Web Intelligence and Agent Systems: An International Journal 0 (2008) 1–13
IOS Press

Anonymity Services for Multi-Agent Systems

Martijn Warnier and Frances Brazier
Section Systems Engineering
Faculty of Technology, Policy and Management
Delft University of Technology
Jaffalaan 5, 2628BX Delft, The Netherlands
E-mail: {M.E.Warnier,F.M.T.Brazier}@tudelft.nl *

Abstract. Anonymity can be of great importance in distributed agent applications such as e-commerce & auctions. This paper proposes and analyzes a new approach for anonymous communication of agents based on the use of handles as pseudonyms. A novel naming scheme is presented that can be used by agent platforms to provide automatic anonymity of communication for *all* agents on its platform, or, alternatively, to provide anonymity *on demand*. The paper furthermore introduces new approaches that provide authentication and anonymous payment schemes for agents. Performance measures for an anonymity service implemented for the AgentScape platform provides some insight in the overhead involved.

Keywords: Multi-Agent Systems, Anonymity, AgentScape, Agent Middleware

1. Introduction

Agent technology provides state-of-the-art solutions for distributed applications such as e-commerce, e-health, e-government and electronic auctions [16,31]. The sensitive nature of data in these domains makes privacy an important requirement.

Anonymity in multi-agent systems can be acquired by conventional methods such as the use of a mediator or another outside trusted third party. In these methods a mediator or trusted third party acts on behalf of an agent (and its owner) and relays messages without revealing an agents identity. However, these methods require an explicit effort on the part of an agent application developer. A developer not only needs to design the application in such a way that all communication is relayed to a mediator, special care also needs to be taken to ensure that no information regarding the agent's identity is leaked in the process. Note that for certain classes of applications complete anonymity is undesirable.

This paper proposes a new approach to anonymous agent-to-agent communication that guarantees anonymity for all agents running on a platform without any additional effort by agent application developers. In addition, a number of extensions to this proposal are proposed for anonymous currency and authentication. The one main assumption is that agents trust the middleware (the agent platform) on which they run.[1] The middleware is (by definition) trusted and can thus be trusted to keep information about the relation between an agent and its owner confidential, thereby providing anonymity for the agent owner.

Anonymity is, of course, not an absolute notion, one's communication can be anonymous to one person and not to another. It is often not necessary or desirable to make communication with all parties anonymous. In fact, in many cases an agents may wish to reveal its identity. The reasons may vary considerably:

*This paper is an extension and revision of a paper previously presented at the Third International Symposium on Information Assurance and Security [30].

[1] This form of anonymity is sometimes referred to as semi-anonymity.

to access information services for which intellectual property rights play a role, to negotiate a time slot with another agent, to collaborate with other agents to perform a specific task. An agent may also wish to establish several (virtual) identities for different situations. Such cases require a more fine-grained solution than fully automatic anonymization. Of the several degrees of anonymity that can be distinguished, ranging from full anonymity to total non-repudiation [2,10,23], this paper focuses on *pseudonym based anonymity*. The anonymity scheme this paper introduces ensures that each agent's true identity and its pseudonyms are unique and cannot be linked to each other by any outside party.

A typical application for the proposed anonymity scheme is an online marketplace, hosted by an independent organization or company, such as eBay. Agents can run in this marketplace and try to find the lowest price for a certain item. Anonymity is important to prevent sellers from conspiring against an agent to (artificially) raise the price. Anonymous communication is then essential. If anonymity over a longer time period is required anonymous communication alone no longer suffices: items need to be paid, agents need to be authenticated to be able to use (paid) services. There are currently no suitable alternatives for anonymous currencies in the digital world. These issues are discussed in more detail in Section 8.

The next section provides a more detailed explanation of anonymity as commonly defined in Computer Science focusing on its relevance to distributed agent systems. The section also gives a brief introduction to agent systems, focusing on the AgentScape platform. Section 3 discusses the advantage of anonymity in multi-agent systems and gives examples of typical scenarios. Section 4 introduces a naming scheme for anonymity, together with an approach with which communication anonymity in agent systems can be acquired. The range of options: from complete integration in agent platform middleware to solutions based on individual agent implementations are discussed. Section 5 briefly discusses the actual implementation in AgentScape of an anonymizing service and Section 6 illustrates the performance overhead to be expected. Section 7 describes two alternative approaches in relation to our work. Section 8 illustrates how anonymous authentication and anonymous (digital) payments can be used in multi-agent systems. The paper ends with a discussion and conclusions.

2. Background

This section provides background on anonymity as defined in Computer Science. Some of the major advantages and disadvantages of anonymous currencies are discussed. Multi-Agent Systems are introduced. The AgentScape platform is used to illustrate the discussed approaches in the remainder of this paper.

2.1. Anonymity

Classical anonymity in Computer Systems focuses on anonymity of the underlying communication layer [27], i.e., anonymous communication. The typical goal is to anonymously browse the Internet, or communicate with other parties without revealing the parties true identity. Using terminology proposed by Pfitzmann and Köhntopp anonymity is defined as:

> Anonymity is the state of being not identifiable within a set of agents, the *anonymity* set. [23]

The anonymity set is the set of all agents running on an agent platform. Thus, agents are anonymous with respect to other agents. Three related notions can be used to further describe different types of anonymity:

- sender anonymity
- recipient anonymity
- unlinkability

The first two, sender and receiver anonymity, require that the sender and receiver, respectively, are not known (anonymous) to the other communicating party. Thus an agent has the sender anonymity property if it is impossible to identify the agent as the sender of a certain message in the anonymity set. An equivalent property holds for recipient anonymity. Unlinkability, also known as link anonymity, ensures that the link between two agents or messages in the system remains anonymous to all third parties: it is impossible for any outside party to observe if two parties are communicating with each other. And, similarly, it is impossible for an outside party to know if two messages were sent by the same agent. Note that the communicating parties themselves are often aware of the (true) identity of the other party. In practice these forms of anonymity can be combined.

This general notion of anonymity in Computer Science can be applied to agent technology. The focus is on anonymity of communication between individual agents. As stated earlier the relation between agent owner and agent is assumed to be confidential, and

guaranteed by the agent platform. Full communication anonymity, i.e., receiver, sender and link anonymity taken together, can only be established when an agent cannot be linked to a legal entity, either directly or indirectly via its communication with other agents. A platform based solution that enables the middleware to (automatically) provide link anonymity and location anonymity for each individual agent is the main focus of this paper. Pseudonyms are introduced for this purpose.

