Untraceability of Mobile Agents

Rafał Marek Leszczyna
Joint Research Centre,
Ispra, Italy
Institute for the Protection and security of the Citizen
Rafal.Leszczyna@jrc.it

Janusz Górski
Gdańsk University of Technology,
Gdańsk, Poland
Department of Software Engineering
jango@pg.gda.pl

Abstract

User untraceability is an issue of increasing importance as it helps in implementing user privacy in modern Internet services. In mobile agent environments each platform can easily constitute an anonymizer for an agent, assuring untraceability of the agent sender and the receivers. Based on this observation two protocols are proposed. Comparing to other solutions, the advantage of the protocols is that they support agent's autonomy in choosing the migration path. Security of the protocols was analyzed against known generic and traffic analysis attacks. The first protocol assures more balanced distribution of processing over all agent platforms but an attacker can compromise untraceability if he/she manages to perform the costly cordoning-off attack. The second eliminates this vulnerability at the cost of putting more computation workload on the source platform and restricting agent autonomy in the beginning of its route. Some implementation decisions aiming at improving performance of the proposed protocols are also presented.

1. Introduction

1.1. The need for anonymity

Anonymity is the property that ensures that a user may use a resource or service (the items of interest) without disclosing his/her identity [1]. Different *settings* [2] of anonymity may be proposed depending on the way of accessing a resource by a user. The most often considered setting refers to the *anonymity of communication* [2], where one item of interest is sending and another is receiving messages. Three models of communication anonymity are defined. *Sender anonymity* assures that it is not possible to identify the sender of a message. *Receiver anonymity* means that the intended recipient cannot be inferred from the message. *Relationship anonymity* means that it is untraceable who communicates with whom. Although the above models refer to communication anonymity, they can be easily adapted to other types of anonymity, for instance that related to accessing resources through a network.

Anonimity plays a crucial role for variuos activities conducted in the Internet. For example, in health counseling, a patient suffering from an embarrassing disease or from an addiction may wish to stay anonymous when asking for an advice [3]. Gulcu et al [4] describe four categories of the Internet applications where the anonymity is required[1]. These are: discussion of sensitive and personal issues, information searches, freedom of speech in intolerant environments and polling/surveying. For some of these uses, the anonymous access was an enabler and contributed to increasing the popularity of the Internet.

Moreover, currently many service providers have developed their tracking tools [5], so anonymity is being requested even for the services which did not required it previously. Users, who are bothered with advertisements, offers and so on, wish to be sure that visiting a new portal would not result in flooding them with unwanted e-mail, add-aware and other spam even if the functionality they take advantage of is not confidential itself.

1.2. The state of affairs

The Internet is based on the TCP/IP. The protocol has numerous advantages resulting mainly from its simplicity, nevertheless it provides weak support for anonymity of communication. The source and destination addresses are stated explicitly allowing easy tracing who is sending what.

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

1.3. Attacks against untraceability in TCP/IP networks

The attacks can be classified as follows [4] [6].

1.3.1. Generic attacks *Generic attacks* are the most atomic activities the attacker can perform after discovering a system vulnerability. If information about the source and destination of a message is not protected, it can be read, deleted or modified by an attacker. Encryption of these data protects them from generic attacks.

1.3.2. Passive attacks *Content Correlation*: The attacker attempts to gain the information about the network traffic by comparing packets detected by *monitors*. The monitor is a software or hardware equipment allowing the attacker to view the content of packets passing the network connection at the place where the monitor is installed. Then if two packets observed by the monitors in different time have common contents or are of the same size, they are likely to be the same. *Time Correlation*: The attacker deduces the relation between packets by measuring the time relations of the packet arriving to and leaving a network location.

1.3.3. Active attacks *Isolate and Identify*: This attack is the answer of attackers for the batching of transferred packets. The attacker unable to identify a packet directly, sends its own packets to the mix, in the amount equal to the number of messages in the mix batch minus one. This causes the mix to form a batch with only one packet not known to the attacker, so he/she can easily extract it. Now the attacker is able to connect the packets leaving and entering the mix. *Message Replay*: The attacker figures out the relationships between packets entering and leaving mixes by simply replaying the messages before they enter the mixes. In unprotected environments the packets leaving the mixes will also be repeated, so the attacker could associate them.

