PRIVATE COMPUTING WITH BEEHIVE ORGANIZED AGENTS

Bartek Gedrojc, Jan C.A. van der Lubbe
Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands

Martin van Hensbergen
Fox-IT Forensic IT Experts, Delft, The Netherlands

Keywords: Mobile Agents, Secret Sharing, Threshold Signature, Hash Chaining, Untrustworthy Hosts, Private Computing.

Abstract: Consider the problem of using mobile agents within an e-commerce setting where the goal is to purchase a desired item for a user. The problem is that the mobile agents visit a collection of hosts which are untrustworthy and therefore could tamper with the correct execution of the agents. Our approach to the e-commerce problem prevents the hosts to retrieve the price the user is willing to pay for a desired item, it prevents the hosts to retrieve the offers given by other hosts (confidentiality) and it ensures the integrity of the agents' code, the query and itinerary. The key to our approach is the use of multiple agents for our goal; to purchase a desired item for a user. Analogous to a beehive, the user creates Drone agents that can collect data but which do not have the capability to process this data. Also, one Queen agent is deployed which uses the outputs of the Drone agents and makes a decision on that data. Simplified, we let the Drone agents do the work, while the Queen computes the result.

1 INTRODUCTION

Searching, comparing and buying e.g. airline tickets on the Internet is still a major problem, especially if this task has to be done on a mobile device. It takes time to query various websites and to compare the vast amount of offers of various e.g. airline companies. Also, due to the limited computational powers of mobile devices and the limited bandwith of mobile networks, it would be desireable to shift the computation cost from the mobile device to the server side and to limit the communications between the mobile device and the server.

Mobile software agents are based on mobile code which performs a task for the owner of the agents. Mobile code is code which can be transfered from one computer (sender) to another (recipient) over a network and which is executed on the recipient's side. Using techniques from artificial intelligence, mobile code can be programmed to execute tasks autonomously in behalf of his owner. A simple but clear definition of agents has been given by Ted Selker of the IBM Almaden Research Centre, "An agent is a software thing that knows how to do things that you could probably do yourself if you had the time".

E-commerce scenarios, like buying airline tickets on the Internet using a mobile device, could be improved by using mobile software agents. Mobile agents can be deployed on mobile devices with a task given by its owner, the mobile device user, and sent to various servers on the Internet. Because the mobile agents are autonomous they require no interaction with their owner, consequently, the owner can disconnect the mobile device from the network and go online again to obtain the information collected by his agent.

The mobile software agent collects information from various sources on the Internet and compares this with the preferences of his owner. This preference can contain, in the case of buying online airline tickets, the maximum price the owner is willing to pay, a list of prefered airline companies, destination, *et cetera*. Which is private and sensitive information. After processing all collected data and matching it with the owners preference, the agents returns to his owner where he presents the result of his comparison.

Execution of code on the recipient's computer poseses a number of security issues. From the recipient point of view, the execution of untrustworthy code can harm the recipients computer e.g. mobile code can be a virus, a worm or a Trojan. A number

Gedrojc B., C.A. van der Lubbe J. and van Hensbergen M. (2007).
PRIVATE COMPUTING WITH BEEHIVE ORGANIZED AGENTS.
In *Proceedings of the Second International Conference on Security and Cryptography*, pages 289-294
DOI: 10.5220/0002127402890294

of techniques have been suggested, and used, to alleviate some of these issues, like sandboxing (Lai et al., 1999) and the use of certificates (Tan and Moreau, 2002). From the sender of the agents point of view, the execution of the mobile code on the recipients computer can be manipulated or spied on because the recipient has full control over the mobile code. The latter issue is also called Private computing, where the goal is to let a recipient compare private data of the sender without learning any relevant information about this private data of the sender which could influence the outcome of the comparison.

