Behavioral Pattern Analysis of Secure Migration and Communications in eCommerce using Cryptographic Protocols on a Mobile MAS Platform

S. Pozo, R. M. Gasca, R. Ceballos
Computer Languages and Systems Department
ETS Ingeniería Informática, University of Seville
Avda. Reina Mercedes S/N, 41012 Seville, Spain
sergio@us.es, {ceballos,gasca}@lsi.us.es

Abstract

Mobile Multi-Agent Systems (MAS) systems can be used with real success in a growing number of eCommerce applications nowadays. Security has been identified as numerous times by different researchers as a top criterion for the acceptance of mobile agent adoption. In this paper we present an in-depth analysis of behavior patterns of a mobile MAS platform when using different cryptographic protocols to assure communication and migration integrity and confidentiality. Different use case sceneries of eCommerce applications as well as many other aspects have been studied, such as overhead, different communication patterns, different loads and bandwidth issues. This work is also extensible to other mobile and non-mobile MAS platforms. The results obtained can be used and should be taken into account by designers and implementers of secure mobile and also non-mobile agent platforms and agents.

1. Introduction

Mobile Multi-Agent Systems (MAS) can be used with real success in a growing number of eCommerce applications nowadays. For example, there are a lot of research efforts in agent cooperation and negotiation protocols and their applications, such as cooperating agents in problem resolution and negotiating agents in eCommerce. MAS are also being used with great success in Tele-Assistance platforms for elderly care [21, 22].

Most negotiation and cooperation protocols that are widely deployed over big networks or over Internet have been used with mobile MAS. Security has been identified numerous times by different researchers as a top criterion for the acceptance of mobile agent adoption [1]. There are even many domains of application which have specific security requirements imposed by legislation, like the applications used in health care and eCommerce environments. There is also a trade off between trust and security, and the overhead created by cryptographic protocols that should be taken into account when designing eCommerce applications, since it can affect severely its usefulness.

In this paper we present an in-depth study of behavior patterns of a mobile MAS platform when using cryptographic protocols to assure communication and migration integrity and confidentiality. Different eCommerce use case sceneries as well as many other aspects have been studied, such as overhead, different communication patterns, different loads, and bandwidth issues. This work, as we are going to expose below, is also extensible to other mobile and in some cases non-mobile MAS platforms. The test system has been implemented from a previous work [11], which is based on the possibility of using secure tunnels to mobile MAS.

The rest of the paper is organized as follows: In section 2 we present a brief introduction to security problems of platforms and mobile agents, and related works. In section 3 we expose an overview of a secure tunnel approach in which we have based our test system. In section 4 we propose our testing methodology and describe the testing environment, and present the results obtained, discussions, and conclusions about them. We conclude this work in chapter 5 with general conclusions and future research works suggested during this work.

2. Mobility and security issues in mobile MAS. Related works

Mobile software agents are goal-directed programs capable of suspending their execution state on one platform and moving to another, where they resume their execution. Static agents have the same characteristics except that they are not mobile, i.e. they cannot move from the system they are running on (which is also the system which created them.)

There are different kinds of mobility: strong, in which code and execution state are moved; and weak, where only the code is moved. That is, every time the agent arrives at a different destination, it runs back from the beginning, or from a marker expressly placed for this purpose.

Mobility can also be classified as single hop and multi-hop. Most security problems which arise in mobile multi-agent paradigm occur with multi-hop mobile agents. This is due to the fact that only the home platform (the platform where the agent was first created) is trusted. That is, if we are not sure that a mobile agent comes from a trusted source, on which we know that neither the code nor the state have been altered, we cannot know whether this agent will carry out an unplanned activity on the destination platform.

The most secure location for an agent is its home platform. Although neither agents nor home platforms are invulnerable, a number of conventional techniques can be applied to construct adequate defenses. Each time an agent migrates, security risks arise, and so it is needed a way to transmit this trusted environment to other platforms where agents may travel. The greatest problem with multi-hop MAS is just the trust relationship which can be established in single-hop MAS between two platforms thanks to the security mechanisms derived from Client/Server architecture. These trust relationships are not transitive, nor have to be bilateral.

There are some practical solutions for securing communications and migration in multi-hop MAS, but the vast majority of them include restrictions on itineraries. For example, it is possible to set up a restricted itinerary in which all platforms have mutual trust in all others, so agents can move freely between them (each platform signs and/or encrypts the agent before it migrates).