1.4. Existing solutions for the TCP/IP

Chaum was the first who introduced the concept of the anonymous proxy, called by him a *mix* [7]. The idea comes from the observation that since the packet's source address indicates the station from which the packet was sent, we should send the packets to the intermediate station which will resend our packets further. Because after leaving the mix, its address replaces the address of previous source station nobody can learn the real originator of the packets. The concept of mix, due to its simplicity, has found many followers which has results in the introduction of multiple variants of the original protocol, both in research and practical implementations available in the Internet. The first group includes Non Disclosure Method (NDM) [8], BABEL [4], Onion Routing [9] and Crowds [10]. To the latter belong: raw remailers, Cypherpunks (Type I remailers), Mixmasters (Type II remailers) and others [11]. These solutions differ in complexity and provide various levels of protection.

In 1988 Chaum described the Dining Cryptographers problem and by this opened the alternative way of assuring the user anonymity [12] . The new solution, called DC network, is far from the idea of mixes and roughly saying it is based on interoperation of all communication partners who must share some secret information between them. However this protocol is very resource consuming because all partners are involved in transmission of each message (by means of forecasting and receiving packets to/from the others), no matter who is the real originator of the message, and apart of a few experimental cases [13] [14], the protocol has not reached any popular implementations.

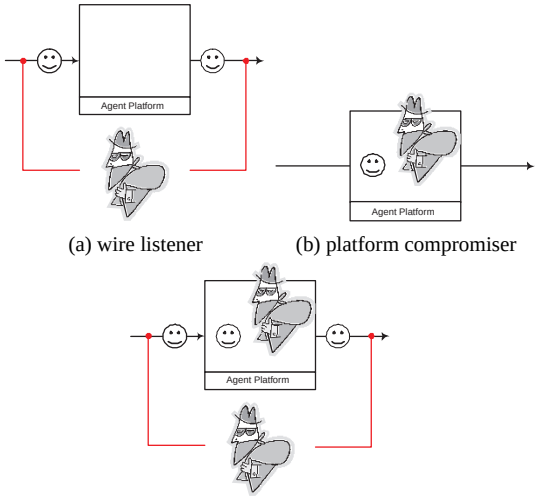
1.5. Anonymity with Mobile Agents

While applying the communication anonymity model to mobile agents, the sender anonymity assures that it is impossible to infer neither the agent owner nor the source (base) platform of the agent. The receiver anonymity guarantees the obfuscation of the agent goal which can be stated, for example, as en explicit list of platforms to visit, a task to complete or a resource to access. Relationship anonymity means that the agent source (a user or a platform) is unlinkable to its goal.

1.6. Attacks against untraceability in mobile agents networks

1.6.1. Attacks The attacks already identified for TCP/IP networks apply to the mobile agents networks with only slight modifications. In case of generic attacks the concern is the *agent's route* information instead of just the information about the source and the destination. The agent's route (or *migration path*) is the list of platforms the agent intends to visit. While considering other (non-generic) attacks in TCP/IP networks, adapting them to mobile agents networks means putting the word 'agent' in place of the word 'packet'.

1.6.2. Attackers For mobile agents we can distinguish two atomic types of the attackers against user untraceability. A *wire listener* is the attacker capable to observe the packets entering and leaving an agent platform. For example, it can be a person using a sniffer program [15]. A wire listener looks at the packets and reads their content, if the communication is not encrypted. If it is, the attacker is still able to perform some statistical analysis by finding the chronological or content relations between the packets (see Section 1.3.2). A *platform compromiser* is the attacker succeeding in compromising an agent platform. Platform compromisers are more powerful than wire listeners since not only



(a) wire listener (b) platform compromiser

(c) colluding wire listener and platform compromiser

Figure 1: Different types of attackers: a) a wire listener b) a platform compromiser c) a party formed of colluding a wire listener and a platform compromiser.

they have access to the whole information about agents (if this information is not encrypted) but also can control the agents or modify them. Finally, the attacking party may be set up of colluding attackers.