Using multiple (online) identities, i.e., pseudonyms, can be useful for many purposes [23]. For example, if an agent uses the same pseudonym more then once the pseudonym can be used to build a reputation. The use of a pseudonym, however, on its own does not suffice for anonymity. If outsiders can observe agent communication, these observations can be used to obtain/deduce (unwanted) information about an agent. If an agent uses the same pseudonym to communicate with several other agents, together they can infer that they have been talking to the same party, breaking anonymity.

A class of applications for which anonymity can be useful is that of mobile agents in open environments. If a mobile agent is malicious it can run havoc on an agent platform. Agent owners cannot be traced, the agent platform owner is unable to take any (legal) action against the agent owner, nor can it recognize future malicious agents coming from the same source. Note that In agent systems that support mobility of agents another form of anonymity exists: migration anonymity. In essence this hides the migration path of an agent, and thus hides the original starting platform of an agent which results in a form of anonymity. Migration anonymity is, however, outside the scope of this paper, see Rafal Leszczyna and Janusz Górski [15] for more detail on migration anonymity.

2.2. Anonymous Currency

Anonymous electronic payment systems [8] build on the anonymous communication principles explained in the previous section. Such systems are needed if privacy and anonymity are required in a commercial setting such as a market place or auction site. One obvious solution is to use a third party (a middle man) for the payment of services and/or goods. In principle banks or online payment systems such as PayPal are not required to give information about the buyers identity. As long as third parties can be trusted not to reveal an agent's identity this works. However, in practice these third parties typically *do* provide information about a customer's identity [3].

This is partly due to security requirements: fraud is a lot easier with anonymous electronic payment [26]. It hinders law enforcement agencies to operate against money laundering practices. In this respect the trend seems to be in the other direction, as can, for example, be seen by the European Central Bank's proposal to use RFID [7] chips in Euro bills,[2] making normal cash no longer anonymous in use.

Another reason against anonymous currencies is that service providers typically want to know a consumer's identity, for example for marketing purposes. Third party facilitators (banks, PayPal) typically have a good relation with service providers and are often willing to help identify 'interesting' customers. Hence, anonymous digital currencies are not very widely accepted nor implemented. Clearly anonymous digital currencies still have a long way to go, but the main problem seems to be more procedural than technical.

For these reasons the solutions proposed in Section 8 all use the (trusted) agent middleware as a trusted third party. In principle, the middleware does not provide any details about an agent's identity to outside parties. However, if need be, the middleware can keep a log of all actions. This log can then at a later stage be used for auditing purposes by, for example, law enforcement agencies if so authorized by a Court of Law.

2.3. AgentScape

The multi-agent platform AgentScape supports agents as autonomous processes. A uniform middleware layer provides an agent runtime that is available at numerous heterogeneous platforms. Other examples of such systems are JADE [1], SeMoA [24] and Cougaar [11].

Within AgentScape, *agents* are active entities that reside within *locations*, and *services* are external software systems accessed by agents hosted by the AgentScape middleware (see Figure 1). Agents in AgentScape can communicate with other agents and can access services. Agents may migrate from one location to another.

All operations of agents are modulo authorisation and security precautions. For example, an agent should have the appropriate credentials (ownership, authorisa-

[2] See for example, `http://www.eetimes.com/story/OEG20011219S0016`.

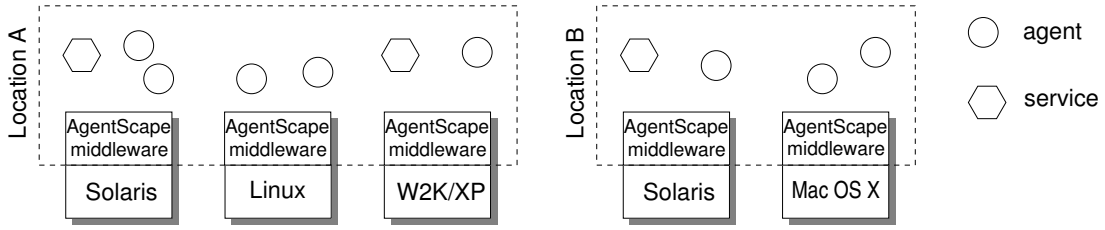

Fig. 1. Conceptual model of the AgentScape middleware environment.

tion, access to resources, and so on) to access a specific service, possibly for a limited time period.

The leading principle in the design of the AgentScape middleware has been to develop a minimal but sufficient open agent platform that can be extended to incorporate new functionality or adopt (new) standards into the platform. This design principle has resulted in a multi-layered architecture with (1) a small *middleware kernel*, called the AgentScape Operating System (AOS) kernel [29], that implements basic mechanisms, (2) high-level *middleware services* that implement agent platform specific functionality and policies (see Figure 2) and (3) external (to the middleware) directory services. This approach simplified the design of the kernel and has made it less vulnerable to errors or improper functioning. The current set of middleware services includes agent servers, host managers, location managers, a lookup service and a web service gateway.

AgentScape's middleware services implement the agent specific functionality. The current set of middleware services include:

- **Location Manager** - Each *location* has a Location Manager, which runs on one of the hosts within that location. This process manages that location's hosts. Locations are typically formed by hosts that belong to one single administrative domain.
- **Host Manager** - Each host (typically, one physical machine) runs a Host Manager. This process is responsible for managing the middleware components running on that host. It also regulates and guards access to its resources.
- **Agent Server** - An AgentServer provides a runtime environment for agents. Each host can run one or more AgentServers to host agents supporting e.g. different programming languages.
- **Web Service Gateway** - The Web Service Gateway enables agents to communicate with web services using the SOAP/XML protocol [21].
- **Lookup Server** - This external (to the middleware) service keeps track of the current location of agents. Strictly speaking, this service is not part of the AgentScape middleware as it can be run as a stand-alone application. Two versions exist, a centralised, unsecured version and a decentralised secured one [20].

Agent Servers provide agent access to the AgentScape middleware (see Figure 2). Multiple code base support in AgentScape is achieved through the provision of multiple agent servers, at least one per code base. From a security perspective, it is important to note that agent servers 'sandbox' agents.

The Location Manager is the coordinating entity in an AgentScape location (managing one or more hosts in its location). Note that for fault tolerance a location manager may be replicated, but that is beyond the scope of the paper, see [18]. Agent creation, migration, and all policy related issues relevant in the context of a location, are managed or coordinated by the location manager. The Host Manager manages and coordinates activities on a host. It acts as the local representative of the Location Manager, but is also responsible for local (at the host) resource access and management. The policies and mechanisms of the Location and Host Manager infrastructure are based on negotiation and service level agreements [19].

3. Privacy through anonymous agents

Both content providers and (ordinary) users can have a need or desire for privacy. This paper focuses on the privacy of individual users. Privacy protection of (privacy sensitive) databases (inference control [9,25]) by content providers and/or administrators is a different problem outside the scope of this article.