Literature on mobile software agents in e-commerce settings provides a variety of solutions e.g. sequential execution of a single agent or parallel distribution of multiple agents. E-commerce scenarios with mobile devices should not be based on multiple agents due to the computational and communicational limitations of the mobile devices i.e. the more agents that are deployed, the more agents that need to be managed. Therefore, it is desirable to deploy a single agent per owner which is able to query various servers, compare this with the preferences of his owner and return with his result to that owner. The main problem with this latter approach is the execution of mobile code on untrustworthy computers (Sander and Tschudin, 1998a; Sander and Tschudin, 1998b; Cachin et al., 2000; Algesheimer et al., 2001).

When a single agent is deployed on the Internet to persue a task given by its owner, then this agent is vulnerable and likely to be attacked by malicious hosts e.g. with the replay attack, where an agent is used as an oracle to extract private information from it by executing it numerous times with different inputs and observing the outputs.

In this paper, a number of issues are addressed in an e-commerce setting where the mobile users has to execute his mobile code on untrustworthy computers (hosts). The focus of this paper lies on minimizing the risk of losing private data of the owner, making sure replay attacks are not worthwile and protecting the pre-defined path the agents is willing to travel (itinerary). In section 2 the problem is described together with the assumptions and requirements of the solution. Section 3 explains the approach that was chosen to solve the problem together with a brief description of the main techniques that where used. While section 4 explains the complete protocol, section **??** makes a security analysis of the protocol. Finally this paper concludes with a conclusion in section 6.

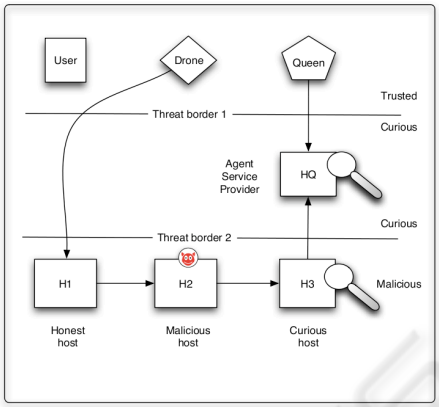

Figure 1: Agent model.

2 PROBLEM STATEMENT AND ASSUMPTIONS

This paper addresses the problem of using mobile agents within an e-commerce setting where the goal is to purchase a desired item for a user. The issue is that the mobile agents visit a collection of environments (merchants) which are untrustworthy and therefore could tamper with the correct execution of the agents.

Figure 1 shows a schematic overview of the agent model which is used in this paper. The model is divided into three parts and shows the various environments an agent can operate, trusted, curious and malicious. The user of an agent is located at an trusted environment where he is able to create mobile agents securely and where he is able to hide his private information.

The curious environment will execute agents correctly and therefore will not tamper with the agents, but it is interested in the agents' code and memory in order to find secrets. This environment can be compared with an Internet Service Provider (ISP) which is able to read all Internet traffic of his clients but it is not in his interest to misuse this information (reputation or for a fee). Therefore, this curious environment can also be named Agent Service Provider (ASP).

Malicious environments cannot only read the agents' code and memory, but are also able to alter the agents code, clone it, or only partially execute it. Agents that are executed by these environments should not leak any private information. It is impossible to prevent tampering of the agents by these environment but it should be possible to detect missuse. It must be noted that merchants which are visited by the agents are considered to be malicious.

By assuming the use of mobile software agents the scenario starts with the creation of the agent(s) by the user. The user provides the agents the necessary information which enables them to visit the prefered merchants, be ability to collect information from those merchients and make a decision on the collected information. After executing the agents they return to the user with the computed result.

The solution to our agent model should fulfill the following requirements. The collected information from the merchants should be hidden for other merchants. For example, in the case of collecting offers from airline companies, these companies should not be able to offer a price for a ticket based on already collected offers from other companies. The goal is therefore to have confidentiality of the retrieved offers.

Agents should also not reveal the full list of merchients they are planning to visit (itinerary), consequently the itinerary should be tamper-proof.