1.7. Discussion

A disadvantage of existing solutions based on TCP/IP is that users have to trust third parties which provide for the anonymization. In the most popular mix-based applications users have to choose a party which serves as an intermediate point in their communication. All user data are then transferred through the hosts belonging to this party. Another drawback of using such anonymization services in the Internet is that it requires some technical competence from users: first they have to make a right choice when selecting the anonymization provider then they must configure their operating environment accordingly.

With Mobile Agents the above concerns are automatically resolved since every agent platform establishes a mix. The mix does not need any set up by a user because it is provided as internal functionality of the agent environment. Additionally Mobile Agents introduce some level of untraceability by default. Attacker analyzing source addresses of IP packets obtains only the address of the host carrying the last platform visited by the agent on its way. These properties of Mobile Agents were already discovered by Enzmann et al [16] who proposed the onion like protocol to protect an agents' route. Other solutions for agents were also onion routing like [17]. The drawback of using these protocols is that the agent's route must be predetermined and composed on the agent's source platform so the autonomy of agents in choosing their migration path is lost.

This article presents two protocols which assure agent untraceability while not constraining its autonomy in choosing the migration patch. The protocols are little resource consuming and, apart of the first protocol and the cordoning-off attack, are resistant to known generic and traffic analysis attacks. The first protocol assures more balanced distribution of computation over all agent platforms, the second puts more processing workload on the source platform and restricts agent autonomy, but eliminates the exposure to the cordoning off attack. The protocols are planned to be implemented based on the JADE [18] middleware and incorporated into the JADE platform for the sake of experimental performance and efficiency validation. When successful, the protocols could serve as an extension of the JADE platform.

2. The Protocols

2.1. Assumptions

It is assumed that a platform guarantees that third parties (other agents, users) are not informed about the presence of other agents if the latter do not want to do so. It means that it is impossible to introduce spying agents i.e. the agents aiming at observing and following other agents. It is also assumed that each platform is well isolated, so it is impossible to learn its state from outside. Each platform owns an individual symmetric key and a private asymmetric key. A platform must also have access to all needed public keys of other platforms. However, the actual implementation of the key management is out of scope of this article. Each platform stores the identifier of the previous platform visited by an agent until the agent leaves out. This identifier is available to the agent.

2.2. Basic protocol

The *basic protocol* is dedicated to obfuscate the identity of an agent source platform while keeping the agent capable to autonomously select the next platform to visit. The protocol is intended to be as little resource consuming as possible. Encryption is employed only when necessary and only to an essential content.

Generally, the idea of the protocol is that while migrating, the agent encrypts the identifier of the last visited platform (using the public key of the present platform) and puts it to the LIFO queue stored in the agent. After achieving the goal, when the agent wishes to come back to its base platform, it uses the queue to find its way back. Down the route

back the identifiers are subsequently decrypted using each platform private key.

The pseudo-code of the protocol is presented below (Listing 1). This is important to note that the variable m is used only to illustrate subsequent steps of algorithm and is not stored explicitly in the agent state. Storing the counter of visited platforms in the agent state would allow the attacker to identify the base platform, since he/she could always read how many platforms are left to the base platform. If the attacker located himself/herself right next to the base platform, it could easily recognize that the preceding platform is the source platform of the agent. This doesn't take place in the proposed protocol. Even very close to the base platform attacker can not recognize the situation. For the same reasons the LIFO queue of platform identifiers is initially filled with a number of random values. The notation $(B_1, B_2, \ldots, B_n)$ represents the binary concatenation of the values $B_1, B_2, \ldots, B_n$. The nonce N_k is used to assure uniqueness of obtained values.