Software agents act on behalf of users. If a user's identity is linked to its agent(s), content providers (and others) can use this information, thereby breaking a

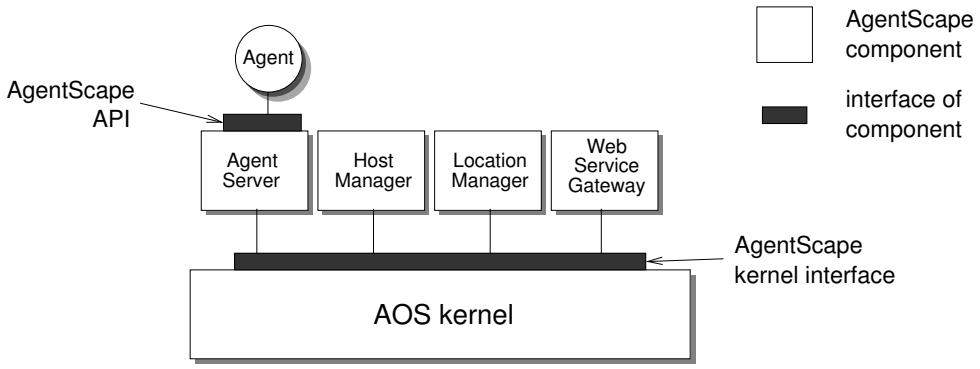

Fig. 2. The AgentScape software architecture.

user's privacy. Buyer profiling, explained in Example 1 below, is an example hereof:

Example 1
Mobile software agents are envisioned to play an important role in future e-commerce systems [16]. Such agents search for products based on users' preferences and purchase (possibly after consulting with their owners) goods and services for their owners. In such environments Buyer Profiling *can be very lucrative for service providers. Customized advertisements, based on past purchases by agents, can be deployed to lure agents (and their owners) into buying other products. Profiles of users can also be sold to third parties. Buyer profiling, and especially re-selling profiles, without the explicit consent of a user is a clear violation of user privacy regulations.*

Anonymous agents can be used as a means to acquire privacy for users, e.g. against buyer profiling. Agent auction systems [33] provide another opportunity for the use of anonymous agents. Example 2 below sketches such a typical scenario concerning agent auction systems.

Example 2
In agent auction systems, analogous to 'real world' auction systems, buyers may want to hide their true identity, e.g. due to the expected effect on price development. In such cases their agents should be anonymous with respect to other agents. *In 'real world' auctions this might be facilitated by a buyer bidding over a phone.*

Another possible solution to this problem is to use a mediator that acts on behalf of a buyer. As long as the mediator uses its own agent, anonymity of the real buyer can be guaranteed. Of course, if the mediator is well known, e.g. as a representative of wealthy clients, others might infer this, thereby canceling the original reason for the mediator.

A better solution is to let the auction organizer use an agent platform that facilitates automatic anonymity for its agents. In such an agent system the identity of the agent owners is known to the agent platform (and thus the auction organizer), but the anonymity of the agents with respect to each other can be guaranteed. Agents can communicate with each other, without revealing their identity. They can even use different pseudonyms for each separate auction event, which ensures that other agents cannot infer information during an auction (something which is quite hard to facilitate in 'real world' auctions). This is also know as using transaction pseudonyms *[23].*

As the auction organizer has to know the identity of agent owners, e.g. to check their credit or to deliver bought goods etc, an agent platform that can ensure this form of anonymity *is more suitable to agent auction systems.*

The remainder of this paper discusses how an agent platform can be implemented that adheres to the restrictions posed in Example 1 and 2. The next sections explain some of the subtleties involved.

4. Anonymous Communication in Agent Systems

Anonymous communication forms the cornerstone for anonymity of agents in this paper. This section describes how anonymous communication is achieved.

4.1. A naming scheme for anonymity

An agent platform can identify an agent by its *globally unique identifier*(GUID). This GUID corresponds

uniquely with the identity of the agent. The following example illustrates how the use of one single pseudonym for all communication does not suffice in multi-party negotiation situations:

Example 3
There are three agents A, B and C, each with their own pseudonym P_A, P_B and P_C respectively. Agent A considers obtaining services from either agent B or C. It first uses its pseudonym P_A to communicate with agent B and asks agent B the price of its service, then agent A uses the same pseudonym P_A to ask agent C the price of its services. Although agent B and agent C do not know A's real identity, together they can still determine that the same agent *has been asking price information of the services they provide. Thus agent A has not been communicating anonymously.*

Example 3 above illustrates that privacy protection against buyer profiling (as discussed in Example 1) cannot be obtained by only using *one* pseudonym. As the same pseudonym can be linked to multiple events over a longer period of time, a buyer profile can be constructed, and privacy cannot be guaranteed. Even if the 'real' identity of the agent owner is not known!
The example also clearly demonstrates the need for agents to use more than one pseudonym to obtain (link) anonymity - one pseudonym for each individual communication event (or communication session). For similar reasons, agents should also use a different pseudonym each time they communicate with the same party at some later point in time.

In our approach each agent has one globally unique identifier and multiple unique pseudonyms. The agent platform is responsible for ensuring that all GUID's and pseudonyms are unique, that GUID's are hidden from other agents, that pseudonyms cannot be linked to each other and that pseudonyms can not be linked to the agent they represent.

4.2. Using handles for anonymity

Each agent is assumed to have a globally unique identifier (GUID) known only to the agent platform. Such a GUID can, for example, be implemented by a Universally Unique Identifier (UUID, ISO 11578:1996). Furthermore, each agent can acquire as many (globally unique) handles as it requires. These handles serve as pseudonyms and are used for communication purposes.

As handles have no intrinsic meaning and they do not leak any information about an agent or its owner, agents can safely use handles as pseudonyms. Link anonymity is acquired if agents use a new handle for each individual communication event.

The agent platform is responsible for creation of agent GUID's and handles. Handles and GUID's should be related. An agent platform should be able to check if the handle that an agent uses indeed belongs to it. However, others (other agents in particular) should not be able to determine if two handles belong to the same agent. The binding between handles and GUID's can be acquired using a cryptographic hash function [12]. The following algorithm can, for example, be used by an agent platform to generate handles for agents, where 'sha' is a cryptographic hash function:

Protocol 1 *Handle generation*

$handle_1$ = *sha(*GUID + *1)*
$handle_2$ = *sha(*GUID + *2)*
...
or more generally stated:

$handle_n$ = *sha(*GUID + *n) with* $n \in \mathbb{N}^+$

Using Protocol 1 has two specific advantages:

1. If the GUID is not known then handles cannot be linked to each other or one specific GUID.
2. If the GUID is known then the platform cannot deny that a specific handle belongs to a specific GUID.