Also, the amount of network traffic the agents generate should be minimized. Especially the interaction with the users due to the use of a mobile device which has minimum computational powers.

The final requirement is related to the tampering of the malicious merchants. As stated, one cannot prevent tampering of the agent by the merchant, but it should be the case that any merchant should be able to detect malicious activity.

The assumptions of this agent model are that there is a Public Key Infrastructure (PKI) available for the distribution of the public keys of all participants. This agent model does not consider a denial of service attack and it also assumes that not all merchants collude with each another.

3 APPROACH

An optimal solution to the problem described in section 2, in the sense of minimal network traffic, would be to use a single mobile agent which hops from one merchant to another without interacting with his owner. The problem with this solution is that the agent is carrying all private information of the user and therefore it is likely to be reveal private information if it is tampered with e.g. using a replay attack. By deploying multiple agents and sending to every merchant a different agent, tampering can be deteceted and no private information will leak. The downside to this solution is the increase in network traffic. Considering the above solutions, it would be desirable to develop a trade-off between these solutions. How many agents are there needed to meet the requirements of section 2?

The approach of this paper is based on beehive organization where there is one queen who makes the decisions and multiple drones that do the actual work. These two types of agents can shortly be characterized as:

- **Drone** An agent that can only collect data but does not have any decision making logic
- **Queen** An agent that takes the output of the drone(s) and makes a decision based on that data

By splitting up the tasks of the agents and letting a drone collect information from various merchants, the private information of the owner cannot be revealed, because the drone is not able to process collected data e.g. match with previously collected offers and deciding which of the offers is the maximum. This process in only available with the queen. The gathering of information is a rather neutral activity however and sending only a drone to collect the information will keep the decision logic well away from the malicious environments.

The queen, which does carry the decision logic, will not pass any malicious hosts. Rather, it is executed on a fixed host HQ and is quite immobile compared to the drone. The queen will thus travel only to a curious host where it waits for the drone's arrival. Once the drone and queen are together they will make a decision on the best offer. See figure 1.

There is also a third agent involved, called the helper-agent, which will be involved when the drone cannot be sent to its next destination. This helper-agent will also reside on a curious host (which can either be the same or a different host as where the queen resides) but will not move from there. The helper-agent is only needed when a host H_i in the drone's itinerary does not function, otherwise the helper-agent will just shut down after a pre-determined time.

This paper is based on (Singelée and Preneel, 2004) which present a mobile agents scenario that can securely collect information, protect the collected information against untrusted hosts, and it can digitally sign transactions in an untrusted environment. In order to protect different aspects of the agent model a variety of cryptographic techniques and concepts is needed. None of the protocols are described at algorithm level, which means that a suitable algorithm can be chosen whenever a hash-function or encryption algorithm is used. The encryption of a value v with a symmetric encryption algorithm using key K is denoted as $E_K(v)$, whereas the encryption of the value using the public key of a particular host H_i is denoted as $E_{H_i}(v)$ (or $E_{P_{H_i}}(v)$). A digital signature over a value v with host H_i's private key will be

denoted by $Sig_{H_i}(v)$.

Secret Sharing Scheme. For the storage of the secret parameters in the queen a public key encryption algorithm is used. Since this storage will need to be decrypted once the drone has reached the queen, a secret sharing algorithm is used to split the decryption key in two parts; one which is given with the drone and one which is given to the queen e.g. (Shamir, 1979; Blakely, 1979). Without the presence of the drone, the host which executes the queen cannot decrypt the secret data stored in the queen.

Threshold Signature Scheme. Since the agents need to sign the best bid at the end of the journey, a signing key must be stored by the agents. For obvious reasons this key cannot be placed in the drone. Placing the signing key in the encrypted storage of the queen, while better suited, has the drawback that the signing key will be reconstructed in its entirety on a curious host.