Listing 1: The basic protocol pseudo-code.

```
1. The LIFO queue of encrypted platform
   identifiers (stored in the agent's state)
   is initially filled with a number of random
   values
2. m=2
3. The agent moves to the platform AP_m
4. The agent processes its task on the platform
   AP_m
5. If mission accomplished then go to 12
6. The agent decides which platform to visit
   next (AP_m+1)
7. The platform AP_m computes the hash value of
   the predecessor identifier ID_m-1, the own
   identifier ID_m and the successor identifier
   ID_m+1 obtaining H(ID_m-1, ID_m, ID_m+1)
8. The platform AP_m encrypts the identifier
   ID_m-1, the hash value (ID_m-1, ID_m, ID_m+1)
   and the nonce N_m with its secret key K_m
   obtaining K_m(ID_m-1, H(ID_m-1, ID_m, ID_m+1),
   N_m)
9. The agent adds K_m(ID_m-1, H(ID_m-1, ID_m,
   ID_m+1), N_m) to the end of the queue of
   encrypted platform identifiers
10. m=m+1
11. Go to 3
// Coming back to the source platform (AP_1) from
the last platform on the route (AP_n)
12. m=m-1
13. The agent moves to the platform AP_m
14. If AP_m == AP_1 then finish()
15. The platform AP_m takes out (and does not put
    it back later) the first encrypted platform
    identifier available from the LIFO queue of
    encrypted platform identifiers
16. If the queue was not compromised the taken
    part should be the one encrypted with the
    secret key K_m of the platform, if it is not
    then go to 20
17. The platform decrypts the encrypted platform
    identifier and obtains ID_m-1, H(ID_m-1, ID_m,
    ID_m+1) and the nonce
18. The platform verifies the hash value
    H(ID_m-1, ID_m, ID_m+1)
19. If the verification does not fail then go to
    12
// The verification of hash value has failed -
the string of encrypted platform identifiers was
compromised - perform the emergency scenario
20. Perform the emergency scenario
```

2.3. Extended protocol

The *extended protocol* was designed as an extension of the basic protocol to make it resistant to the cordoning-off attack, i.e. the attack against the anonymity of an agent source platform which during the security analysis (described in 3) was recognized as effective. In the protocol, the source platform arbitrarily chooses a particular number of platforms the agent has to visit initially and creates the list of their encrypted identifiers. This serves as the initial route letting the agent to obfuscate its source address. After leaving this initial route, the agent is free to make decisions about which platforms to visit next. It autonomously roams the network to achieve its goal and after succeeding returns to the last platform of the initial route. Then, to come back to its source platform, it must follow the initial route in the reverse order. In the extended protocol, only the last platform of the route knows the destination address but it is not able to recognize the source address. The first platform knows the source address but without the knowledge of being the first in the route (it could be just another platform on the route).

The protocol is presented in the pseudo-code (Listing 2). As in the pseudo-code of the basic protocol, the variables m and i are used only to illustrate subsequent steps of algorithm and are not stored explicitly in the agent state.

Listing 2: The extended protocol pseudo-code.

```
// Preparations
1. The agent's base platform defines the initial
   route of an arbitrary length i: AP_2, AP_3,
   ...AP_i+1
2. The platform encrypts the route into the
   string - K_2(ID_3, H(ID_1, ID_2, ID_3), N_2),
   K_3(ID_4, H(ID_2, ID_3, ID_4), N_3), ..., K_i(ID_i+1,
   H(ID_i-1, ID_i, ID_i+1), N_i) using the public keys
   K_2, K_3 , ..., K_i of the chosen platforms, and
   stores it into the agent's state
3. The platform produces the return route string
   K_i+1(ID_i, H(ID_i+1, ID_i), N'_i+1), K_i(ID_i-1,
   H(ID_i+1, ID_i, ID_i-1), N'_i), ..., K_2(ID_1, H(ID_3,
   ID_2, ID_1), N'_2) using the public keys K_2,
   K_3, ..., K_i+1, and stores it into the agent's
   state
```

4. The platform encrypts the agent's goal G using the public key K_{i+1} of the last platform AP_{i+1} on the initial route, obtaining $K_{i+1}(G)$, and stores it into the agent's state

