discovered. Then it is very important to exchange knowledge about the attacks and protection against them.

2.3.11 Agent Delaying

Description

In this case, an attacker stops an agent until he gathers enough network resources or until the network becomes easier to monitor or to see if possible recipients receive other messages, etc.

The attack is performed by active (static or adaptive, external or internal) adversaries in order to facilitate further investigations of k-present adversaries.

Countermeasures

To protect from this attack, administrators should consider introducing *authenticated timing information* – securely encapsulating arrival and departure times at containers into agents. For example time windows, as in Stop-and-MIX'es [57] can be used.

2.3.12 Agent Tagging

Description

An adversary purportedly 'marks' the agent (alters the agent's data or behaviour to make the agent distinctive) to facilitate its tracing. There are three types of attacks identified in this group: *tagging data*, *tagging through delaying* and *tagging through shadowing*.

The first and the most intuitive type of tagging attacks relies on changing the agent's data, so the agent was easily recognised from other agents. The attacker may add some characteristic data to the agent but also remove or change existing data.

Feasibility of tagging data attacks is very dependant on a particular MAS architecture and should be discussed in relation to a real environment. In most cases, because of network protocol characteristics, such tagging is not feasible on the network layer, what makes the attack unavailable to an external attacker. Thus, in the first, tagging stage, the attack must be performed by active internal adversaries.

Once the agent is tagged, its tracing, as in case of correlation based on distinguishing features (see Section 2.3.3) belongs to roving adversaries, k-listening or omnipresent adversaries. Either external or internal.

In contrast to agent delaying attack (see section 2.3.11), the tagging through delaying attack aims at distinguishing an agent in the network. During this attack, the attacker forces the agent to stop at each container for a specific, characteristic time. After this correlating arrivals and departures of such tagged agent to and from containers is trivial.

The attack is available to active, internal/external attackers, with ability to observe the agent at multiple containers (thus k-present).

The tagging through shadowing attack (more often known as *replay attack* [49]) is based on intercepting the agent and copying it. After this, k copies of the agent traverse the same route. This attack is effective in traditional networks and in the MAS'es where mobile agents plan their routes using a deterministic algorithm. In this systems, the copies of messages / agents will follow the same route as the original.

In MAS'es where agents have freedom in autonomously choosing their route the attack appears to be useless since each of the copies autonomously chooses its own route. Thus it is important to make sure that agents choose their routes in a nondeterministic way. This is called *nondeterministic routing*. Deterministic routing forms another serious vulnerability – if adversaries can discover its algorithm they could predict agent routes.

Countermeasures

Tagging through delaying attacks are to be avoided by means of early detection of alterations of the agent's code (*tamper detection*), or, in the best case, by not allowing the adversary to change the code of the agent (*tamper-resistance*). The methods aiming at detecting agent data alterations are Agent Tampering Detection by Storage Jamming [72] and various schemas based on hashing agents' data. In the latter case it is important to assure that the hashes are verified at each container, so the alteration was detected before the agent covered its route.

As explained in Chapter 1 preventing agent data from being altered is very difficult if not infeasible. So far not an effective method has been proposed to resolve this problem.

Protecting from tagging data attacks is similar to protecting from tagging through delaying attacks since in both of them it is crucial either to detect or not to allow the alteration. In case of tagging data attacks the protection task is slightly easier since this is *data* – a more static part of agent (comparing to code) to be processed.

The basic technique for preventing tagging through shadowing attacks is *replays detection* which uses one of the following techniques: sequence numbers, random numbers, data and time stamps [49] or the most ineffective – keeping record of previous agents [18]. On the other hand if agents choose their routes in nondeterministic way, as it was described above, then the copies of agent don't follow the same way as the original, which makes the attack ineffective.

2.3.13 Corrupted Party Attacks (a.k.a. 'Sting' Attacks)

Description

The corrupted party attacks rely on taking over either the agent's base or the destination and masquerading as genuine party of the communication. Describing the attack on the ground of traditional message-based communication Raymond brings in the example of government agencies setting up home sites with illicit looking content (such as fake drug-production instructions) and observing queries to the sites. The same way, the attacker can query sites, and observe responses, behaving as the initiator of communication.

The mentioned observations of queries and responses can be performed because the attack assumes that an adversary is able to encode some indicative information into an agent (see [93]), and then to follow the tagged agent as in case of agent tagging attacks (see previous Section (2.3.12)).

Thus the corrupted party attacks are primarily attributed to internal active k-present adversaries.