In an earlier work [11], we proposed a different way to secure agents based on secure tunnels. We obtained a secure agent community usable in production eCommerce, health care and Tele-Assistance environments with existing technology, with little impact in network topology, and without having to modify existing agent applications or agent platforms. Specifically we were looking for a system that provide integrity, confidentiality, data origin authentication, MAS independence, compliance with standards, existence of cryptographic acceleration hardware, cost-saving and reuse of existing software and hardware infrastructure. Only communications and migration were protected. No direct protection to malicious agents or platforms was provided, as there are other techniques for that purposes, such as fault isolation [2], safe code interpretation [20], state appraisal [3], path histories [4, 5], proof carrying code [6], execution tracing [7], environmental key generation [8], computing with encrypted functions [9], code obfuscation [10], and others.

Aglets [13] have been used for our reference implementation in the laboratory, since it is Open Source, cross-platform, easy to develop, has weak mobility (the maximum mobility a Java-coded MAS could have), high acceptance and relatively good documentation [14].

We have focused our research in the parameterization of behavior patterns of secured mobile agents in eCommerce: behavior with different agent loads, behavior modification by the use of different compression algorithms, behavior with different communication patterns… concluding with a performance analysis. Other issues related to the use of our particular approach to securing communications and migration in mobile MAS have arisen, such as scalability, NAT and proxying issues. All of these collateral issues, as well as an in-depth study of the applicability of the cryptographic protocols used in this work, have been discussed in more detail in a previous work [11].

The conclusions derived from this paper are easily applicable to other mobile MAS platforms, as our security mechanism [11] is applicable in a wide variety of them without having to modify platforms or agents themselves. Moreover, the cryptographic protocols used are based on standards, and can easily be encountered implemented at platform level in many mobile MAS platforms.

3. Overview of secure tunnels

Tunneling is the capability of encapsulating one protocol within another, using this second protocol to traverse network nodes. A secure tunnel encapsulates an insecure protocol (like FTP or HTTP) within a secure one (like SSL [19] or TLS [15]). Tunnels may also be used to bypass firewalls, and are also vulnerable to denial of service attacks, since they use a public and untrusted network as transmission media.

We are going to give a very brief explanation of cryptosystems applied to mobile MAS platform, because an in-depth study is given in [11], where we developed a trusted agent community with applications which generate application independent secure tunnels.

- Stunnel [16] is an application which acts as SSLv3 server and/or client, providing a secure SSL-based secure tunnel (wrapper) for insecure protocols or applications with the only need of the installation of the application in each of the systems that needs to secure. Stunnel is distributed under a GPL license and has versions for Microsoft Windows, some flavors of UNIX and many other OS. Stunnel also supports cryptographic accelerator hardware and client and server authentication with X.509 digital certificates (as SSL does).
- SSH [17] is a protocol for securing network services over an insecure network. It is traditionally used for protecting insecure UNIX protocols such as telnet, rlogin, etc. Moreover SSH can be used to secure other services creating a wrapper around them using a local port redirection scheme very similar to the one used by Stunnel. SSH is widely accepted by the scientific community as being a trusted security protocol. SSH architecture is very similar to SSL or TLS one, and provides basically the same functionality.
- Zebedee [18] is another Open Source application used to create secure tunnels with implementations in Windows, UNIX, Linux, Java and Ruby. Zebedee (from its documentation) has a small memory footprint and low wire protocol overhead. Zebedee uses a plain Diffie-Hellman protocol for its (weak) key agreement process and a symmetric key cryptographic algorithm, Blowfish. Zebedee does not provide any features for data integrity. We have included this application in our tests to verify the claims about the overhead compared to other protocols.

4. Behavior patterns of cryptographic protocols in eCommerce applications

Different behavior patterns have been analyzed, such as

- Behavior in single-hop and multi-hop with different migration and communication patterns
- Behavior with different agent loads
- Behavior modification by the use of different compression algorithms (before encryption phase)

We have also done a complete performance analysis, since performance could be a discriminatory factor in several environments, and more especially in eCommerce. Moreover, some issues have also been detected during the tests, as we are going to explain below.

4.1. Environment

During our experiments there have been used two computers connected between a dedicated 100Mbit Ethernet switch isolated from the rest of the network. All results have been taken when the machines were fully dedicated and with a fresh installation of the operating system. Both machines are Pentium 4 1,7GHz processor with a 400MHz FSB and 256Mb DDR266 RAM. Both were running SuSE Linux 9.0 fully patched (at the time of the writing of this paper) and compiled for i586 architecture. Linux Kernel 2.4.21-99, JDK 1.4.2b28 and Aglets 2.0.2 were the base software used.