// Following the initial route

5. m=2
6. The agent moves to the platform AP_m
7. If the string of encrypted platform identifiers is empty go to 14 // at the same time it means that the agent reached the last platform on its initial route (and m=i+1 at the moment)
8. The platform AP_m decrypts the dedicated part of the string of encrypted platform identifiers $K_m(ID_{m+1}, H(ID_{m-1}, ID_m, ID_{m+1}), N_m)$ obtaining the identifier of the succeeding platform ID_{m+1}, the nonce and the hash value $H(ID_{m-1}, ID_m, ID_{m+1})$
9. The platform verifies the hash value $H(ID_{m-1}, ID_m, ID_{m+1})$
10. If the verification fails go to 31
11. The platform removes the $K_m(ID_{m+1}, H(ID_{m-1}, ID_m, ID_{m+1}),N_m)$ entry from the string of encrypted platform identifiers
12. m=m+1
13. Go to 6
14. The platform AP_m, where m=i+1 at the moment, decrypts $K_m(G)$ to unhide the agent's goal
15. The identifier ID_{i+1} of the platform AP_{i+1} is stated explicitly in the agent's state

// Autonomous migration to achieve the goal

16. The agent migrates autonomously from one to another agent platform to achieve its goal G
17. The agent decides to return to its base platform

// Returning to the base platform

18. The agent moves back to AP_{i+1}
19. The identifier ID_{i+1} of the platform AP_{i+1} is removed from the agent's state
20. The platform AP_{i+1} decrypts the dedicated part of the string of encrypted platform identifiers $K_{i+1}(ID_i, H(ID_{i+1}, ID_i), N'_{i+1})$ obtaining the identifier of the proceeding platform ID_i, the hash value $H(ID_{i+1}, ID_i)$ and the nonce
21. The platform verifies the hash value $H(ID_{i+1}, ID_i)$
22. If the verification fails go to 31
23. m=m-1
24. The agent moves to the platform AP_m
25. If the string of encrypted platform identifiers is empty then finish() // at the same time it means that the agent reached its base platform (and m=1 at the moment)
26. The platform AP_m decrypts the dedicated part of the string of encrypted platform identifiers $K_m(ID_{m-1}, H(ID_{m+1}, ID_m, ID_{m-1}), N'_m)$ obtaining the identifier of the proceeding platform ID_{m-1}, the nonce and the hash value $H(ID_{m+1}, ID_m, ID_{m-1})$
27. The platform verifies the hash value $H(ID_{m+1}, ID_m, ID_{m-1})$
28. If the verification fails go to 31
29. The platform removes the $K_m(ID_{m-1}, H(ID_{m+1}, ID_m, ID_{m-1}), N'_m)$ entry from the string of encrypted platform identifiers
30. Go to 23

// The verification of has value has failed

31. Perform the emergency scenario

2.4. Design decisions

During the design of the protocols the reference agent platform was JADE [18], which is envisaged to be the platform on which the practical examinations of the protocols will be performed. JADE was selected from a nine FIPA compliant agent platforms, after studying available agent platform evaluations and assessing the platforms against the criteria related to the platform accessibility and availability, the level of support for the platform, its maintenance etc [19].

According to JADE documentation, the containers are identified using `ContainerID` class which wraps the characters' string of the URL-alike container name. This means that in current representation the container identifier may take up to 2083 characters because this is the limitation for URL addresses imposed by the internet browsers. Having this in mind we can perform some introductory estimations of the performance of the generic operations (mainly encryption related) necessary in security protocols. Knowing the fact that using encryption always reduces the efficiency of the solution, the protocols were designed to involve only necessary encryption steps. The symmetric encryption instead of asymmetric is used to provide confidentiality of the route information. The simplest hashing is employed to obtain basic guaranties of integrity. As the symmetric cipher the official AES Algorithm was chosen - Rijndael [20] because its efficiency and security were comprehensively examined during AES selection process. For the reasons of efficiency and simplicity the MD5 [21] algorithm was selected to be used for hashing.

The tables 1 and 2 show the results of performance tests of AES Candidate Algorithms obtained on a Pentium Pro machine[2][22]. The examinations of MD5 performance indicate that the hashing is about four times faster [23]. These results accompanied with the knowledge of the representation of platform address stored in agents were the reason why the proposed protocols store route information in concatenated representation rather than the onion-like [9] [16]. The latter would impose linear growth of computational costs of protection of the route information since each mix-platform had to encrypt/decrypt not only the one necessary identifier but the whole already constructed onion. With this careful selection of route representation and reduction of used encryption the sketchy analyses of protocols performance shows that decryption of the address of preceding platform will take about 0.001 s, meaning that platforms will be able to process one thousand agents per

second. This performance might be significantly improved (for about an order) when changing the string representation of agent platforms identifiers to the numerical (32-, 64- or 128- bit).

