Mutual Protection of Co-operating Agents

Volker Roth

Fraunhofer Institut für Graphische Datenverarbeitung
Rundeturmstraße 6, 64283 Darmstadt, Germany
`vroth@igd.fhg.de`

Abstract. Security is a fundamental precondition for the acceptance of mobile agent systems. In this paper we discuss protocols to improve agent security by distributing critical data and operations on mutually supporting agents which migrate in disjunct host domains. In order to attack agents, hosts must collude/conspire across domains. Proper selection of itineraries can minimize the risk of such coalitions being formed.

Keywords: mobile agent security, malicious host, distributed applications, itinerary recording, agent cooperation, ecash

1 Introduction

Mobile agents are bundles of program and state, that move within a network to perform tasks on behalf of their owners. The benefits offered by mobile agents unfold in areas where it is advantageous to move the computation process over a network to the source of data instead vice versa; for instance if huge amounts of data must be processed, or the network is slow, expensive, or is not permanently available. Among the manifold uses for mobile agents, electronic commerce applications are noted most frequently. With good reason – electronic commerce provides perfect grounds to illustrate the benefits of mobile agents as well as the threats that keep us from using them in open networks, so far.

One example benefit of a mobile agent is its ability to carry out certain tedious or time–consuming tasks autonomously while its owner is offline. The goal of, for instance, a shopping agent might be finding the best offer for a given product description, thus optimising the benefit to its owner (which is legitimate). However, shops might be tempted to optimise for their own benefit, even if it means optimising at the expense of the agent in unfair ways (which is not legitimate). This includes manipulation of offers previously collected by the agent, as well as abusing agents as mediators of attacks on competitors. Hence, neither party trusts the other, and both share their distrust in the interconnecting network. As a consequence, the security requirements for mobile agent systems are manifold as well as demanding. Quoting from [3]:"*It is difficult to exaggerate the value and importance of security in an itinerant agent environment. While the availability of strong security features would not make itinerant agents immediately appealing, the absence of security would certainly make itinerant agents very unattractive.*"

In general, the problem of *malicious hosts* is considered particularly challenging. The threats imposed by malicious hosts are immanent in the way mobile agent systems are built. Since agents are executed on their host, each instruction in the agent's code is observed by the controlling (virtual) machine which also maintaines the agent's state. This causes a number of security concerns of which the most prominent ones are:

1. The integrity of the agent, in particular the integrity of its mutable part, need to be protected. Agents might be abused as innocent carriers of illegal or offensive materials such as *warez* (pirated software). Hosts may also try to delete, replace, or invalidate commitments to the agent, such as terms negotiated in electronic commerce applications. This would enable the host repudiate said terms later.
2. Maintaining the secrecy of the agent's computations and data is a fundamental requirement for fair negotiations as well as for computations on confidential information such as the preferences (or *profile*) of the agent's owner, or secret keys.
3. Protecting the integrity of the agent's control flow is a precondition for any agent to trust its own decisions. Otherwise, malicious hosts might make agents believe that an offer is acceptable when it is actually not.

Notable advances were made with regard to items 1 and 2. Karjoth et al. [4] introduced the notion of *strong forward integrity* and proposed protocols for protecting the computation results of free–roaming agents. Their work is an extension of *partial result authentication codes* introduced by Yee [15]. Roth and Jalali proposed an agent structure that supports access control and authentication of mobile agents [6]. Agent authentication and state appraisal is covered by Berkovits et al. [1]. Sander and Tschudin [7] introduced the notion of *mobile cryptography* and devised approaches for non–interactive computing with encrypted functions using *homomorphic encrytion schemes*. Tschudin also proposes an approach for securing termination signals for mobile services [11] which incorporates ideas from Riordan and Schneier [5]. Regarding item 3, Vigna [12] proposed *cryptographic traces* to create verifiable execution traces of agents. Vigna also discusses a number of drawbacks with his approach such as limitation to single–threaded agents, extensive computational and memory overhead, as well as limitation to *a posteriori* detection. The notion of *trust*, its role in mobile code systems, as well as trust management is discussed for instance by Swarup et al. [10] and Blaze et al. [2].