The attacks prove to be adequate for active internal adversaries who are not able to observe larger areas of network (in practice – to single adversaries). In this case, the adversary tries to involve into conversation the user located on the other side of the communication channel, so the agent after returning to the user, would came again to the adversary. Then the attacker tries to provoke the user to release some identifying information about them or to make the agent obtaining address data of the base container.

Countermeasures

The most effective protection method is to not allow to tag the agent at the destination (tamper resistance, see Section 2.3.12). However this is possible only in situations where the agent was not envisaged to collect any data from the destination.

If the agent gets involved into a conversation, then it is difficult to detect if the exchanged data may serve for the tagging purpose. One approach could be to detect the data which are not used at the user's side but returned to the destination (tamper detection, see Section 2.3.12).

In case of single adversaries aiming at making the user or the agent to realise identifying information, it must be assured that such information would not released. Although, it is possible to provide such guaranty for the technical level (through assuring containers not to disclose their identifying data to agents), as it was already described for contextual attacks (see Section 2.3.8) – it is impossible to control users' behaviours. And as in case of contextual attacks, the users must be responsible for protecting themselves: they should be warned not to share their identifying data with not trusted parties.

2.4 Solutions for untraceability of messages

The problem of untraceability was studied extensively for traditional message-based communication, for which during past 25 years various techniques were proposed.

Most of these techniques 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 the application of 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 [13] and they can offer different functionalities, many different mix-extensions have been proposed. Examples include: Non Disclosure Method (NDM) [32], BABEL [49], Onion Routing [43,94,103,104], Crowds [95,96], raw remailers, Cypherpunks (Type I remailers), Mixmasters (Type II remailers) and others [28]. These solutions differ in complexity and provide varying levels of protection.

Another approach for providing untraceability is based on Chaum's *DC* (*Dining Cryptographers*) network [19]. However, due to the complexity of the proposed solutions it still remains in the experimental phase [4,101]. 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. An extension of the DC network, called DC+ network, was proposed by Waidner (et al) [114, 115]. An interesting solution are XOR Trees of Dolev (et al) [26].

An alternative conception for obtaining untraceability is publishing the message among multiple parties, while only one of them (or a subset of them) is the intended reader. Since the ID of this intended addressee is not stated in the message directly (implicit addresses are used instead [92]), and the message itself is available to all participants, there is no indication to whom the message was addressed. This kind of publishing can be performed through message pools such as newsgroups and mailing lists [42] or through message broadcasting (with implicit addresses [92]).

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 [46]. Very few solutions for MAS untraceability have been proposed so far [29,117]. They are based on the conception of onion routing [94]. In further chapters an untraceability protocol which does not impose such limitations will be described.

In this chapter an overview of the techniques is provided. Alternative overviews can be found at [13,42,57,79]. Also an attempt to evaluate some of the techniques in regard to their resistance to TA attacks was made – see [12].

2.5 Chapter summary

The chapter described the problem of untraceability. The problem has been studied extensively for traditional message-based communication for the last twenty five years and resulted in the thorough description of both: attacks and countermeasures. After explaining the notions of untraceability and related terms, the complex issue of Traffic Analysis is described. The attacks and the relevant countermeasures are presented. Then the protection techniques developed for the area of traditional communication are recalled. The techniques appertain to three main groups depending on the method on which they are based: mixes and their modifications, DC networks, publishing. It was pointed out that not all the methods can be directly applied to MAS and only mixes can be adapted to a certain degree. The only existing untraceability solutions for MAS were not actually proposed for directly addressing the problem of untraceability, and yet as they are based on the conception of onion routing, they restrict agents ability to spontaneously choose their containers during migration. This results in the loss of some of the advantages of agents.

Chapter 3

Summary

This report summarised results of studies aiming at the identification of threats to privacy in agent-based systems and the methods of their protection.

Although anonymity was studied thoroughly for traditional message-based communication and resulted in proposals of various protection techniques, for agent systems this problem has never been sufficiently addressed. The research presented in this report aimed at filling this gap.

The first chapter introduced to the area of software agents. It started with a brief description of the history of software agents, followed by the explanation of the main concepts related to software agents. Section 1.3 gave an overview of agents applications and showed that agent based Internet environments are an interesting alternative to the existing approaches of building software systems. The enabling feature of agents is that they allow software development based on the 'metaphor' (see [70]) of elements of the real world. This means that they allow building software systems which work as human societies, in which members share products and services, and cooperate or compete with each other. Organisational, behavioural and functional models applied in MAS can be copied from the real world. However, together with many advantages of the new technology, there comes a big issue – providing sufficient security to agent systems is a demanding challenge to the researchers and developers of agent systems. This is mainly due to the fact that agents are fully dependant on their underlying execution environments. The issues of agents security were presented in Section 1.4.