3. Security Analysis

This chapter presents the results of the examination of security of the two presented protocols against the traffic analysis attacks (see Sections 1.3 and 1.6). In addition to the cryptographic means used in the protocols (i.e. computing hash values, using nonces and encrypting the obtained binary strings), agents may also be encrypted as a whole (i.e. the agent's code and the state are encrypted), while being transferred from one platform to another. This functionality however, is not a part of the proposed protocols but is optionally provided by third parties (usually agent platforms). For the sake of the completeness of the analyses both cases, with and without encryption, are discussed. This may also give the picture how the proposed protocols can get along with the protection mechanisms already embedded in agent platforms. As we will see encrypted agents are better protected from wire listeners (so hybrids) attackers. But at the same time the agents are less efficient, which can be significant while experiencing heavy traffic. The use of hardware acceleration can help to resolve this problem. From other side, encrypting agents while sending them to other platforms is often the functionality embedded into the agent platform by default, to satisfy other (non untraceability related) security needs, mainly confidentiality.

3.1. Basic protocol

The protocol's objective is to protect the identity of the agent base platform without restricting agent's autonomy in choosing its migration path. To enable this autonomous migration, the agent goal has to be stated explicitly and can not be hidden from agent platforms. The platform compromiser can read, delete or modify the goal of the agent but can not compromise its source. The wire listeners could also read, delete or modify the goal, if agents were not encrypted. There is an attack which can lead the attacker to recognize the base platform, although it is difficult to perform in case the agent is encrypted. If the attacker manages to cordon off[3] the agent platform, then he/she can deduce if the platform is the base for an agent by means of the content correlation attack (Figure 2). In practice, compromising all the platforms surrounding the chosen one is very difficult (if not impossible) since the set of surrounding platforms usually will be large (in particular it might comprise all platforms), distributed across wide geographical area. This picture gets even more complicated if the platforms are deployed on mobile devices such as PDA's or mobile phones, communicating from different, often moving, locations (e.g. cars, aircrafts and ships). With unencrypted agents it is much easier to conduct this attack - usually platforms use only one network connection. This is an additional argument for encrypting agents.

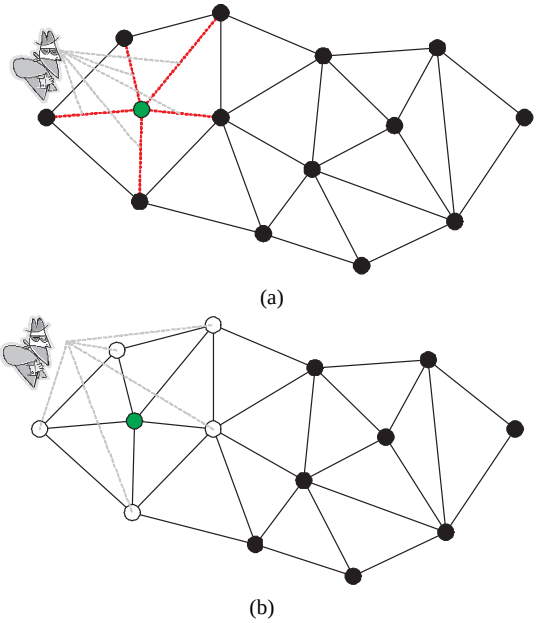


Figure 2: Cordoning off the agent platform is the only way to learn when the platform is the base for an agent. When agents are not encrypted listening at all links to the agent platform is enough - figure a). If agents are encrypted then compromising all surrounding platforms are necessary - figure b).

The protocol protects the agent's route from deleting or exchanging entries, but it is not resistant to collusion attack described by Westhoff et al [17]. The attacker succeeded in tampering two agent platforms may remove the information about all the intermediate platforms without that fact being recognized. Nevertheless the deletion and the insertion attacks don't aid disclosing the identity of the base platform.