The majority of risks stem from the fact that agents are deployed in open untrusted networks. However, even the hosts connected through this network do not know and do not necessarily trust each other. In the remainder of this paper we intend to exploit exactly this fact for improving the security of mobile agents against attacks by malicious hosts. The general idea is based on the co–operation of multiple agents with the goal of mutual protection (of the application built on them). Thus, co–operation of agents is used to establish a distributed “virtual shelter” inside the open network, in which agents can hide and to which they may retreat while assuring the security of their counterparts.

In Section 2 we motivate and formalise this idea, and give definitions and notations used in our paper. In Section 3 and Section 4 we devise two protocols, based on co–operating agents, suitable to protect different aspects of the general purchase agent against malicious hosts. One protocol is directed at recording the actual itinarary taken by a free–roaming agent; the second protocol addresses the delegation of electronic payments to mobile agents. Conclusions will be drawn in Section 5.

2 Co–operating Agents

With regard to mobile agent security, the most conservative approach is to assume that each host on a mobile agent’s itinerary is hostile and willing to collaborate with other malicious hosts visited by the agent on its route. This assumption is as realistic as the assumption that hosts can be generally trusted, though. A more reasonable assumption is probably the following:

> Given a particular mobile agent, at any point in time, a certain percentage of hosts might be malicious.
> Not all malicious hosts are willing to collaborate with other hosts in attacking a mobile agent.

The percentage of malicious hosts likely depends on the gain which can be expected from successfully attacking mobile agents weighted against the costs of mounting the attack as well as the risk of detection and the consequences of being detected. Collaborations of multiple hosts on an agent's itinerary yields more power but it also requires close coordination and increases the danger of leaks which might lead to disclosure of the collaboration. Whether an agent will be attacked by a single malicious host or a collaboration of hosts on its itinerary depends on a sea of unpredictable parameters that are unique for each instantiation of an agent.

Still, we would like to model partitions of hosts based on their willingness to collaborate with regard to attacking a fixed agent. For this reason, let $\mathcal{H}$ be the set of hosts interconnected by a network. For a given instantiation of an agent let $\mathcal{R}$ be a relation defined as $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{H}$ with the interpretation $(\mathrm{h}_i, \mathrm{h}_j) \in \mathcal{R} \Leftrightarrow$ *h_i and h_j collaborate in attacking the agent*. Let $\mathrm{H}_a, \mathrm{H}_b$ be non–empty subsets of $\mathcal{H}$ with $(\mathrm{H}_a \times \mathrm{H}_b) \cap \mathcal{R} = \emptyset$. These two sets are denoted *non–colluding*. A special host is the first host (the origin) of an agent, since this needs to be a *trusted* host.

Two *co–operating agents* are defined to be agents *a* and *b* such that the itinerary of *a* includes only hosts in H_a and *b*'s itinerary only includes hosts in H_b. Let h_a and h_b be the computing environments currently executing agents *a* and *b* respectively. Occasionally, we will say that h_a is "the host of" agent *a*, or "h_a is agent *a*'s host".

Although agent *a* might be attacked by host h_a, by definition this host may not attack the co–operating agent *b* without breaking into host h_b. This can be exploited to design protocols for securing agents against attacks by single malicious hosts as well as hosts that collaborate with other hosts as long as the collaboration does not span the itineraries of both agents.

A number of strategies can be used as starting points for developing said protocols:

Secret sharing: Authorisation data is secretly shared by both agents. A single share conveys no information about the shared secret. Therefore, a malicious host must get the remaining share by asking a host on the co–operating agent's itinerary to steal it, or it must break into that host.

Remote authorisation: The decision whether the share of the authorisation data is passed to the host executing the agent is taken by the co–operating agent. The agent itself prefilters the data upon which the decision is based, and transfers it to its co–operating agent. Therefore, a malicious host must ask the remote host to manipulate the decision, or it must break into that host.

Remote storage of commitments: An agent transfers commitments of the host on which it runs, such as commercial offers, to its co–operating agent (probably in the course of remote authorisation), which verifies and stores it for future reference and non–repudiation. In order to undo or invalidate its commitment, a malicious hosts needs to break into a host on the co–operating agent's remaining itinerary.

In the remaining sections we illustrate the idea of mutual protection of co–operating agents by giving example protocols. For those protocols to work we need to make two additional assumptions:

- Hosts transport agents through authenticated channels.
- Hosts provide an authenticated communication channel to the two cooperating agents.
- The authenticated identity of the remote host, the authenticated identity of the host the agent came from as well as the local host's identity are provided to the hosted agent.