The second chapter described the problem of untraceability. After explaining the notions of untraceability and related terms, the tracing attacks against mobile agents were presented together with the adequate countermeasures. Then the protection techniques developed for the area of traditional communication were recalled. The techniques can be divided into three categories depending on the method on which they are based: mixes and their modifications, DC networks, publishing. The observation was made that not all the methods can be directly applied to MAS and only mixes can be adapted to a certain degree. The only existing untraceability solutions for MAS were not actually proposed for directly addressing the problem of untraceability, and as they are based on the conception of onion routing, they

restrict agents ability to spontaneously choose their containers during migration. This results in the loss of some of the advantages of agents.

Bibliography

[1] Ad Astra Engineering. Jumping beans – the mobility framework, December 1998. Available at http://www.jumpingbeans.com/ (last access: March 21, 2006).

[2] AgentLink. AgentLink III – European Co-ordination Action for Agent Based Computing. Website. Available at http://www.agentlink.org/ (last access: July 21, 2006).

[3] Joy Algesheimer, Christian Cachin, Jan Camenisch, and Günter Karjoth. Cryptographic scurity for mobile code. Technical Report RZ 3302 (no. 93348), IBM Research, Rüschlikon, Switzerland, August 2003. Available at citeseer.ist.psu.edu/602631.html.

[4] Anon. Experimental crypto software: DCchat. Website. Available at http://www.geocities.com/cryptosw/dcchat.html (last access: May 31, 2006).

[5] R. Atkinson. RFC 1825: Security architecture for the Internet Protocol. Website, August 1995. Available at http://www.faqs.org/rfcs/rfc1825.html (last access: May 16, 2006).

[6] Roxana Belecheanu, Michael Luck, and Vince Darley. Agent-based factory modelling: Eurobios and sca packaging a case study. Technical report, AgentLink III, November 2005. Available at http://www.agentlink.org/resources/webCS/AL3_CS_002_EUROBIOS.pdf (last access: July 21, 2006).

[7] Roxana Belecheanu, Steve Munroe, Michael Luck, Tony Delaney, and Martyn Fletcher. Intelligent human variability in computer generated forces: Agent oriented software a case study. Technical report, AgentLink III, November 2005. Available at http://www.agentlink.org/resources/webCS/AL3_CS_001_AOS.pdf (last access: July 27, 2006).

[8] Roxana Belecheanu, Steve Munroe, Tim Miller, Peter McBurney, Christian Danneger, and Klaus Dorer. Living systems/adaptive transportation networks: Whitestein technologies a case study. Technical report, AgentLink III, December 2005. Available at http://www.agentlink.org/resources/webCS/AL3_CS_006_Whitestein.pdf (last access: July 24, 2006).

[9] Roxana Belecheanu, Steve Munroe, Tim Miller, Peter McBurney, and Chris van Aart. Software agents in international traffic insurance: Acklin and interpolis a business case study. Technical report, AgentLink III, November 2005. Available at `http://www.agentlink.org/resources/webCS/AL3_CS_004_Acklin.pdf` (last access: July 21, 2006).

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

[11] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *Jade programmer's guide*, February 2003.

[12] Oliver Berthold, Hannes Federrath, and Marit K ohntopp. Project 'Anonymity and Unobservability in the Internet'. In *CFP '00: Proceedings of the tenth conference on Computers, freedom and privacy*, pages 57–65, New York, NY, USA, 2000. Association for Computing Machinery (ACM) Press. Available at `http://citeseer.ist.psu.edu/berthold00project.html` (last access: May 30, 2006).

[13] 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*, volume 2009 of *Lecture Notes in Computer Science*, pages 30–45. Springer-Verlag New York, Inc., July 2000. Available at `http://www.tik.ee.ethz.ch/~weiler/lehre/netsec/Unterlagen/anon/disadvantages_berthold.pdf`.

[14] Steven Bird, Ewan Klein, and Edward Loper. Nltk tutorial: Introduction to natural language processing. Website, April 2005. Available at `http://nltk.sourceforge.net/tutorial/introduction/index.html` (last access: March 20, 2006).