To counter this a threshold signature scheme is chosen e.g (Desmedt and Frankel, 1994). With a threshold signature scheme, the signing key is split into two parts. Each party in the signature process can use its share to create a partial signature on a message. These partial signatures can then later be combined to create one complete signature. In our model we let the merchants partially sign their bids all with the same share and give that to the drone. Once the queen finds the best offer, it creates a partial signature of that same offer with its share and combines the two partial signature to one complete signature. This way, the complete signing key is never reconstructed in one place.

Hash Chaining. Is a method of providing integrity of data when this data is being augmented by multiple parties. The within this section described hash chaining technique is taken from (Singelée and Preneel, 2004). The protocol described here will need some modifications for our agent model, but the principles will remain the same. Each host H_i that the drone visits will add its offer o_i to an encrypted storage in the agent. By using hash chaining, host H_i not only adds its offer to the storage, but also makes a commitment that he adds it to the proper storage, by including a hash of the previous storage state. Each host H_i will follow the following protocol for storing its offer o_i to the storage.

- Encapsulated offer:

$$O_i = Sig_{H_i}(E_{P_Q}(o_i, r_i), h_i), 0 \leq i \leq n \qquad (1)$$

- Chaining relation:

$$h_0 = h(o_0, H_1) \qquad (2)$$
$$h_i = h(O_{i-1}, H_{i+1}), 1 \leq i \leq n \qquad (3)$$

Here, o_0 is initial information (e.g. identitiy of the agent), o_i the offer of host H_i, O_i the encapsulated offer from host H_i, r_i a random number generatated by H_i, $E_{P_Q}(v)$ is the encryption of value v with the public key of the queen and $h(.)$ a cryptographic hash function. The encrypted storage will consist of the chain $O_0, O_1, \ldots, O_n$.

The essence of the protocol is that a host H_i signs both its offer and a hash value taken over the last encapsulated offer and the next destination of the agent. If a malicious host H_i would like to delete, for example, an offer $O_k (k < m)$, from the storage, then this will be detected during verification of the hash chain because the committed value h_{k+1} will not verify.

4 PROTOCOL

This section will describe the complete protocol. It must be noted that the complete process of creating agents, sending them out and gaining the results is called a mission.

4.1 Instantiation of the Agents

For each mission, three agents will be constructed. Let the itinerary of the drone be given by the hosts $H_1, H_2, H_3, \ldots, H_n, HQ$ where $H_i (1 \leq i \leq n)$ are merchants and HQ is the location of the host which executes the queen. Let H_P denote the location of the host which hosts the helper-agent. Fix a security parameter $m \leq 1$ which denotes the maximum amount of succeeding hosts that may fail so that the drone can still continue its journey. The case $m = 0$ does not require a helper agent so we ignore this case. Denote the parameters which describe the item(s) that the agent seeks with Q and the secret decision parameters or logic by F.

Mission instantiation.

- Generate a public key P_Q, which will be used to encrypt the Queen's data, and split the corresponding private key in two parts S_{Q_1} and S_{Q_2} using a secret sharing scheme.
- Generate a secret signing key S, which will be used to sign the best offer by the merchants, and split the signing key in two parts s_1 and s_2 using a threshold signature scheme.

Drone instantiation. In this protocol, h is a strong cryptographic hash function which generates d bit hashes (with d sufficiently large) and $r|_k$ means the first k bits of the bit-string r.

1. Choose a k such that $2^k = |\{H_i\}|$ and $k < d/2$.
2. For each host H_i to visit, generate a symmetric encryption key K_i and a (small) random nonce r_i $(1 \leq i \leq n)$, such that $r_i|_k \neq r_j|_k$ for $j < i$ and a random nonce n_i.
3. Calculate iteratively the itinerary as

$$I_n = HQ \tag{4}$$

$$I_i = [E_{P_{H_i}}(K_i, r_i), E_{K_i}(H_{i+1}, S_{Q_2}, I_{i+1})] \quad \text{with } i = n-1, \ldots, n-m \tag{5}$$

$$I_i = [E_{P_{H_i}}(K_i, r_i), E_{K_i}(H_{i+1}, I_{i+1})] \quad \text{with } i = n-m-1, \ldots, 1 \tag{6}$$

Store I_1, P_Q, S_{Q_2} and the location of the helper-agent in the drone and send it to H_1.