- Aglets 2.0.2. Aglets HTTP proxy feature have been used to overcome a tunnel dodging problem [11], as we think this is the way Aglets are going to be used in a real environment. No embedded platform authentication has been used in the tests.
- OpenSSL 0.9.7b. This library provides algorithms used by other applications and protocols, in this case Stunnel and OpenSSH. Default configuration was used.
- Stunnel 4.04 PTHREAD. 128-bit AES encryption have been used instead of 3DES, since AES is faster than 3DES and 3DES is also being replaced with AES. 1024-bit RSA certificate based authentication scheme and no compression have also been used, since Stunnel does not support it, as it uses SSL. Stunnel was a bit difficult to configure because of its lack of up to date documentation.
- OpenSSH 3.7.2.1p2-18. 128-bit AES encryption, 1024-bit RSA certificate based authentication and no compression have been used. We got some stability problems when transferring big agents

(bigger than 3Mb). We will explain more about it later on this chapter, as well as behavior changes when using compression.

- Zebedee 2.5.2 with libBlowfish 0.9.5a, libBzip2 1.0.1 and zlib 1.1.4. 128-bit Blowfish and SHA1 hashes (160 bit) have been used. Also compression has been deactivated. No embedded authentication mechanism has been used because of unusability reasons explained before in this paper and in other works [11].

4.2. Methodology and parameters

Nine variations of the same test application have been used, varying in the number of hops the agent traverses, and in the load the agent carries on. The application is composed of a mobile agent that migrates to a platform and then goes back to its home platform. The agent does not compute anything at any platforms.

For behavior recognition with different communication patterns, the number of itinerary laps performed by the agent changes between 1, 10 and 100. That is, if the number of itineraries is set to 10, then the agent goes to the destination and then to its home platform ten times.

For behavior recognition with different loads, the agent was loaded with three different data vectors (0Kb, 100Kb or 1Mb) using OpenOffice 1.1 sw645mi.dll file from its Microsoft Windows version, in order to get some degree of randomness in data content. The size of the compiled Aglet itself was 3,7Kb.

Compression was also used to test behavior changes and overhead incurred on its usage, as it is common to use compression on very low bandwidth links (typically used in B2C eCommerce applications). Two compression algorithms (Zlib and BZip2) were used in order to test the relation between different communication patterns, agent loads, and cryptographic protocols.

Special interest has been put in a comparative analysis of each behavior pattern detected and its derivates. Each test has been repeated three times. The execution time presented in the performance analysis is the average of the three results. We think that these tests cover the majority of situations present in B2C and B2B eCommerce applications.

4.3. Results and discussion

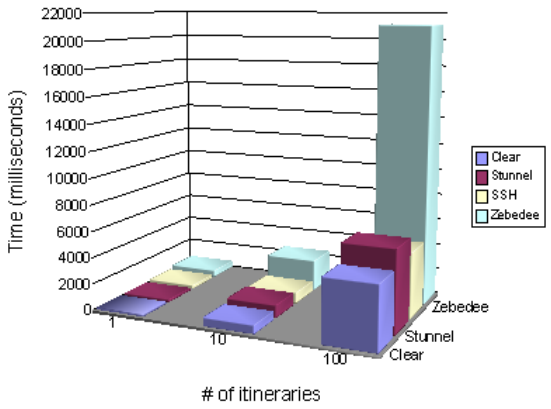
In this section we present the three main use case eCommerce sceneries which have been conducted, showing from only one itinerary to a hundred of them, with results for different agent loads. Finally behavior changes due to the use of compression are also cited. With these tests near all possibilities regarding B2C and B2B communications are covered, since migration is a form of communication in which an agent can do computations at destination platforms. Since our test agents do not compute anything, a migration can be thought as a message in terms of communication time, adding the constant time taken to serialize and reinstantiate the agent. Also the size of an unloaded agent (3,7Kb in our unloaded test aglet) could be similar to a FIPA-ACL small message, since it uses a complex syntax, and data can be embedded into messages. Testing a range of possible lengths in the 0Kb-3Kb range is irrelevant as is going to be exposed below, since the differences in behavior are too small to be taken into account. Again, we must note that the results presented in this section are directly applicable to other agent platforms, including platforms with the cryptographic protocols cited below embedded in them.

4.3.1. Behavior depending on agent communication and migration patterns. We evaluated the behavior of an aglet migrating only from one platform to another and its way back to the home platform. Then the aglet disposed itself. Then we tested all cryptographic protocols explained in an earlier section (Figure 1) for this behavior pattern (the aglet in clear, Stunnel, SSH and Zebedee). Then a ten hop pattern migration was used, and finally a hundred hop migration pattern. No compression was used, and the aglet was not loaded with any extra data (but the size of its compiled code was 3,7Kb).