Several properties of cryptographic hashes, such as Sha-1 and MD5 are used: First the fact that cryptographic hashes are *one way* is used, i.e., easy to compute but very hard (computably infeasible) to reverse. This guarantees that if an agent knows a handle it can not compute the corresponding GUID. Cryptographic hashes also have the property that they are *collision free*, i.e., it is computably infeasible to construct two different GUID's that have the same hash-image. This ensures that all handles are unique. Finally, cryptographic hashes also have the property that a *small change* in the input results in a big change in the output (roughly half of the bits should change). This ensures that agents can not determine if two handles belong to the same GUID (agent). In summary, the one-way property of cryptographic hash functions ensures that advantage 1 above is valid. The collision free property, which ensures uniqueness of handles, establishes advantage 2, i.e., all hashes generated from a GUID are

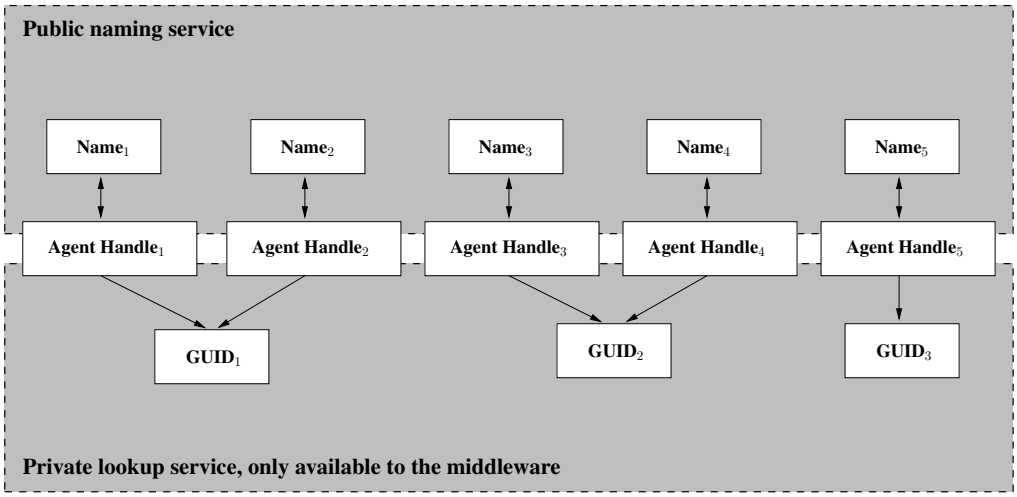

Fig. 3. Schematic overview of the handle approach

unique, therefore if the GUID is known then handles can be linked to it.

A private lookup service is necessary for platforms to be able to link handles back to GUID's. It is essential that the lookup service that maps handles to GUID's is indeed *private* to the middleware. If this is the case then link anonymity for agents can be guaranteed. Agents can, however, also use an optional human readable name, where each name corresponds with one handle. A *public* naming service can then be used to link names with handles and vice versa. The platform ensures that all human readable names are unique, bit no other limitations on human readable names are enforced. How (and if) these are used will depend on the application type. Figure 3 schematically displays the handle approach, where arrows indicate a possible lookup.

As handles are used for all communication between agents, where handles are unique meaningless strings, no information about the location of a particular agent is revealed. Hence this approach also provides location anonymity and thus also sender and receiver anonymity. Whenever two agents communicate they do not have to share the location (host) on which they reside.

Agents can implicitly revoke a handle (simply by no longer using it) or explicitly (in which case the handle is removed from the private lookup service on the platform on which it runs). By definition, handles are also unique over time, due to the large number of possible handles no handle is ever used twice. However, if so required, a time-stamping mechanism [4] limits the lifetime of individual handles and hence make handles applicable for reuse. For example, a handle can be released for a period of two days, at which point the handle is automatically revoked. If necessary, the corresponding agent can at this point ask for a new (and different) handle from the middleware and continue its job. Note that guaranteeing uniqueness of handles is important for reliability, correctness and accountability.

5. Middleware implementation

The anonymous communication scheme proposed in Section 4 above can be implemented in several ways, some fully automatic, others on demand, and all but the most fine-tuned approaches as middleware services.

An automatic solution for anonymity requires a solution that is *integrated in the middleware* of the agent system. This provides the option for all communication between agents to be completely anonymous without any additional effort by the agent developer.

At the level of the middleware *policies* can be defined that support different strategies for communication anonymization on a per agent basis. More advanced policies, e.g. supporting self-learning or self-organizing strategies for anonymization [32], can refine this further and allow agents to be anonymous to some agents while not anonymous to others.

A middleware *anonymizing service*[3] is the option this paper proposes. This dedicated service acts as a proxy, routing all communication through the service. The proxy agent uses a new handle for each new communication event. This 'gateway' approach has the obvious advantage that agent developers have more fine tuned control of anonymity. The anonymizing service can be used when circumstances so require. It may also be the default option for all communication events, depending on the policies defined, thereby integrating the service complete into the middleware.

Complete self management by an agent developer is the most finely tuned approach. Agent developers can implement anonymity themselves using handles. This allows control on a very fine scale, but it also requires the most effort on the agent developers part.

The proposed approach has the advantage that it provides an automatic solution for all agents that can easily be integrated into the agent platform middleware. Location anonymity is also guaranteed as all communication is routed via handles that do not reveal any information about an agent's location. Finally, the performance overhead of this approach is limited, see Section 6. The largest disadvantage is that the approach cannot be implemented in most existing agent platforms without significant changes to the platform's middleware.

AgentScape uses a handle-model as described above. Each agent has its own globally unique identifier (GUID). The GUID is generated upon creation of the agent and is kept private to the middleware. An agent always has an initial handle that can be linked —by the middleware– to an agent's GUID. Optionally, a name can be assigned to an agent's handle. Additional handles can be requested from the middleware. An agent's handles and names are registered in a Name Look-up Service (NLS), that is assumed to be private to the middleware. In AgentScape, two different lookup services can be used: one or more *default public lookup services*, that is completely open and accessible and *a secure, decentralized lookup service* [28] that is only accessible to the AgentScape middleware.

[3]The anonymizing service in AgentScape is implemented by an agent. However as far as other agents are concerned this service is simply part of the platform middleware. Technically this is accomplished by removing (all) the handle(s) of the anonymizing service from publicly available *name services* and publishing the handle of the anonymizing service in a *yellow page service* [17].