Yee already pointed out that "if an agent is running on an honest server, both these answers (for the peer identity and the local host's identity) will be correct. . ." [15]. We assume that a host is honest unless it may successfully attack an agent on its own, or with the help of other hosts on this agent's itinerary. In other words, we assume that hosts do not randomly introduce lies.

3 Tracing Loose Routes

A simple yet effective attack on a mobile agent is to not let the agent migrate to the servers of competitors. This particularly affects mobile agents with *loose itineraries* in comparison to agents whose itineraries are defined a–priori, because deviations from a fixed itinerary are easier to spot and prove.

We would like to record the actual loose route taken by a free–roaming agent without any possibility of manipulation by the hosts on its route. Let *a* and *b* be two co–operating agents, and let H_a,$\mathrm{H}_b \subseteq \mathcal{H}$ be two non–colluding sets of hosts. Both agents shall return to their origin upon completition of their tasks. Each agent *b* records and verifies the route of its co–operating agent *a* as described below.

Definition: Let $\mathrm{h}_i \in \mathrm{H}_a$ be the i^{th} host being currently visited by agent *a* and let $\mathrm{id}(\mathrm{h}_i)$ be the identity of host h_i. Let prev_i be agent *a*'s idea of the identity $\mathrm{id}(\mathrm{h}_{i-1})$ of its previous hop. i shall denote the identity of the next hop agent *a* wants to take while being on host h_i. The agents start at host h_0, hence h_n=h_0 for a route with n hops.

Initialization: Let h_0 be the origin of agents *a* and *b*. h_0 has to be a trusted host with respect to *a* and *b*. For agents *a* and *b*, 0 is set to the first hop of their respective itineraries. Both agents are subsequently sent to their first hop.

Step $i, i \in \{1, \ldots, n\}$**:** Agent *a* sends a message containing the next hop i and the previous hop prev_i to agent *b*. The authenticated channel enables Agent *b* to learn id(h_i). Agent *b* verifies that id(h_i)=i-1$\wedge$$\text{prev}_i$=id($h_{i-1}$) and appends i to the stored route.

Security of the protocol: It is straightforward to see that if host h_i forwards agent *a* to a host h_{i+1} with id(h_{i+1})$\neq$i then host h_{i+1} must either successfully masquerade as the host with id i or it has to deny communication between the co–operating agents. On the other hand, if host h_{i+1} permits the communication and properly authenticates itself (in other words, h_{i+1} is honest regarding the protocol) then agent *b* discovers that host h_i sent agent *a* to the wrong destination.

If id(h_{i+1})$\neq$i then host h_{i+1} cannot put agent *a* back on its route by sending *a* to the host with identity i because agent *b* recorded id(h_i) as prev_{i+1}. As a matter of fact, h_{i+1} must either be honest (identifying h_i as a cheater in the process) or h_{i+1} has to collaborate with h_i in putting the agent back on its expected route (more precisely, h_i has to put back the agent on its route itself, or it must disclose its authentication keys to h_{i+1}). The last case reduces to either simply sharing a copy of the agent with h_{i+1}, or merging hosts h_i and h_{i+1} into a single host under the identity of h_i – hardly something which can be prevented or detected at all.

A malicious host h_{i+1} might incriminate a *honest* host h_i by claiming to have received agent *a* from some other host h', hence implicating that h_i sent *a* to h' instead of the host with identity i=id(h_{i+1}). The protocol is not able to decide which one of the two hosts is the culprit. However, if h_{i+1} really received agent *a* from h' then h_{i+1} should be able to produce a copy of *a* which is signed by h' given some additional agent protection mechanisms are implemented (see [6]).

If agent *a* is killed, one of two hosts might be responsible and the protocol cannot decide which one. In addition to that, some host h_{i+1} might take two agents a_1 and a_2 both being received by the same host h_i and switch the recording of the route of a_1 to agent b_2 and vice versa. Therefore, co–operating agents should also exchange and verify (unique) identity information that is bound to the agent's static part by their owner's signature (see [6]). In that case, attempts to send fake ids are detected on the first honest host. The protocol must be enhanced accordingly.