For one itinerary, differences are too small to be representative, but some interesting results must be noted. There is a minimal difference between the behaviors of the aglet using no cryptographic protocol and the one tunneled by Stunnel (SSL). But the time taken by SSH is nearly twice the time taken by the clear aglet to migrate and come back. Finally, the time taken by a Zebedee tunneled aglet is three times the one of the clear aglet. This basic test has given a sight about the behavioral changes created by the overhead of the different protocols with small loads. Surprisingly, Stunnel (SSL) and SSH (which also used RSA authentication) created less overhead than Zebedee with no authentication. For the ten and a hundred itineraries tests, these results can be seen with

great detail, as overhead differences are more pronounced.

As it can be seen with more detail in the table (Figure 1), there is no difference in practical terms between behavior of a clear text aglet and of a SSH or Stunnel ones as the number of itineraries increases. It can be thought that computers used in the test were powerful enough to encrypt the migrating aglet with no performance hit. But if that were true, then Zebedee results would be different, with less overhead. Clearly Zebedee uses a poor designed protocol. The differences between Zebedee and the other two ones are too big. These differences are not related to the small differences in performance of AES (used by Stunnel and SSH) and Blowfish (Blowfish is used by Zebedee, being only a bit slower than Twofish), according to a wide range of studied cases [12, 23] in 32 bit platforms with C coded algorithms.



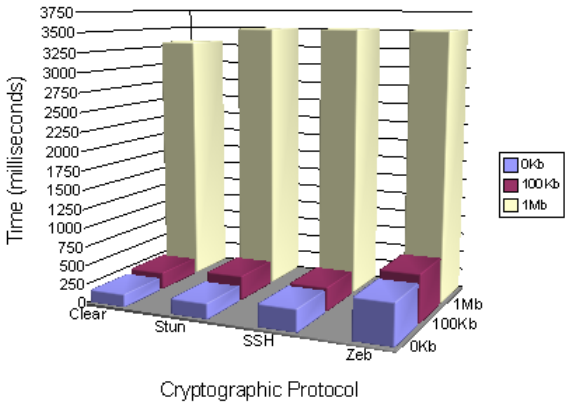
	1	10	100
Clear	174	667	4792
Stunnel	210	974	6231
SSH	301	845	5008
Zebedee	540	2420	21108

Figure 1. Behavior depending on the number of itineraries (time in milliseconds)

4.3.2. Behavior depending on agent load patterns. We evaluated the behavior of an aglet with different loads. We tested it with no load (3,7Kb was the size of compiled code), 100Kb and 1Mb. We tested all cryptographic protocols explained in an earlier section. Three sceneries were used in order to give a comparative analysis of differences in behavior due to different communication and migration patterns described in the previous section, with one itinerary, ten and a hundred ones (Figures 2, 3, and 4).

The first scenery (Figure 2) represents the one itinerary behavior pattern studied in the previous section, but with different loads. For no extra load, the results are equal to the ones of the previous section for one itinerary.

When the agent was loaded with 100Kb, the protocol overhead becomes more evident due to the fact that, even with more data to encrypt, the migration times have not increased too much. This can be seen with more detail in SSH tests, where the behaviors of 0Kb and 100Kb aglet have minimal differences (301ms Vs. 314ms). For the rest of the protocols, behavior differences are more evident. But for all of them, behavior differences between unloaded aglets and 100Kb loaded ones are minimal. A load increase is necessary to know about behavioral changes due to encryption overhead rather than protocol overhead.



	0Kb	100Kb	1Mb
Clear	174	225	3312
Stunnel	210	319	3512
SSH	301	314	3522
Zebedee	540	633	3520

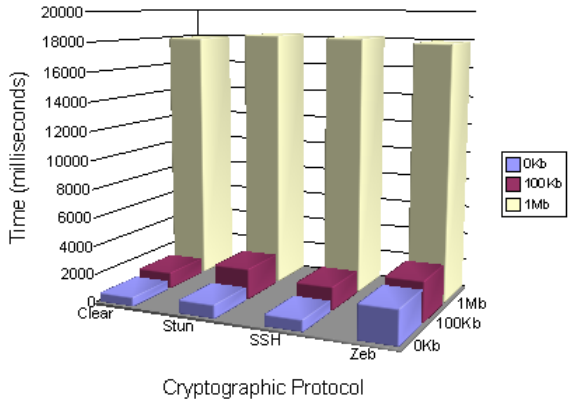
Figure 2. Behavior depending on agent load. One itinerary (time in milliseconds)

It is difficult to think that typical eCommerce agent applications would use big messages or big pieces of code because of the nature of mobile agent paradigm and eCommerce applications, but we have also studied a 1Mb loaded aglet. In this case, the overhead is created by the encryption algorithm and not by the protocol overhead. There is a severe behavioral change between a 100Kb aglet and a 1Mb one in all cases. This difference in quantitative terms is a multiplicative factor of approximately 10 in time, which is consistent with the increase in load (from 100Kb to 1Mb). Moreover there is an important fact to note: at 1Mb load, the times between different cryptographic protocols are minimal (including the aglet used with no

encryption), like if it were a performance bottleneck. This bottleneck could be due to many external, non platform and non algorithm related factors, like memory bandwidth limitation, data too big to fit in cache, transmission media limitation, etc.