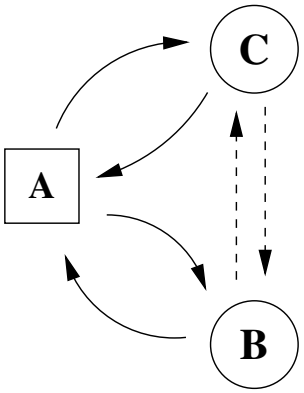

Fig. 4. Anonymizing service **A** facilitates anonymous communication for agent **B** with agent **C**. The dashed arrows represent logical communication (between agent **B** and agent **C**) and the solid arrows represent the real message flow via the anonymizer (**A**).

An anonymizing service is provided by the AgentScape platform.[4] This service is implemented by means of a simple router, depicted in Figure 4 and illustrated in the following example:

Example 4
There is one anonymizing service **A** *and two agents* **B** *and* **C** *in Figure 4 each with their own handles H_A, H_B and H_C respectively. Additional handles are numbered sequentially, thus H_{An+1} is the n_{th}+1 handle of anonymizer* **A**.

Assume that agent **B** *wants to communicate with agent* **C** *using the anonymizer* **A**. *To this purpose, in its message (the* `Envelope`*), Agent* **B** *not only specifies, the 'to' (H_C) and 'from' (H_B) handles, but also a 'via' field for the anonymizer's handle (H_A). Anonymizer* **A**, *in turn, generates a new handle H_{An} for this message, stripping agent* **B***'s handle (H_B) from the message and forwarding the message to agent* **C** *using only this new handle (H_{An}). If Agent* **C** *chooses to reply to the message, agent* **C***'s reply is sent 'to' (H_{An}), 'from' (H_C) via anonymizer* **A**. *Anonymizer* **A** *forwards the reply to agent* **B**.

Agent **B** *can decide when the communication session ends. For each new session (with the same or another agent) a new handle is generated by anonymizer* **A**.

[4]AgentScape version 0.9.0beta2 and upwards, available at `http://www.agentscape.org`, contains this anonymizing service.

Normal (non anonymized) messages in AgentScape are implemented with an `Envelope` data type. This data type has three fields:

- `fromHandle`, the handle of the sending agent.
- `toList`, a (non-empty) list of recipient agent handles.
- `data`, a data field.

An agent can send a message to (one or more) other agents by constructing an Envelop data type and sending it to the middleware delivery is ensured (assuming the recipients are valid addresses of agents).

Anonymized messages use a derived data type: the `ViaEnvelope`. This is a normal `Envelope` data type with an additional field:

- `viaHandle`, the agent handle of an anonymizing service.

The handle of an anonymizing service can be found in the lookup server. Agents can now create `ViaEnvelopes`, thereby using the anonymizing service explicitly. Alternatively, the platform administrator can enforce a policy that all communication occurs via anonymizing. In this instance all `Envelope` data structures will actually be `ViaEnvelope` data structures with an implicit anonymizing service added. The latter option has the advantage that *all* communication occurs anonymously.

The platform throughput overhead of the anonymizer service is largely determined by the network: If either the anonymizer service and the sender (**A** and **B**), or the anonymizer service and the receiver (**A** and **C**), run on the same host, this overhead is almost non-existent (in the range of 1 to 3%). The next section describes experiments that indicate the performance of the platform throughput overhead in other circumstances.

6. Experimental Validation

The experiments described in this section are all performed on nodes on the DAS-3 cluster.[5] The cluster consists of 85 dual-CPU / dual-core 2.4 GHz AMD Opteron DP 280 computers, each with 4 GB of memory and 250 GB of local HD space. The cluster is equipped with 1 and 10 Gigabit/s Ethernet, i.e., only wired interfaces are used. All machines run the same, dedicated, GNU/Linux distribution: *Scientific Linux*.[6], which is basically Enterprise Linux, recompiled from source.

Two sets of experiments have been performed. Both are designed to measure the *communications overhead* of the anonymizing service. Other types of performance overhead, such as cpu or memory consumption, are small enough ($< 0.5\%$) to be ignored in the next set of experiments. All agent platforms are run on separate hosts, thus for these experiments (in Figure 5 and 6) number of platforms equals number of hosts.

In the first set of experiments a number of 'sender' agents send messages to a number 'receiver' agents, either with or without the use of an anonymizing service. The experiment is designed to measure the platform throughput, which in our case is the total number of messages that all the receiver agents receive, on average, per second. The results are depicted in Figure 5. Different configurations, message sizes and optimizations have been used. Full optimizations means that all Java programs,[7] i.e., agent platform middleware, middleware services and agents, on one host are run in one Java virtual machine (JVM). This allows the use of shared memory and speeds up most operations, especially method calls and message sending (on one host). In the normal configuration AgentScape uses, for security reasons[8], separate JVMs for each agent, the agent platform middleware and the middleware services.

In general, these results show that the *platform throughput* remains the same, with or without (possibly multiple) anonymizing services. Another notable, but unsurprising, conclusion is that both message size and optimization (on or off) effect the message throughput.

However, if one considers the number of messages sent and received (ping-pong style, i.e., sender-receiver-sender), then the non-anonymizing communication outperforms the anonymized communication by a factor two. This is not unexpected, as in the latter case twice the number of messages are sent, i.e., sender-receiver-sender versus sender-anonymizer-receiver-anonymizer-sender. Figure 6 shows these results for one of the three configurations of Figure 5, the other results are similar and are not shown.

If this overhead for anonymous communication is acceptable depends on the specific (type of) applica-

[5] `http://www.cs.vu.nl/das3`

[6] `https://www.scientificlinux.org/`

[7] Recall that AgentScape is, amongst others, implemented in Java, which was used for these experiments.

[8] Each JVM runs its own sandbox, which makes it impossible for malicious agents to effect critical parts of AgentScape that run in separate JVMs.