[15] Niklas Borselius. Mobile agent security. *Electronics & Communication Engineering Journal*, 14(5):211–218, October 2002. Available at `citeseer.ist.psu.edu/borselius02mobile.html` (last access: June 28, 2006).

[16] Gary DeWard Brown. *System/360 job control language*. J. Wiley, New York, 1970.

[17] Antonio Carzaniga, Gian Pietro Picco, and Giovanni Vigna. Designing distributed applications with a mobile code paradigm. In *Proceedings of the 19th International Conference on Software Engineering*, Boston, MA, USA, 1997. Available at `citeseer.ist.psu.edu/carzaniga97designing.html`.

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

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

[20] 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`.

[21] Joris Claessens, Bart Preneel, and Joos Vandewalle. (How) can mobile agents do secure electronic transactions on untrusted hosts? a survey of the security issues and the current solutions. *ACM Trans. Inter. Tech.*, 3(1):28–48, 2003. Available at `http://doi.acm.org/10.1145/643477.643479` (last access: Jun 14, 2006).

[22] Antonio Corradi, Rebecca Montanari, and Cesare Stefanelli. Mobile agents protection in the internet environment. In *Proceedings of Twenty-Third Annual International Computer Software and Applications Conference*, Phoenix, Arizona, USA, October 1999. IEEE.

[23] Lance Cottrell. Mixmaster and remailer attacks, 1995.

[24] DARPA. The DARPA agent markup language homepage (DAML). Website, 2006. `http://www.w3.org/2004/OWL/` (last access: March 22, 2006).

[25] David R.A. de Groot, Frances M.T. Brazier, and Benno J. Overeinder. Cross-platform generative agent migration. In *Proceedings of the Fourth European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems*, pages 356–363, June 2004. Available at `http://www.iids.org/publicationdata/db/davidra03_1/pubdetail`.

[26] Shlomi Dolev and Rafail Ostrobsky. Xor-trees for efficient anonymous multicast and reception. *ACM Transactions on Information Systems Secururity*, 3(2):63–84, 2000. Available at `www.cs.ucla.edu/~rafail/PUBLIC/31.ps`.

[27] Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 39–48, Marina del Rey, CA, USA, February 1997. Association for Computing Machinery (ACM) Press. Available at `citeseer.ist.psu.edu/doorenbos97scalable.html` (last access: August 3, 2006).

[28] Arnoud 'Galactus' Engelfriet. Anonymity and privacy on the internet. Website, January 1997. Available at `http://www.stack.nl/~galactus/remailers/` (last access: May 31, 2006).

[29] 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.

[30] Boi Faltings. Intelligent agents: Software technology for the new millenium. *Informatik / Informatique*, 1:2–5, February 2000. Available at `http://www.svifsi.ch/revue/pages/issues/n001/no001.html` (last access: March 13, 2006).

[31] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for mobile agents: Issues and requirements, 1996. Available at `gunther.smeal.psu.edu/farmer96security.html`.

[32] 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`.

[33] Tim Finin, Jay Weber, Gio Wiederhold, Mike Genesereth, Don McKay, Rich Fritzson, Stu Shapiro, Richard Pelavin, and Jim McGuire. Specification of the KQML Agent-Communication Language – plus example agent policies and architectures, 1993. Available at `citeseer.ist.psu.edu/finin94specification.html` (last access: March 21, 2006).

[34] Foundation for Intelligent Physical Agents (FIPA). Foundation for Intelligent Physical Agents (FIPA). Website. Available at `http://www.fipa.org/` (last access: March 21, 2006).

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

[36] 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 New York, Inc. Available at `citeseer.ist.psu.edu/franklin96is.html`.

[37] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, 1998. Available at `citeseer.ist.psu.edu/fuggetta98understanding.html`.

[38] Michael R. Genesereth and Richard E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, USA, June 1992. Available at `citeseer.ist.psu.edu/article/genesereth92knowledge.html` (last access: March 21, 2006).

[39] Global IDs Inc. Global ids inc. – global data integration solutions. Website. Available at `http://www.globalids.com/` (last access: July 26, 2006).

[40] Global IDs Inc. Global data integration with autonomous mobile agents: White paper. Technical report, Global IDs Inc., Rockaway, USA, June 2002. Available at `http://www.globalids.com/Download_PDF/Global_IDs_Global_Data_Integration_White_Paper.pdf` (last access: July 26, 2006).