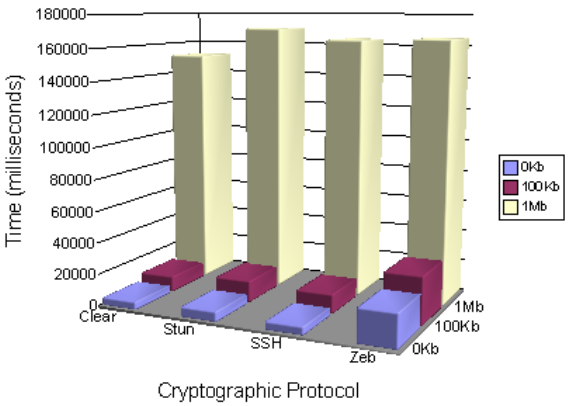
The second scenery (Figure 3) represents the ten hop itinerary pattern studied in the previous section, but with different loads. For no extra load, the results are equal to the ones of the previous section for ten itineraries.

The behavior of these aglets is very similar to the ones of Figures 1 and 2 but in a different, higher scale. Even the behavioral differences between Stunnel and SSH for one and ten itineraries are the same for the same number of itineraries, with an unloaded aglet and with a 100Kb aglet. Note that a load increase does not change behavior of the aglets in other way than the increase in itineraries did. For a hundred of itineraries (Figure 4), the results are even the same, but again at a bigger scale.



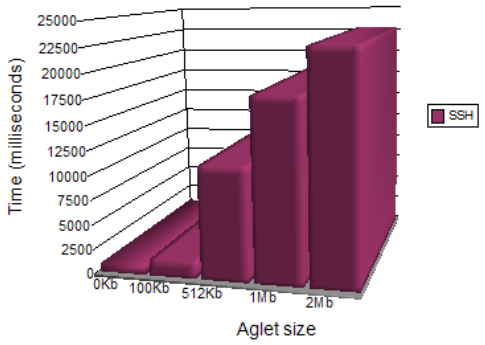
	0Kb	100Kb	1Mb
Clear	667	1155	17921
Stunnel	974	2168	18249
SSH	845	1660	18125
Zebedee	2420	2857	17896

Figure 3. Behavior depending on agent load. Ten itineraries (time in milliseconds)



	0Kb	100Kb	1Mb
Clear	4792	9709	151796
Stunnel	6231	13431	170393
SSH	5008	11961	163912
Zebedee	21108	30341	165208

Figure 4. Behavior depending on agent load. A hundred of itineraries (time in milliseconds)



0Kb	845
100Kb	1660
512Kb	11309
1Mb	18125
2Mb	22629

Figure 5. Detailed view of SSH protocol behavior (time in milliseconds)

The performance bottleneck also remains there for 1Mb load, but at a higher level. The jump between 100Kb and 1Mb is too big to analyze with detail the causes of the heavy behavioral change. Because of that, some tests with aglets loaded in the range 128Kb-512Kb were conducted. We have tested this pattern (Figure 5) with SSH, as this has been the protocol with a more stable behavior and with better performance from all of the studied ones.