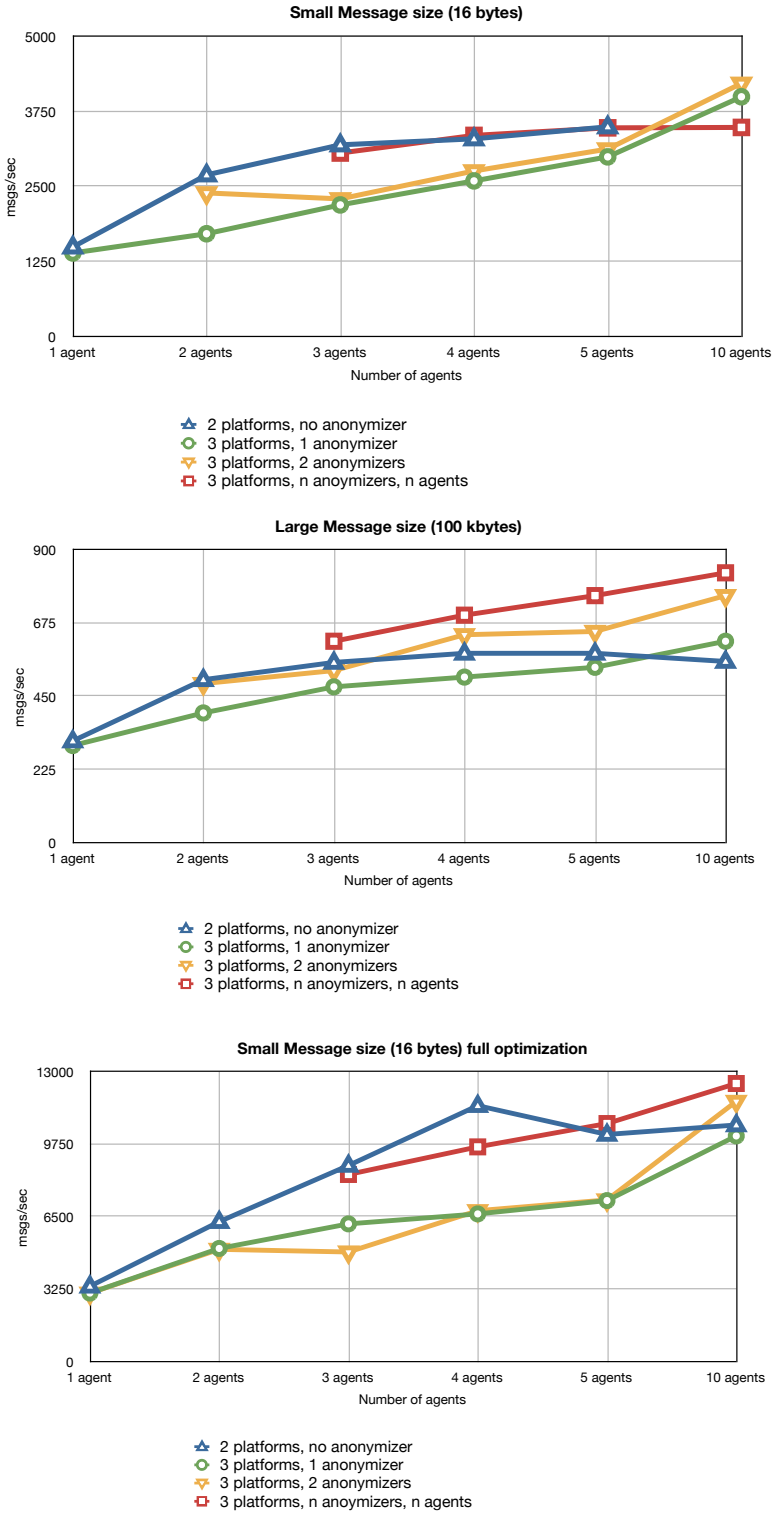

Fig. 5. Throughput (in messages per second) for anonymizing service compared to non-anonymizing communication.

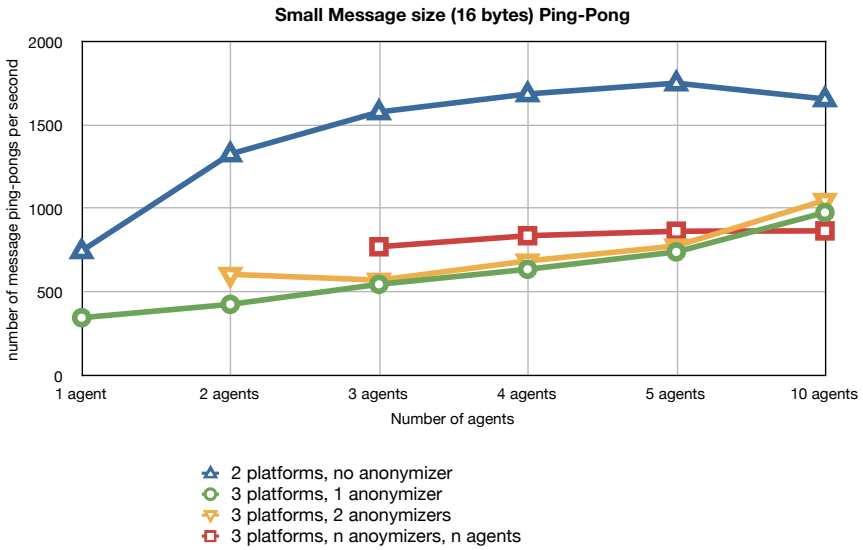

Fig. 6. Number of 'ping-pong' style messages send per second.Comparing several anonymizing configurations with non-anonymizing communication.

tion. However, improved security always comes at the price of additional overhead.

7. Alternative approaches

There are two alternative approaches for anonymous communication in agent platforms, either of which has been implemented in AgentScape which makes experimental comparison impossible. However, some theoretical arguments can be given that suggest that the current implementation in AgentScape has the lowest overhead.

7.1. Onion routing in JADE

Korba, Song and Yee [13] use onion routing [5] for all inter-agent communication. In essence, onion routing provides anonymous communication by redirecting messages via a number of routers using an unpredictable path. Their approach is implemented in the JADE agent platform [1] where a dedicated communication layer facilitates all anonymous inter-agent communication. This approach can guarantee the same level of anonymity as our own. Our approach however is a dedicated solution for agents that can be fine-tuned to meet a range of agent specific settings, ranging from automatic anonymous communication between all agents to agents that manually decide for each individual message if it should be anonymous or not.

Moreover, onion routing uses a number of different hosts, typically four or more, to route messages anonymously from one agent to another. As our approach in the worst case only uses three distinct hosts,[9] one for the sender, one for the receiver and one for the anonymizer, the total number of send messages is lower in our approach.

7.2. Using agent spawning for anonymity

Our approach requires the use of handles, which is not available on most agent platforms.

In these cases an alternative technique can be used to acquire anonymity. For this technique to work the agent platform should allow agents to spawn their own (other) agents and it should not be possible to determine which agents are related via spawning, similar to the (secret) relation between agent GUID and agent handle . Thus if an agent is spawned then other agents should not be able to tell whether the agent really is a spawned agent and which agent created it.

If these requirements are met then agents can use spawned agents for communication purposes. In essence, a spawned agent is used as a pseudonym for its original agent. The spawned agent works as a sim-

[9]This number reduces to two if one runs an anonymizing service on each hosts and requires that the sender uses the anonymizing service running on the same host.

ple proxy whose sole task it is to redirect messages to other agents and back to the originating agent. In principle, an agent can spawn a new agent for each communication event and hence can be anonymous with respect to other agents. Note that in this technique spawned agents are basically used as handles. Again, the agent platform is the only party that may be aware of the link between spawned and 'original' agents, which is similar to the private lookup service in used in our approach, as explained in Section 4.