[41] Ian Goldberg and David Wagner. TAZ servers and the rewebber network: Enabling anonymous publishing on the world wide web. *First Monday*, 3(4), August 1998. Available at `http://www.ovmj.org/GNUnet/papers/goldberg97taz.ps`.

[42] Ian Goldberg, David Wagner, and Eric Brewer. Privacy-enhancing technologies for the internet. In *Proceedings of the 42nd IEEE Spring COMPCON*. IEEE Computer Society Press, February 1997.

[43] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer-Verlag New York, Inc., May 1996.

[44] Robert S. Gray. Agent Tcl: A flexible and secure mobile-agent system. In *Proceedings of the Fourth Annual Tcl/Tk Workshop (TCL 96)*, Monterey, CA, USA, 1996. Available at `http://www.cs.dartmouth.edu/\~{}agent/papers/tcl96.ps.Z`.

[45] Robert S. Gray, George Cybenko, David Kotz, Ronald A. Peterson, and Daniela Rus. D'agents: Applications and performance of a mobile-agent system. *Software, Practice and Experience*, 32(6):543–573, May 2002. Available at `http://citeseer.ist.psu.edu/528536.html` (last access: Jun 14, 2006).

[46] 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. Available at `citeseer.ist.psu.edu/gray00mobile.html`.

[47] grid.org. GRID.ORG – grid computing projects – home. Website, 2004. Available at `http://www.grid.org/home.html` (last access: Jun 14, 2006).

[48] Arne Grimstrup, Robert S. Gray, David Kotz, Maggie R. Breedy, Marco Carvalho, Thomas Cowin, Daria A. Chacòn, Joyce Barton, Chris Garrett, and Martin Hofmann. Toward interoperability of mobile-agent systems. In *MA '02: Proceedings of the 6th International Conference on Mobile Agents*, pages 106–120, London, UK, 2002. Springer-Verlag New York, Inc. Available at `http://www.cs.dartmouth.edu/~dfk/papers/grimstrup:gmas.pdf` (last access: March 24, 2006).

[49] 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. Available at `citeseer.nj.nec.com/2254.html`.

[50] 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`.

[51] Douglas Harper. Online etymology dictionary. Website, 2006. http://www.etymonline.com/ (last access: May 4, 2005).

[52] Fritz Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In Giovanni Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 92–113. Springer-Verlag New York, Inc., London, UK, October 1998. Available at citeseer.ist.psu.edu/hohl98time.html (last access: May 10, 2006).

[53] European Commission IST Advisory Group (ISTAG). Scenarios for ambient intelligence in 2010. Website, February 2001. Final Report. Compiled by K. Ducatel, et al. Feb-2001. EC. Brussels, 2001. Available at ftp://ftp.cordis.lu/pub/ist/docs/istagscenarios2010.pdf (last access: Jun 14, 2006).

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

[55] Dag Johansen, Robbert van Renesse, and Fred B. Schneider. An introduction to the TACOMA distributed system version 1.0. Technical Report 95-23, University of Tromsø, Norway, 1995. Available at http://www.cs.uit.no/forskning/rapporter/Reports/9523.html (last access: March 21, 2006).

[56] Günter Karjoth, N. Asokan, and C. Gülcü;. Protecting the computation results of free-roaming agents. In *MA '98: Proceedings of the Second International Workshop on Mobile Agents*, pages 195–207, London, UK, 1999. Springer-Verlag New York, Inc.

[57] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*, volume 1525 of *Lecture Notes in Computer Science*. Springer-Verlag New York, Inc., 1998.

[58] Panayiotis Kotzanikolaou, Mike Burmester, and Vassilios Chrissikopoulos. Secure transactions with mobile agents in hostile environments. In *Proceedings of ACISP*, volume 1841 of *Lecture Notes in Computer Science*. Springer-Verlag New York, Inc., 1994.

[59] Danny B. Lange and Daniel T. Chang. Programming mobile agents in java – a white paper. Technical report, IBM Corp., 1996. Available at http://www.trl.ibm.co.jp/aglets/whitepaper.htm.

[60] Samual J. Leffler, Marshall Kirk McKusick, Machael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Addison Wesley, 1989.

[61] 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.

[62] 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.

[63] 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).

[64] 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.

[65] Rafał Leszczyna and Janusz Górski. Performance analysis of untraceability protocols for mobile agents using an adaptable framework. In Peter Stone and Gerhard Weiss, editors, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06)*, pages 1063–1070, New York, NY, USA, May 2006. Association for Computing Machinery (ACM) Press.