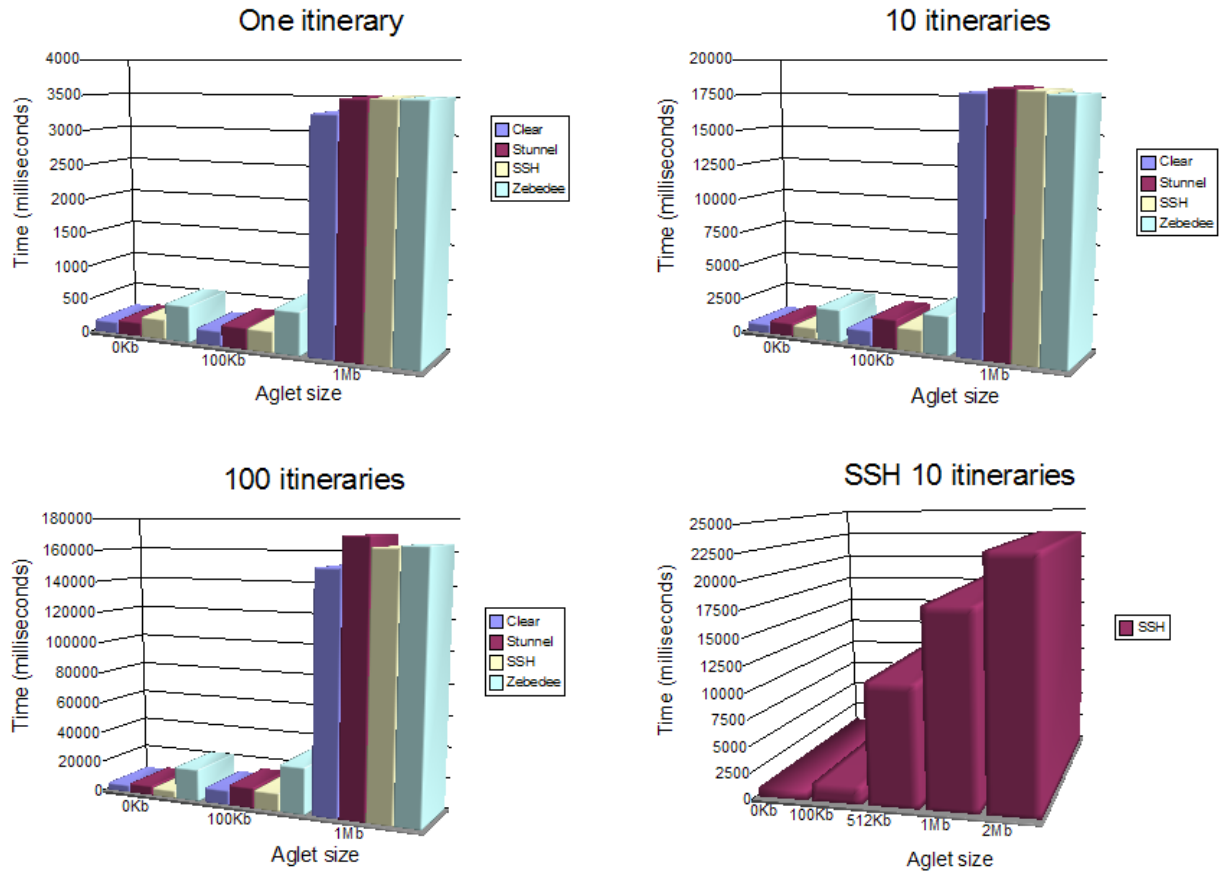


Figure 6. Comparative analysis of all studied behavior patterns

We discovered that only from 512Kb to higher ones produces the severe behavioral change, a scale change to a higher one (note the 2Mb test). For loads fewer than 512Kb only reasonable changes take place. In our opinion this confirms that the performance bottleneck should be due to external, non platform and non cryptographic algorithm related factors, like memory bandwidth limitation, data too big to fit in cache, transmission media limitation, etc. Another scale change would be possible at even higher than 2Mb loads, but we have not tested it.

4.3.3. Comparative behavior analysis depending on migration and load patterns. A graphical comparative analysis (Fig. 6) is necessary to note the behavioral changes between the different studied behavior patterns. When looking at the comparative graphic, representing one, ten or a hundred itineraries, with different load and cryptographic protocols in them, the three graphics appears to be equal.

Nevertheless the time scale is different. When increasing the number of itineraries and aglet load at the same time, the differences in execution time between the ciphered aglets and the plain ones widen. An increase in itinerary number is also more critical than an increase in terms of agent size, except for really big agents, where the performance bottleneck arises.

4.3.4. Behavioral changes when using compression. The use or not of compression is a controversial issue, because some cryptographic protocols define compression as a desirable feature (SSL, SSH) rather than a required one. Of course, interoperability is guaranteed during the negotiation phase between peers. The use of compression is an interesting issue to test, since compression is usually used in very low bandwidth links, as telephone lines or cell phones.

In our tests (with a high bandwidth link) compression have been activated and deactivated to

test its impact in all the studied patterns. The results are not convincing, with better results in some cases and worse in others. Since the use of compression cannot be changed dynamically, this issue is not easy to solve nor the focus of this work. Only SSH (Zlib) and Zebedee (BZip2) were tested using compression, as Stunnel does not support compression.