Conceptually this approach is the same as the one proposed in this paper. However, the overhead (with regards to computing resources) is higher, as it is more expensive to create a new agent, in essence a `Thread` (group), than it is to create a new handle, which is in essence a pointer. The communication overhead is comparable.

8. Extension: Anonymous Authentication and Payment

As briefly mentioned in the introduction and background sections, if anonymity is required over a longer time period anonymous communication often does not suffice: goods and services need to be paid and there are no accepted alternatives for anonymous currencies in the digital world. To use a payment service an agent typically needs to authenticate itself (to the service) to prove it has indeed paid for the offered services. This section discusses these two problems of anonymous authentication and payment for agents and proposes a solution.

8.1. Anonymous Authentication

Anonymous agents are particularly useful when deployed in open environments. However, if agents need to authenticate themselves, e.g. to obtain access to a service, the anonymity of individual agents, and thus the privacy of its users, can no longer be upheld. A technical solution for this problem focuses on the use of *derived access tokens* that can be used to gain access to a services without revealing ones 'true' identity to the service provider. A trusted intermediate, *the trusted third party*, is used to authenticate an agent anonymously to a service provider. For this solution to work the trusted third party has to know the identities of both the agent and the service provider. The agents are thus not anonymous with respect to the trusted third party, but only with respect to service provider. For the protocols proposed in this section the (trusted) agent middleware plays the role of the trusted third party. The proposed scheme works as follows:

Protocol 2 *Anonymous Token Generation*

An agent (A), with handle (H_A), is provided with an anonymous access token (AT). This access token is generated by a trusted third party (TTP). It can be used to access a (specific) service provide by service provider (SP).

$$
\begin{array}{lll}
1.\ A \rightarrow TTP & : & H_A, SP \\
2.\ TTP \rightarrow SP & : & \mathsf{Access?} \\
3.\ SP \rightarrow TTP & : & T \\
4.\ TTP \rightarrow A & : & AT
\end{array}
$$

Where the TTP computes sha$(H_A, T, R) = AT$ between step 3 and 4. T is an access token, and R is a random number (that differs for each run of the protocol). The TTP has to keep R secret. It is assumed that all communication occurs over SSL connections [6].

If an agent wants to use a particular service it uses Protocol 2 above to obtain an anonymous access token. The agent requests a new access token each time it invokes the service (by using a new handle in step 1 of the protocol). With the token the agent can use a service anonymously, while the service provider can still check via the trusted third party if the access token is valid. All communication in this and the next protocols occurs over SSL connections [6] which ensures the confidentiality and integrity of all the messages. It also ensures that both the agent A and the service provider SP establish mutual authentication with the trusted third party TTP (but not with each other).

This scheme prevents the generation of false access tokens for both the service provider SP and the agent A. Neither the service provider nor the agent can generate an anonymous access token, as neither party has enough information: both do not know the random number R, which the TTP generates and stores securely. The access token T can, for example, be a password or a one time access token (a random number) generated by the service provider. When an agent accesses a service it presents the access token AT, which the service provider can check for validity using the trusted third party TTP:

Protocol 3 *Anonymous Authentication*

An agent (A) can access a service if it possesses the anonymous access token (AT) to authenticate itself. This token can be generated with Protocol 2 above. The service provider (SP) can check the validity of the access token (AT) with the trusted third party (TTP).

$$\begin{array}{lll} 1.\ A \rightarrow SP & : & H_A, AT \\ 2.\ SP \rightarrow TTP & : & \mathsf{Valid?}, H_A, AT \\ 3.\ TTP \rightarrow SP & : & \mathsf{yes \backslash no} \\ 4.\ SP \rightarrow A & : & \mathsf{grand \backslash deny\ access} \end{array}$$

Note that the anonymous access token is not transferable, as the trusted third party can verify if a token belongs to the agent handle (here H_A) that invokes the service. The service provider needs to forward both the handle and the anonymous access token to the trusted third party. If the trusted third party stores the 'normal' token T, the agent handle H_A and the (generated) access token together with the anonymous access token AT it can verify if an anonymous access token belongs to an individual agent, without revealing the agents identity to the service. Again, the connection between the service provider and the trusted third party is encrypted using SSL. This is not the case for the connection between the agent and the service provider, which would remove the anonymity of agent A. Neither is SSL encryption necessary in this case since the anonymous access token is linked to the agent A's handle H_A and can thus not be redistributed to other agents (eavesdropping and replaying AT does not work).

8.2. Anonymous Payment

Protocol 2 above can be altered to allow anonymous payment for services through the trusted party TTP. The general idea is the same, but the access token AT_M now also depends on the amount transferred:

Protocol 4 *Anonymous Payment*

An agent (A), with handle (H_A), is provided with an anonymous access token (AT) that is used for the anonymous payment of a fee (M) to a service provider (SP).

$$\begin{array}{lll} 1.\ A \rightarrow TTP & : & H_A, SP, M \\ 2.\ TTP \rightarrow SP & : & \mathsf{Access?}, M \\ 3.\ SP \rightarrow TTP & : & T \\ 4.\ TTP \rightarrow A & : & AT_M \end{array}$$

Where the TTP computes sha$(H_A, T, R, M) = AT_M$ between step 3 and 4. T is a 'normal' access token and R is a random number (that differs for each run of the protocol). All communication occurs over SSL.[10]

The above protocol makes anonymous payments in agent systems possible. More elaborate schemes, for example based on WS-Agreements [19], allow agents to negotiate the payment fee for accessing a service. Payment service access is identical to the normal service access explained in Protocol 3.

9. Discussion

This paper introduces a new anonymizing approach for agent systems that guarantees pseudonym based anonymity for individual agents. Such anonymous agents can be used to ensure the privacy of users, e.g. with respect to service providers. The proposed solution has a limited performance overhead. Extensions to the system, covered in this paper, include anonymous authentication and payment; making practical applications more feasible.

Although the approach discussed is theoretically secure, in practice a number of risks remain. If the number of agents on an agent platform is known to all agents[11], then, in theory, it is possible that all agents conspire together against one agent. This breaks link anonymity. A simple solution to this problem is to use a number of 'dummy' agents that belong to the agent platform itself. A dummy agent is basically no more then the registration of a couple of handles in the lookup service. In this case other agents cannot determine if an agent is real or belongs to the platform an thus the attack no longer works.

Another risk, although highly unlikely due to its intrinsic properties, is the use of side channel attacks [14]. Timing attacks, as discussed in [22], form a particular challenging problem. Using a combination of approaches that observe timing behavior together with statistical analysis almost certainly breaks anonymity.