4 Electronic Cash Payments

Current descriptions of mobile purchase agents primarily address collecting and filtering of offers. In this section, we would like to consider delegating payment authorisation to mobile agents as well. The risks in doing so are obvious. Digital representations of money may be copied, payment decision altered, and commitments erased after payment, by malicious hosts. However, modelling a complete purchase cycle with mobile agents, including payment authorisation, is an attractive thought. Cash–like systems are particularly well–suited because in contrast to for instance credit card based systems the possible loss is limited to the value of the digital coin.

Co–operating agents provide an approach to tackle security problems involved in payment authorisation. Protocols such as Chaum's digital electronic cash protocol (as described for instance in [8]) can be adapted to work with co–operating agents as described in the protocol given below. Chaum's protocol provides detection of double spending with cheater identification.

Let *a* and *b* be two co–operating agents, and let H_a,$\mathrm{H}_b \subseteq \mathcal{H}$ be two non–colluding sets of hosts.

Initialisation The owner of the agents prepares a money order m as described in [8]. The identity strings are prepared by randomly choosing keys k_i^l, k_i^r, $i = 1, \ldots, n$ and computing

$$\{I_i = (E_{k_i^l}(I_i^l, H(I_i^l)), E_{k_i^r}(I_i^r, H(I_i^r))) \mid i = 1 \ldots n\}$$

where E is a suitable encryption function and H is a strong cryptographic one–way hash function. The money order as well as the keys are secretly shared between agents *a* and *b*. Both agents are sent on their respective itineraries.

Step 1: Agent *a* locates a product or service on host h_a (the one on which the agent runs) that it wishes to purchase, and requests a signed offer describing the subject and terms of the purchase.

Step 2: Host h_a provides the requested offer as well as the *selector string* defined in the ecash protocol.

Step 3: Agent *a* forwards the offer and the selector string to its co–operating agent *b*.

Step 3: Agent *b* validates the offer's signature. In order to do so, it verifies that the identity of the offer's signer matches the one determined from the authenticated channel, and that the offer's signature is valid. It goes on to verify the terms of the offer. If the verification fails then *a* is notified of the result and the protocol is aborted.

Step 4: Agent *b* stores the offer (if h_a later on repudiates having made that offer then its signature may be used as a proof), transfers its share of m to *a*, and opens the selected halves of the identity strings by sending the appropriate key shares as well.

Step 5: Agent *a* passes the data on to h_a.

Step 6: Host h_a reconstructs the money order m, verifies the bank's signature and makes sure that agent *b* properly opened the selected halves of the identity strings. The ecash protocol goes on as described in [8]. If the verification fails then the agents are notified, and the protocol is aborted.

Step 7: Host h_a delivers the purchased goods.

Security of the protocol: Clearly, host h_a cannot manipulate the payment *decision* in step 3 without collaborating with h_b. Neither of the hosts involved can steal the money order without the help of the other (or breaking into the remote host). Neither can host h_a alter, invalidate or delete its offer after step 4. Of course, either the host or the co–operating agents may terminate the protocol at will. However, this is a general problem of electronic payment protocols.

However, any two hosts may profit from a joint attack on the cooperating agents by sharing the additional wealth gained from defrauding the agents compared to the profit gained from honest behavior. Therefore the domains of both agents' itineraries must be chosen with great care in order to assure $(\mathrm{H}_a \times \mathrm{H}_b) \cap \mathcal{R} = \emptyset$ holds with reasonable confidence. Security can be improved by additional measures [6] which ensure that payment can only be made by *b* while being on particular hosts.

5 Discussion and Conclusions

In this paper, we introduced the concept of co–operating agents as well as the basic principles on which protocols for mutual protection of such agents rest. These principles were illustrated by giving two example protocols, one for recording the actual itinerary taken by free–roaming agents, and one for protecting ecash payments with co–operating agents.

The ability to determine the actual route taken by an agent in principle allows further improvements of the security of agents against unauthorised and unnoticed manipulation. Hosts may commit to changes in the structure of an executed agent by digitally signing incremental logs of the agent's state upon its migration. According to the recorded route, the chain of signatures can be validated. Hence, in addition to *strong forward integrity* [4] co–operating agents may provide resistance against truncation and erasure with non–repudiation. Hosts which do not want to forward agents to the servers of certain competitors might still do so. However, the itinerary of the cooperating agent will always show this fact.