- One itinerary. Performance only gets better if aglet size is less than 100Kb. If the aglet is loaded with more data, performance gets worse.
- Ten itineraries. Only SSH improves performance with the use of compression, but in very particular cases. SSH can use a wide range of symmetric key cryptosystems. The two faster ones are AES and Blowfish. Performance only improves when using Blowfish and loads under 100Kb. Blowfish is bit a slower algorithm than AES, so the performance improvement could be due to the fact that Blowfish has to encrypt less data because it was compressed. This should be true if the compression phase is faster than the encryption one, as usually is, and only if the compression ratio is enough to justify the time spent in that phase, as there is no way to know dynamically how well or bad a file is going to compress (but we can estimate it). Nevertheless, the same fact should apply to AES: with the use of compression transmission time should also improve. But this is not the case, as compression only improves transmission time using Blowfish. This is a very strange issue we have not solved.
- A hundred itineraries. Performance always got worse results in any case.

We think that there are enough reasons not to use compression with any cryptographic protocol in a mid to high bandwidth link, mainly due to heterogeneity issues between different protocols and lack of stability at different situations. Behavior testing of agents with a very low bandwidth requires a totally new analysis, since there are a lot of new variables to take into account: system architecture (cell phone, PDA, etc.), memory constraints, programming language design and implementation issues, OS constraints, line quality, etc.

4.3.5. Conclusions about eCommerce behavior pattern analysis. During this in-depth analysis and experimentation we have assured some well known facts, such as that is faster to send a big agent than sending a high number of small agents (the same applies to messages, Figures 2, 3, 4). Other not so well know facts have also been discovered, such as that the difference in execution time of a clear agent and a tunneled one shorten when sending a burst of messages or doing a burst of migrations (Figure 2). From our point of view, due to stability and performance during our tests with different patterns, SSH2 is the best suited protocol (from the studied ones) for securing communications and migration in B2B and B2C eCommerce applications. Moreover, the use of cryptographic protocols does not suppose a significative performance penalty. We recommend using 128 bit AES (AES have a significative performance penalty at higher strengths) and at least 1024 bit RSA key for exchange. Table 1 reflects a summary of all the studied aspects.

5. Conclusions and future work

Mobile multi-agent systems introduce a great deal of security issues. Some of them are new and others are very similar to client-server paradigm ones. Security has been identified as numerous times by different researchers as a top criterion for the adoption of mobile agents in real applications.

We have presented an in-depth study of behavior patters of a mobile MAS platform when using cryptographic protocols to assure communication and migration integrity and confidentiality in eCommerce applications. Different eCommerce use case sceneries as well as many other aspects have been studied, such as protocol overhead, communication patterns, load patterns, compression use and bandwidth issues, covering a wide range of eCommerce real situations. This work is also extensible to other mobile and in some cases non-mobile MAS platforms.

During this in-depth study we have assured some well known facts. Other not so well known facts have also been discovered, such as the performance bottleneck (and scale change) Vs. burst-sending of agents and messages. From our point of view, due to stability and performance during our tests with different patterns, SSH2 is the best suited protocol (from the studied ones) for securing communications and migration. Another significant conclusion is that the use of cryptographic protocols does not suppose a significative performance penalty compared with not using them at all.

The results obtained can be used and should be taken into account by designers and implementers of secure mobile and non-mobile agent platforms and agents, as these results can also be applied to message transmission, because of the implications of behavioral changes cryptographic protocols, communication, migration, load and compression patterns have in mobile agent paradigm.

Table 1. Summary of eCommerce behavior analysis results

Pattern	*Results*
Communication and migration behavior patterns	• It is preferable to send a burst of agents or migrations than isolated ones • Bad protocol design could lead in a really big performance hit
Load behavior patterns	• For small loads protocol overhead is more evident, with few changes in performance between clear and encrypted agents • As load increases, protocol overhead becomes less evident. Performance hit derives from the cryptographic algorithm itself • For very high loads (>=512Kb), there is severe behavior change and a scale change in measured time (even for the clear agent)
Communication, migration and load blend behavior patterns	• In general, any increase in number of itineraries or messages sent, or in agent load produces an increase in execution time, but at different scales • Increasing both number of itineraries and agent load widen the difference in execution time between clear and ciphered agents more than in whatever isolated case • Increasing the number of messages sent or the number of migrations necessary to transfer data to a destination rather than sending a big block of data, decreases performance, except for the performance bottleneck case
Compression in a mid to high bandwidth media behavior pattern	• Improves performance for isolated and small messages or migrations. The performance is worse in the rest of cases • A study in very low bandwidth link is necessary to test behavioral changes in other environments

The use of compression is a controversial issue, since we have obtained very interesting, and in some cases contradictory results. These results suggested us that another in-depth analysis of behavioral changes in constrained devices with very low bandwidth (and low quality in some times) links is necessary, as MAS platforms are being developed for constrained devices, with constrained memory, processing power and energy consumption. Behavioral analysis of these mobile MAS platforms are a very interesting topic for future research, since traditional public key cryptography (i.e. based on factorization problems) cannot be used due its high computation power needs. Different cryptosystems, such as elliptic curve ones, need to be used.