3.2. Extended protocol

This protocol is as an extension of the basic protocol aiming to withstand the cordoning-off attack. As the previous protocol, it protects the identity of an agent base plat-

2 The results were obtained by running the tests on a machine with an Intel Pentium Pro 200 MHz CPU and 128 MB RAM running Windows NT 4.0 with Service Pack 4. Performance wise this is virtually identical to the NIST reference platform (64 MB RAM and running Windows 95).

3 If agents are not encrypted listening at all links to the agent platform is enough. If agents are encrypted then compromising all surrounding platforms are necessary.

Encryption Speed (kbit/s)	DES (56 bit)	Triple DES (168 bit)	IDEA	MARS	RC6	Rijndael	Serpent	Twofish
128 bit key	10508	4178	12820	19718	26212	19321	11464	19265
192 bit key	n/a	n/a	n/a	19760	26192	16922	11474	19296
256 bit key	n/a	n/a	n/a	19737	26209	14957	11471	19275

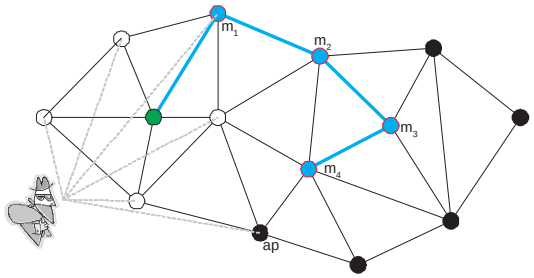
Table 1: Encryption performance of the AES Candidate Algorithms in Java [22].

Decryption Speed (kbit/s)	DES (56 bit)	Triple DES (168 bit)	IDEA	MARS	RC6	Rijndael	Serpent	Twofish
128 bit key	10519	4173	13018	19443	24338	18868	11519	18841
192 bit key	n/a	n/a	n/a	19670	24382	16484	11514	18841
256 bit key	n/a	n/a	n/a	19489	24279	14468	11533	18806

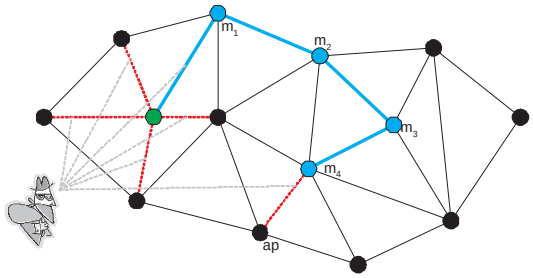
Table 2: Decryption performance of the AES Candidate Algorithms in Java [22].

form whilst the agent is still able to make autonomous decisions concerning its migration path. Only an attacker located after initial route, being it a platform compromiser or a wire listener, can read the agent's goal. But at the moment none of them is able to recognize the agent's base platform.

When the attacker is capable of compromising agent platforms he/she must tamper with all the platforms surrounding the base platform but also it must be in control of one platform passed by the agent on its route but not belonging to the initial route, apart of its last platform (Figure 3a). The attacker performing traffic analysis by means of listening on network connections must surround the agent base platform and the link passed by the agent on its route but not belonging to the initial route (Figure 3b). Then in both cases, the attacker, to succeed, must be also able to recognize when the agent passing the independent platform is the same as the one which left the source platform.



(a)



(b)

Figure 3: To discover the agent's source and the goal at the same time the platform compromiser must compromise all the platforms surrounding the base platform and also it must be in control of one platform passed by the agent on its route but not belonging to the initial route, apart of its last platform - figure a). The wire compromiser must surround the agent base platform and the link passed by the agent on its route but not belonging to the initial route - figure b).

4. Summary

The analysis of complexity of the protocols (see Section 2.4) led to careful selection of encryption tools, the encryption steps involved in the protocol and the representation of the route information. The efficiency considerations led to distinction between two protocols: the basic and the extended one. The basic protocol requires the agent to compute the appropriate route information on each container of its migration path. This results in the equally balanced deployment of workload on all containers. The extended protocol shifts a significant part (the encryption related) of computations to the source platform which is supposed to define the initial route of the agent. However, the user may control the length of the route. The longer the route, the more difficult is to figure out the source address of the agent. But the same time, the source platform must compute a longer route information string. While traversing the route, only decryption is needed, and only on containers belonging to the initial part of the route.