[10] Note again that the agent and service provider only talk to the trusted third party, not to each other. The trusted third party thus ensures the anonymity

[11] Which can happen if each handle uses a corresponding *publicly known* human readable name.

Acknowledgments

This research is conducted as part of the ACCESS project[12] funded by the NWO TOKEN program. The authors thank Stichting NLnet for their support, David de Groot and Benno Overeinder for their comments on earlier drafts of this paper and Reinier Timmer for the realization of the implementation in AgentScape. We also thank the anonymous(!) reviewers for the valuable comments and suggestions to earlier drafts of this paper.

References

[1] F. Bellifemine, A. Poggi, and G. Rimassa. JADE–A FIPA-compliant agent framework. *Proceedings of PAAM*, 99:97–108, 1999.

[2] F. M. T. Brazier, A. Oskamp, J. E. J. Prins, M. H. M. Schellekens, and N. J. E. Wijngaards. Anonymity and software agents: An interdiscplinary challenge. *AI & Law*, 1-2(12):137–157, 2004.

[3] L. Camp, M. Sirbu, and J. Tygar. Token and Notational Money in Electronic Commerce. In *Proceedings of the First USENIX Workshop in Electronic Commerce*, pages 1–12, 1995.

[4] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, 1981.

[5] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, volume 2, 2004.

[6] T. Elgamal and K. E. B. Hickman. Secure socket layer application program apparatus and method, Oct. 20 1998. US Patent 5,825,890.

[7] K. Finkenzeller. *RFID handbook*. Wiley Hoboken, NJ, 2003.

[8] F. D. Garcia and J.-H. Hoepman. Off-line Karma: A Decentralized Currency for Peer-to-peer and Grid Applications. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *3rd Applied Cryptography and Network Security (ACNS 2005)*, volume 3531 of *LNCS*, pages 364–377. Springer-Verlag, 2005.

[9] J. A. Goguen and J. Meseguer. Unwinding and inference control. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1984.

[10] J. Grijpink and J. E. J. Prins. New rules for anonymous electronic transactions? An exploration of the private law implications of digital anonymity. In *Digital Anonymity and the Law, Tensions and Dimensions, Information technology and law series*, volume 2, pages 249–269. T. M. C. Asser Press, 2003.

[11] A. Helsinger, M. Thome, T. Wright, B. Technol, and M. Cambridge. Cougaar: a scalable, distributed multi-agent architecture. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 2004.

[12] C. Kaufman, R. Perlman, and M. Speciner. *Network Security, PRIVATE Communication in a PUBLIC World*. Prentice Hall, 2nd edition, 2002.

[13] L. Korba, R. Song, and G. Yee. Anonymous Communications for Mobile Agents. In *Proceeding of the 4 thInternational Workshop on Mobile Agents for Telecommunication Applications (MATA'02)*, volume 2521 of *LNCS*, pages 171–181, 2002.

[14] B. W. Lampson. A note on the Confinement Problem. *Communications of the ACM*, 16(10):613–615, 1973.

[15] R. Leszczyna and J. Górski. Untraceability of mobile agents. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1233–1234, New York, NY, USA, 2005. ACM Press.

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

[17] Z. Maraikar. Resource and service discovery for mobile agent platforms. Master's thesis, Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, August 2006.

[18] D. G. A. Mobach. *Agent-Based Mediated Service Negotiation*. PhD thesis, Computer Science Department, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, May 2007.

[19] D. G. A. Mobach, B. J. Overeinder, and F. M. T. Brazier. WS-Agreement based resource negotiation framework for mobile agents. *Scalable Computing: Practice and Experience*, 7(1):23–36, 2006.

[20] B. J. Overeinder, M. A. Oey, R. J. Timmer, R. van Schouwen, E. Rozendaal, and F. M. T. Brazier. Design of a secure and decentralized location service for agent platforms. In *Proceedings of the Sixth International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2007)*, May 2007.

[21] B. J. Overeinder, P. D. Verkaik, and F. M. T. Brazier. Web service access management for integration with agent systems. In *Proceedings of the 23rd Annual ACM Symposium on Applied Computing, Mobile Agents and Systems Track*, 2008.

[22] A. Pashalidis and C. J. Mitchell. Limits to Anonymity when Using Credentials. In *Proceedings of the 12th International Workshop on Security Protocols*, LNCS. Springer-Verlag, 2004.

[23] A. Pfitzmann and M. Kohntopp. Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology. In *International Workshop on Design Issues in Anonymity and Unobservability*, LNCS, pages 1–9. Springer, 2001.

[24] V. Roth and M. Jalali-Sohi. Concepts and architecture of a security-centric mobile agent server. In *"Proc. of the Fifth International Symposium on Autonomous Decentralized Systems (ISADS 2001)"*, pages 435–442. IEEE Computer Society, 2001.

[25] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on selected areas in communications*, 21(1), 2003.

[26] T. Sander and A. TaâĂŞShma. On Anonymous Electronic Cash and Crime. In M. Mambo and Y. Zheng, editors, *Proceedings of Information Security, Second International Workshop (ISW'99)*, volume 1729 of *LNCS*, pages 202–206. Springer-Verlag, 1999.

[27] A. Serjantov. *On the anonymity of anonymity systems*. PhD thesis, University of Cambridge, 2004.

[28] R. van Schouwen. Design and implementation of a secure, decentralized location service for agent platforms. Master's thesis, Department of Computer Sciences, Vrije Universiteit Amsterdam, aug 2006.

[12] http://www.iids.org/access

[29] G. van 't Noordende, B. J. Overeinder, R. J. Timmer, F. M. T. Brazier, and A. S. Tanenbaum. A common base for building secure mobile agent middleware. In *Proceedings of the 2nd International Multiconference on Computer Science and Information Technology (IMCSIT)*, volume 2, pages 13–25, Wisła, Poland, Oct. 2007.

[30] M. Warnier and F. M. T. Brazier. Organized anonymous agents. In *the Proceedings of The Third International Symposium on Information Assurance and Security (IAS'07)*. IEEE, August 2007.

[31] M. Warnier, F. M. T. Brazier, and A. Oskamp. Security of distributed digital criminal dossiers. *Journal of Software*, 3(3):21–29, mar 2008.

[32] A. Weimerskirch and G. Thonet. A Distributed Light-Weight Authentication Model for Ad-hoc Networks. In *The 4th International Conference on Information Security and Cryptology (ICISC 2001)*, volume 2288 of *LNCS*, pages 341–354. Springer, 2002.

[33] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet AuctionBot: a configurable auction server for human and software agents. In *Proceedings of the second international conference on Autonomous agents*, pages 301–308. ACM Press New York, NY, USA, 1998.