6. Acknowledgements

This work was funded in part by the Fifth Framework IST program of the European Commission under the TeleCARE IST-2000-27607, and by the Spanish company SKILL Technology Group SL. The authors thank the contribution of the TeleCARE consortium members.

7. References

1. V. Roth: Obstacles to the adoption of mobile agents. Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM 2004). Berkeley, California, USA, January 2004. ISBN 0-7695-2070-7.
2. R. Wahbe, S. Lucco, T. Anderson: Efficient Software-Based Fault Isolation, October 1998.
3. W. Farmer, J. Guttman, V. Swarup: Security for Mobile Agents: Authentication and State Appraisal. Proceedings of the 4th European Symposium on Research in Computer Security (ESORICS), September 1996.
4. J. J. Ordille: When Agents Roam, Who Can You Trust? Proceedings of the First Conference on Emerging Technologies and Applications in Communications, May 1996.
5. D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, G. Tsudik: Itinerant Agents for Mobile Computing. IEEE Personal Communications, vol.2, no. 5, October 1995.
6. G. Necula, P. Lee: Safe Kernel Extensions Without Run-Time Checking. Proceedings of the 2nd Symposium on Operating System Design and Implementation (OSDI), October 1996.
7. G. Vigna: Protecting Mobile Agents Through Tracing. Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems, June 1997.
8. J. Riordan, B. Schneier: Environmental Key Generation Towards Clueless Agents. Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.
9. T. Sander, C. Tschdin: Protecting Mobile Agents Against Malicious Hosts. Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.
10. F. Hohl: Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science No. 1419, 1998.
11. S. Pozo, R. M. Gasca, M.T. Gómez-López.: Securing Mobile Agent Based Tele-Assistance Systems. 1st International Workshop on Tele-Care and Collaborative Virtual Communities in Elderly Care, TELECARE 2004 (in conjunction with ICEIS 2004). Porto, Portugal, April 2004. INSTICC Press 2004, ISBN 972-8865-10-4, pp. 62 72.
12. NIST Adavanced Encryption Standard (AES) Development Effort. CSRC Crypto Toolkit.
13. D. B. Lange, M. Oshima: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, 1998. ISBN 0-201-32582-9.
14. J. Altmann, F. Gruber, L. Klug, W. Stockner, E. Weippl: Using Mobile Agents in Real World: A Survey and Evaluation of Agent Platforms. 5th International Conference on Autonomous Agents, 2nd Workshop on Infrastructure for Agents, MAS, and Scalable MAS at Autonomous Agents. Montreal, Canada, May 2001.
15. S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright: The TLS protocol version 1.0, RFC 3546. Internet Engineering Task Force, June 2003.
16. W. Wong: Stunnel: SSLing Internet Services Easily. SANS Institute, November 2001.
17. T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, S. Lehtinen: SSH Protocol Architecture, September 2002.
18. N. Rinsema: Secure (and free) IP Tunneling using Zebedee. SANS Institute, June 2001.
19. Netscape Technologies, Inc.: Introduction to SSL.
20. J. K. Ousterhout: Scripting: Higher-Level Programming for the 21st Century. IEEE Computer, March 1998.
21. L.M. Cmarinha-Matos, H. Afsarmanesh: Design of a virtual community infrastructure for elderly care". 3rd IFIP Working Conference on infrastructures for Virtual Entrerprises, PRO-VE 2002. Sesimbra, Portugal, May 2002. Kluwer Academic Publishers, ISBN 1-4020-7020-9.
22. L.M. Camarinha-Matos, O. Castolo, J. Rosas: A multi-agent based platform for virtual communities in elderly care. 9th IEEE International Conference on Emerging Technologies and Factory Automation, EFTA 2003. Lisbon, Portugal, September 2003.
23. B. Preneel, B. Van Rompay, S.B. Örs, A. Biryukov, L. Granboulan, E. Dotax, M. Dichtl, M. Schafheutle, P. Serf, S. Pyka, E. Biham, O. Dunkelman, J. Stolin, M. Ciet, J-J. Quisquater, F. Sica, H. Raddum, M. Parker: Performance of Optimized Implementation of the NESSIE Primitives. IST-1999-12324, D21v2, February 2003.