[66] Rafał Leszczyna and Janusz Górski. An untraceability protocol for mobile agents and its enhanced security study. In *15th EICAR Annual Conference Proceedings*, pages 26–37, Hamburg, Germany, 29 April – 2 May 2006.

[67] 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).

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

[69] Michael Luck and Mark d'Inverno. A conceptual framework for agent definition and development. *The Computer Journal*, 44(1):1–20, 2001. Available at `citeseer.ist.psu.edu/luck01conceptual.html` (last access: March 21, 2006).

[70] Michael Luck, Peter McBurney, Onn Shehory, Steve Willmott, and the AgentLink Community. *Agent Technology Roadmap: A Roadmap for Agent Based Computing*. AgentLink III, September 2005.

[71] David Mazières and M. Frans Kaashoek. The design, implementation and operation of an email pseudonym server. In *Proceedings of the 5th ACM*

Conference on Computer and Communications Security (CCS 1998). Association for Computing Machinery (ACM) Press, November 1998. Available at `ftp://cag.lcs.mit.edu/pub/dm/papers/mazieres:pnym.pdf`.

[72] Catherine Meadows. Detecting attacks on mobile agents. In *Proceedings of Foundations for Secure Mobile Code Workshop*, pages 64–65, Monterey, CA, USA, March 1997. Available at `citeseer.ist.psu.edu/meadows97detecting.html` (last access: May 12, 2006).

[73] Merriam-Webster Incorporated. Merriam-Webster Online. Website, 2006. `http://www.m-w.com/` (last access: May 4, 2005).

[74] Tim Miller, Peter McBurney, Paul Burghardt, and Chris van Aart. Multi-agent systems in crisis management: The combined systems a case study. Technical report, AgentLink III, December 2005. Available at `http://www.agentlink.org/resources/webCS/AL3_CS_007_Combined.pdf` (last access: July 24, 2006).

[75] Tim Miller, Peter McBurney, Steve Munroe, Michael Luck, and Nadezhda Yakounina. Intelligent fleet cargo scheduling: Magenta technology and tankers international a case study. Technical report, AgentLink III, November 2005. Available at `http://www.agentlink.org/resources/webCS/AL3_CS_003_Magenta.pdf` (last access: July 21, 2006).

[76] Dejan S. Milojicic. Trend wars: Mobile agent applications. *IEEE Concurrency*, 7(3):80–90, 1999. Available at `http://dlib.computer.org/pd/books/pd1999/pdf/p3080.pdf`.

[77] Marvin Minsky. *The society of mind*. Simon & Schuster, Inc., New York, NY, USA, 1986.

[78] Yaron Minsky, Robbert van Renesse, Fred B. Schneider, and Scott D. Stoller. Cryptographic support for fault-tolerant distributed computing. In *EW 7: Proceedings of the 7th workshop on ACM SIGOPS European workshop*, pages 109–114, New York, NY, USA, July 1996. Association for Computing Machinery (ACM) Press. Available at `http://doi.acm.org/10.1145/504450.504472` (last access: June 28, 2006).

[79] David Molnar and Michael J. Freedman. The free haven project: Related works: Anonymous communications systems. Website, July 2000. Available at `http://www.freehaven.net/links/related-comm.html` (last access: May 23, 2006).

[80] Steve Munroe, Michael Luck, and Peet van Tooren. Agents for intelligent communications systems almende: A case study. Technical report, AgentLink III, November 2005. Available at `http://www.agentlink.org/resources/webCS/AL3_CS_008_Almende.pdf` (last access: July 24, 2006).

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

[82] 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.

[83] George C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM Symposium on Principles of Programming Languages*, Paris, France, January 1997. Available at `http://www.cs.cmu.edu/\~{}necula/popl97.ps.gz`.

[84] Nils J. Nilsson. Shakey the robot. Technical Report 323, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, April 1984. Available at `http://www.ai.sri.com/shakey/` (last access: March 13, 2006).

[85] Object Management Group (OMG). Website. http://www.omg.org/.

[86] Object Management Group (OMG). Mobile Agent System Interoperability Facilities Specification, November 1997. Available at `http://www.omg.org/cgi-bin/doc?orbos/97-10-05` (last access: March 21, 2006).

[87] Object Management Group (OMG). Agent technology — green paper. OMG Document agent/00-09-01, August 2000. `citeseer.ist.psu.edu/group00agent.html`.

[88] James Odell. Introduction to agents. Website, 2000. Available at `http://www.objs.com/agent/agents_omg.pdf`.