Compromising the protocols is hard because the attacker must gain control over numerous containers (or network connections) distributed over the Internet. Cordoning-off the source platform could lead to breaking the source anonimity in case of the basic protocol. If agents are not encrypted, this attack can be particularily efficient because

surrounding the base platform might mean that listening to just one (the only one existing) connection to the platform would be enough. This is a strong argument for encrypting agents, however this encryption brings additional overhead. The conclusion is that the basic protocol is appropriate for the environments where the encryption of agents is already in place, and especially if this encryption is provided by a hardware solution. The extended protocol protects from the cordoning-off attack even against a wire listener and when agents are not encrypted. The attacker gains the knowledge about the source platform but is unable to deduce the agent destination until he/she cover the whole initial route. Since this protocol imposes less requirements for the encryption than the basic version, it is applicable for the environments where the encryption is not provided by default.

Comparing to the onion-like protocols first proposed by Westhoff et al [17] and followed by Enzmann [16] the described solutions have the advantage that the agent is given the liberty in choosing its migration path. In case of the basic protocol, this feature is provided from the beginning, right after the agent leaves the source platform. In the extended protocol, the agent obtains the autonomy after traversing the initial route. The autonomous choice of the roaming path by agents is one of the properties which distinguish mobile agents from other software technologies. It has numerous potentially useful implications: agents may adapt their route to the network traffic conditions, may decide to come back when achieving the goal earlier than anticipated - to save time or to be more competitive than others (auction bidding) etc. Restraining this functionality takes away a part of the virtue of the mobile agents paradigm.

The described protocols are planned to be implemented and validated experimentally upon the JADE platform. If the tests succeed, the protocols could be proposed as an extension of the JADE platform.

Acknowledgement

We would like to thank Marc Wilikens for suggesting valuable improvements to the paper.

References

[1] National Institute of Standards and Technology (NIST). *Common Criteria for Information Technology Security Evaluation - Part 2: Security Funtional Requirements*. U.S. Government Printing Office, 1998.

[2] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. draft v0.21. 2004.

[3] EU IST-2002-507591 PRIME. Requirements version 0 part 3: Application requirements.

[4] Ceki Gulcu and Gene Tsudik. Mixing email with babel. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 2. IEEE Computer Society, 1996.

[5] The lucent personalized web assistant.

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

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

[8] A. Fasbender, D. Kesdogan, and O. Kubitz. Variable and scalable security: Protection of location information in mobile ip, 1996.

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

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

[11] Anonymity and privacy on the internet.

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

[13] Experimental crypto software: Dcchat.

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

[15] National Institute of Standards and Technology (NIST). *An introduction to computer security: The NIST handbook. National Insitute of Standards and Technology (NIST) Special Publication 800-12*. U.S. Government Printing Office, October 1995.

[16] Matthias Enzmann, Thomas Kunz, and Markus Schneider. Using mobile agents for privacy amplification in the trade with tangible goods. In *International Conference on Computing, Communications and Control Technologies (CCCT'04)*, August 2004.

[17] Dirk Westhoff, II Markus Schneider, Claus Unger, and Firoz Kaderali. Protecting a mobile agent's route against collusions. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 215–225. Springer-Verlag, 2000.

[18] Jade-board.

[19] Rafał Leszczyna. Evaluation of agent platforms. Technical report, Institute for the Protection and security of the Citizen, Joint Research Centre, June 2004.

[20] NIST Computer Security Resource Center. Aes algorithm (rijndael) information.

[21] Ron Rivest. The MD5 message digest algorithm, April 1992.

[22] Andreas Sterbenz and Peter Lipp. Performance of the aes candidate algorithms in java. In *AES Candidate Conference*, pages 161–165, 2000.

[23] Wei Dai. Speed comparison of popular crypto algorithms.