Queen instantiation protocol.

- Calculate the encrypted storage: $E_{P_Q}(H_1, n_1, H_2, n_2, \ldots, H_n, n_n, F, s_2)$
- Store the encrypted storage together with S_{Q_1} in the queen

Helper agent instantiation protocol. Calculate and store the following lookup-table:

Table 1: Helper agent lookup-table.

look-up key	value	verification
$r_i\|_k$	$E_{K_i}(E_{P_{H_i}}(K_{i+t}, n_{i+t}))$	$h_{i,t} = h(r_i \| H_{i+t})$

with $t = 1, \ldots, m$ and $i = 1, \ldots, n-1$.

4.2 Execution of the Agents

When the agents are initialized, the queen and helper-agent are sent to their respective agent service providers. The helper-agent awaits instantiations of the helper protocol and is deactivated after a pre-determined period, so helper agents will not leave the ASP once they are there. The queen awaits the coming of the drone or gets deactivated after a pre-determined period whichever comes first. The drone is sent to H_1 and continues its journey from there.

Offer collection protocol - drone at merchant. When a drone is executed on a host H_i, the following protocol is executed

1. Host H_i uses its private key to decrypt the first element of I_i, as given in (5) and (6), to obtain K_i, which can be used to decrypt the second element of I_i to obtain H_{i+1} and the rest of the (encrypted) itinerary.
2. Host H_i creates an offer o_i and uses s_2 to partially sign its offer, denoted by $c_{i,2}$
3. Host H_i calculates using PQ the encapsulated offer using the following hash chaining relations:

   If help-protool was not needed:

$$O_i = Sig_{H_i}(E_{P_Q}(o_i, \hat{r}_i, c_{i,2}), h_i), 0 \leq i \leq n \tag{7}$$

$$h_i = h(O_{i-1}, H_i) \tag{8}$$

   If help-protocol was needed to skip t hosts:

$$O_i = Sig_{H_i}(E_{P_Q}(o_i, \hat{r}_i, c_{i,2}), n_{i+1}, n_{i+2}, \ldots \ldots, n_{i+t}), h_i) \tag{9}$$

$$h_i = h(O_{i-1}, H_{i+t+1}) \tag{10}$$

4. The hash chain, I_{i+1} and agent are sent to the next destination.

In case step 4 fails, the help of the helper-agent is called and the following help-protocol is used:

Help protocol - drone at merchant communicating with helper-agent.

1. H_i sends a help request to the helper agent consisting of the tuple $r_i, \hat{H}_1, \hat{H}_2, \ldots, \hat{H}_s$ where r_i is the value included in I_i and $\hat{H}_1, \hat{H}_2, \ldots, \hat{H}_s$ are the hosts it wishes to skip
2. P checks if $1 \leq s \leq m$ and if $r_i|_k$ is part of its lookup table. If not, it aborts.
3. For each $k = 1, \ldots, s$ execute step 4
4. P verifies if $h(r_i || \hat{H}_k) = h_{i,k}$
5. P returns $E_{K_i}(E_{P_{H_i}}(K_{i+s}, n_{i+s}))$ to H_i

Decision protocol - drone and queen at HQ. Once the drone reaches the queen, they will together decide on the best offer. For this to work, the following steps must be undertaken:

1. S_{Q_1} from the drone and S_{Q_2} from the queen are combined to obtain S_Q
2. The queens encrypted storage is decrypted using S_Q, thereby obtaining the decision logic, the other half of the signing key s_1 and the itinerary that the drone should have followed
3. For each encapsulated offer that the drone has collected execute steps 4 - 6
4. Decrypt O_i to get actual offer o_i

5. Verify if signed by a host which was included in itinerary, if verification fails: abort
6. Verify if each signer is only represented once
7. For each host $H_i \in I_D$ that did not do an offer, verify if n_i is present in encrypted storage. If verification fails: abort.
8. Input the offers $o_1, o_2, \ldots, o_n$ into F to determine b offer o_b
9. Use s_2 to partially sign o_b and combine with $c_{b,1}$ to make the signature complete.