However, the ability of co–operating agents to mutually protect themselves relies crucially on the assumption that no pair of hosts chosen from both agent's itineraries collude. This precondition must be ensured with reasonable confidence. The most simple solution is to have one agent migrate to a trusted host while the other agent is free to roam the network. In this degenerate case, co–operating agents reduce to the trusted third party concept with the important difference, that *even the trusted third party need not be fully trusted*. Therefore, given a sufficient number of agent servers in the Internet, even a randomised strategy might be sufficient: One itinerary is randomly chosen and fixed while the second agent is free to choose its next hop at run time. The randomised strategy is based on the assumption that the sets of collaborating hosts are small compared to the number of available hosts. The overall security might be improved by complementing the approach taken in this paper with research in agent routing policies such as the one indicated by Swarup [9].

In particular, the ecash adaption protocol illustrates how distributed applications based on mobile agents may be securely deployed while preserving in particular the autonomy of mobile agents as well as their ability to perform useful tasks off–line, and maintain confidence in the delegation of authority as well.

In summary, protocols based on co–operating agents have merit since they are less susceptible to attacks by coalitions of hosts than single agents. The underlying principle is a generalization of the trusted third party concept which is less restrictive and easier to meet.

We plan to implement and evaluate the protocols described in this paper within the Java–based SeMoA platform (Secure Mobile Agents). SeMoA denotes an ongoing development project undertaken by the *Fraunhofer Institut für Graphische Datenverarbeitung*, with the goal of exploring and developing techniques and architectures allowing secure deployment of mobile agent technology in the area of multimedia and e–commerce applications.

6 Acknowledgements

I would like to thank the reviewers for their extensive comments on the initial manuscript, which were most valuable for improving the quality of this paper.

References

1. BERKOVITS, S., GUTTMAN, J. D., AND SWARUP, V. Authentication for mobile agents. In Vigna [13], pp. 114–136.
2. BLAZE, M., FEIGENBAUM, J., IOANNIDIS, J., AND KEROMYTIS, A. he role of trust management in distributed systems security. In *Secure Internet Programming* [14].
3. CHESS, D., GROSOF, B., HARRISON, C., LEVINE, D., PARRIS, C., AND TSUDIK, G. Itinerant agents for mobile computing. *IEEE Personal Communications* (October 1995), 34–49.
4. KARJOTH, G., ASOKAN, N., AND GÜLCÜ, C. Protecting the computation results of free–roaming agents. In *Proceedings of the Second International Workshop on Mobile Agents (MA '98)*, K. Rothermel and F. Hohl, Eds., vol. 1477 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin Heidelberg, September 1998, pp. 195–207.
5. RIORDAN, J., AND SCHNEIER, B. Environmental key generation towards clueless agents. In Vigna [13], pp. 15–24.
6. ROTH, V., AND JALALI, M. Access control and key management for mobile agents. *Computers & Graphics 22*, 3 (1998). Special issue *Data Security in Image Communication and Networks*.
7. SANDER, T., AND TSCHUDIN, C. F. Protecting mobile agents against malicious hosts. In Vigna [13], pp. 44–60.
8. SCHNEIER, B. *Applied Cryptography*, 1 ed. John Wiley & Sons, Inc., 1994, section 6.7, pp. 120–122. Digital Cash Protocol #4.
9. SWARUP, V. Trust Appraisal and Secure Routing of Mobile Agents. DARPA Workshop on Foundations for Secure Mobile Code, Monterey, CA, USA, March 1997. Position Paper.
10. SWARUP, V., AND FABREGA, J. T. Understanding trust. In *Secure Internet Programming* [14].
11. TSCHUDIN, C. Apoptosis — the programmed death of distributed services. In *Secure Internet Programming* [14].
12. VIGNA, G. Cryptographic traces for mobile agents. In *Mobile Agents and Security* [13], pp. 137–153.
13. VIGNA, G., Ed. *Mobile Agents and Security*, vol. 1419 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin Heidelberg, 1998.
14. VITEK, J., AND JENSEN, C. *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, vol. 1603 of *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1999.
15. YEE, B. S. A sanctuary for mobile agents. In *Secure Internet Programming* [14], pp. 261–273.