[89] J. J. Ordille. When agents roam, who can you trust? In *First Conference on Emerging Technologies and Applications in Communications (etaCOM)*, Portland, OR, 1996. Available at `citeseer.ist.psu.edu/ordille96when.html` (last access: March 24, 2006).

[90] Julian Perkin. Ai - software agents 'negotiate' production schedules. Website, November 2000. Available at `http://specials.ft.com/ftit/november2000/FT3KXITVWEC.html` (last access: Jun 14, 2006).

[91] 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).

[92] Andreas Pfitzmann and Michael Waidner. Networks without user observability – design options. *Computers and Security*, 6(2):158–166, April 1987. Available at `http://www.semper.org/sirene/publ/PfWa_86anonyNetze.html` (last access: May 18, 2006).

[93] 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 *Lecture Notes in Computer Science*, pages 10–29, New York, NY, USA, 2001. Springer-Verlag New York, Inc. Available at `citeseer.ist.psu.edu/454354.html`.

[94] 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. `citeseer.ist.psu.edu/reed98anonymous.html`.

[95] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998. Available at `http://avirubin.com/crowds.pdf`.

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

[97] James Riordan and Bruce Schneier. Environmental key generation towards clueless agents. In Giovanni Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*. Springer-Verlag New York, Inc., London, UK, April 1998.

[98] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. In Giovanni Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 379–386. Springer-Verlag New York, Inc., London, UK, April 1998.

[99] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, New York, 1st edition, 1994.

[100] SRI International. Timeline of SRI international innovations. Website, 2006. Available at `http://www.sri.com/about/timeline/` (last access: March 20, 2006).

[101] Heiko Stamer. Dining cryptographers networks. Master's thesis, The Institute of Computer Science, Leipzig University, May 2003. Available at `http://lips.informatik.uni-leipzig.de:80/pub/2003-10/en`.

[102] M. Straßer, J. Baumann, and F. Hohl. Mole – a java based mobile agent system. In *Proceedings of the 2nd ECOOP Workshop on Mobile Object Systems*, University of Linz, Austria, 1996.

[103] Paul Syverson, Michael Reed, and David Goldschlag. Onion Routing access configurations. In *DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, volume 1, pages 34–40. IEEE Computer Society Press, 2000. Available at `citeseer.ist.psu.edu/ablayev92lower.html`.

[104] 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 *Lecture Notes in Computer Science*, pages 96–114, New York, NY, USA, July 2000. Springer-Verlag New York, Inc.

[105] Stephen R. Tate and Ke Xu. Mobile agent security through multi-agent cryptographic protocols. In *Proceedings of The 4th International Conference on Internet Computing*, pages 462–468, 2001.

[106] Telecom Italia Lab. Java Agent DEvelopment Framework. Website. `http://jade.tilab.com/`.

[107] The RoboCup Federation. RoboCupRescue – official web page. Website. Available at `http://www.rescuesystem.org/robocuprescue/` (last access: July 27, 2006).

[108] Pavel Tichý, Vladimír Mařík, Pavel Vrba, and Michal Pěchouček. Chilled water system automation: Rockwell automation a case study. Technical report, AgentLink III, December 2005. Available at `http://www.agentlink.org/resources/webCS/AL3_CS_005_Rockwell.pdf` (last access: July 24, 2006).

[109] TrueForce. History timeline of robotics. Website, 2006. Available at `http://trueforce.com/Articles/Robot_History.htm` (last access: March 20, 2006).

[110] Trusted Computing Group. Trusted Computing Grou: Home. Website. Available at `https://www.trustedcomputinggroup.org`.

[111] Giovanni Vigna. Protecting mobile agents through tracing. In *Third Workshop on Mobile Object Systems*, 1997. Available at `https://citeseer.ist.psu.edu/vigna97protecting.html`.

[112] W3C. World Wide Web Consortium (W3C). Website. Available at `http://www.w3.org/` (last access: March 21, 2006).

[113] W3C. Web Ontology Language (OWL). Website, 2004. Available at `http://www.w3.org/2004/OWL/` (last access: March 21, 2006).

[114] Michael Waidner. Unconditional sender and recipient untraceability in spite of active attacks. In J.-J. Quisquater and J. Vandewalle, editors, *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, volume 434 of *Lecture Notes in Computer Science*, pages 302–319. Springer-Verlag New York, Inc., 1990. Available at `citeseer.ist.psu.edu/waidner89unconditional.html` (last access: May 30, 2006).