5 SECURITY ANALYSIS

With the construction of the itinerary a host H_i has sufficient information to be able to determine the next host in the agent's itinerary but cannot get any hosts beyond that without execution of the help-protocol.

Property 1 (Correctness). *Suppose all the hosts H_i are available and execute offer collection protocol correctly. Then*

i) each host H_i is able to obtain the next destination H_{i+1}

ii) each host H_i cannot get destinations H_j with $j > i+1$

Proof. i) The case $i = n$ are trivial. Take $0 < i < n$: at host H_i the itinerary consists of the tuple

$$[E_{P_{H_i}}(K_i, r_i), E_{K_i}(H_{i+1}, I_{i+1})]$$

Since host H_i has access to its private key, it can decrypt the first part of the tuple to obtain K_i. With this K_i it can decrypt the second part of the tuple to get H_{i+1} and I_{i+1}. Sending the latter to host H_{i+1}, the process can be repeated by H_{i+1} to obtain the location of H_{i+2}, *et cetera*.

ii) In order for a host H_i to get knowledge of H_j with $j > i+1$, it will need to decrypt the second entries of the tuples $I_{i+1}, \ldots, I_{j-1}$. But, for any k, in order to be able to decrypt the second entry of I_k one will need K_k, which in turn is only available by decrypting $E_{P_{H_k}}(K_k)$. But only H_{k+1} $(k = i, \ldots, j-2)$ can decrypt these values.

□

6 CONCLUSION

This paper introduces a practical mobile agent scenario which could be used as a basis for an e-commerce setting. By defining strict tasks for each agent, we prevent sensitive data from being exposed to the possibly malicious merchants during the bidding process. Even though we use a multi-hop agent with fixed itinerary to travel to the different merchants, merchants see only a very small part of the agent's itinerary. In case of failures in the agent's itinerary, a help protocol can be executed to overcome this difficulty. If all of the hosts function properly this help protocol is not needed. Protocols are described in a general so that different algorithms could be used depending on other (performance and storage) factors.

REFERENCES

Algesheimer, J., Cachin, C., Camenisch, J., and Karjoth, G. (2001). Cryptographic security for mobile code. In *Proceedings of the IEEE Symposium on Security and Privacy*, page 2. IEEE Computer Society.

Blakely, G. (1979). Safeguarding cryptographic keys. In *Proceedings AFIPS 1979 National Computer Conference*, pages 313–317. AFIPS.

Cachin, C., Camenisch, J., Kilian, J., and Müller, J. (2000). One-round secure computation and secure autonomous mobile agents. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 512–523. Springer-Verlag.

Desmedt, Y. and Frankel, Y. (1994). Perfect homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM J. Discret. Math.*, 7(4):667–679.

Lai, C., Gong, L., Koved, L., Nadalin, A., and Schemers, R. (1999). User authentication and authorization in the Java platform. In *15th Annual Computer Security Applications Conference*, pages 285–290. IEEE Computer Society Press.

Sander, T. and Tschudin, C. F. (1998a). Protecting mobile agents against malicious hosts. *Lecture Notes in Computer Science*, 1419:44–60.

Sander, T. and Tschudin, C. F. (1998b). Towards mobile cryptography. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA. IEEE Computer Society Press.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, 22(11):612–613.

Singelée, D. and Preneel, B. (2004). Secure e-commerce using mobile agents on untrusted hosts. Technical report, COSIC Internal Report.

Tan, H. K. and Moreau, L. (2002). Certificates for mobile code security. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 76–81, New York, NY, USA. ACM Press.