[115] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure servicability. In *Proceedings of EUROCRYPT 1989*, volume 434 of *Lecture Notes in Computer Science*. Springer-Verlag New York, Inc., 1990.

[116] Joseph Weizenbaum. Eliza a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 26(1):23–28, 1983. Available at `http://doi.acm.org/10.1145/357980.357991`.

[117] 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 New York, Inc., 2000.

[118] J. E. White. Telescript technology: The foundation for the electronic marketplace. White paper, General Magic, Inc., Mountain View, CA, USA, 1994.

[119] Wikipedia. Knowledge representation. Website, 2006. Available at `http://en.wikipedia.org/wiki/Knowledge_representation` (last access: March 20, 2006).

[120] Uwe G. Wilhelm. *A Technical Approach to Privacy based on Mobile Agents Protected by Tamper-resistant Hardware*. PhD thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 1999. Available at `http://library.epfl.ch/en/theses/?nr=1961` (last access: March 20, 2006).

[121] Uwe G. Wilhelm, Sebastian Staamann, and Levente Buttyn. Protecting the itinerary of mobile agents. In *Proceedings of 4th ECOOP Workshop on Mobility: Secure Internet Mobile Computations*, 1998.

[122] Terry Winograd. A procedural model of language understanding. In Roger C. Schank and Kenneth M. Colby, editors, *Computer Models of Thought and Language*, pages 152—-186. W. H. Freeman & Co., New York, NY, USA, August 1973.

[123] D. Wong, N. Paciorek, T. Walsh, and J. DiCelie. Concordia: An infrastructure for collaborating mobile agents. In K. Rothermel and R. Popescu-Zeletin, editors, *Proceedings of the First International Workshop on Mobile Agents*, volume 1219 of *Lecture Notes in Computer Science*, pages 86–97. mobile agents, 1997.

[124] Bennet Yee and J. D. Tygar. Secure coprocessors in electronic commerce applications. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, New York, USA, July 1995.

[125] Bennet S. Yee. A sanctuary for mobile agents. In *Proceedings of the DARPA Workshop on Foundations for Secure Mobile Code*, Monterey, USA, March 1997. Available at `citeseer.ist.psu.edu/article/yee97sanctuary.html` (last access: May 08, 2006).

[126] John Zachary. Protecting mobile code in the wild. *IEEE Internet Computing*, pages 78–82, March and April 2003.

European Commission

EUR 22485 EN – DG Joint Research Centre, Institute for the Protection and Security of the Citizen
Title: Privacy in Mobile Agent Systems: Untraceability
Authors: Rafał Leszczyna
Luxembourg: Office for Official Publications of the European Communities
2007 – 56 pp. – 21.0 x 29.7 cm
EUR - Scientific and Technical Research series; ISSN 1018-5593

Abstract

Agent based Internet environments are an interesting alternative to existing approaches of building software systems. The enabling feature of agents is that they allow software development based on the abstraction (a 'metaphor') of elements of the real world. In other words, they allow building software systems, which work as human societies, in which members share products and services, cooperate or compete with each other. Organisational, behavioural and functional models etc applied into the systems can be copied from the real world.

The growing interest in agent technologies in the European Union was expressed through the foundation of the Coordination Action for Agent-Based Computing, funded under the European Commission's Sixth Framework Programme (FP6). The action, called AgentLink III is run by the Information Society Technologies (IST) programme. The long-term goal of AgentLink is to put Europe at the leading edge of international competitiveness in this increasingly important area.

According to AgentLink 'Roadmap for Agent Based Computing' agent-based systems are perceived as 'one of the most vibrant and important areas of research and development to have emerged in information technology in recent years, underpinning many aspects of broader information society technologies.'

However, with the emergence of the new paradigm, came also new challenges. One of them is that agent environments, especially those which allow for mobility of agents, are much more difficult to protect from intruders than conventional systems. Agent environments still lack sufficient and effective solutions to assure their security. The problem which till now has not been addressed sufficiently in agent-based systems is privacy, and particularly the anonymity of agent users. Although anonymity was studied extensively for traditional message-based communication for which during the past twenty five years various techniques have been proposed, for agent systems this problem has never been directly addressed.

The research presented in this report aimed at filling this gap. This report summarises results of studies aiming at the identification of threats to privacy in agent-based systems and the methods of their protection.

EUROPEAN COMMISSION
DIRECTORATE-GENERAL
Joint Research Centre

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

Publications Office
Publications.eu.int