Journal of Information Technology and Engineering
Vol. 1 No. 1 (June, 2016)

Mobile Agent Computation Results Protection with Head Sealing

Abid KHAN, Xia-Mu NIU and Yong ZHANG

[1]Information Security Technique Research Center, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, P. R. China *abidkhan_hit@yahoo.com, xiamu.niu@hit.edu.cn, zhangyong@hitsz.edu.cn*

Received: 13th March 2016 Revised: 24th April 2016 Accepted: 29th April October 2016

Abstract: *Mobile agent technology offers many solutions to various problems faced by traditional client/server technology. However some unique and challenging security problems are also introduced. One such security problem is the protection of the mobile agent from malicious hosts. Providing integrity of execution for mobile agents is the most difficult problem to address. This paper presents a new idea of "seal head protection" for mobile agents. We aim to protect the computation results of mobile agent by using a graph or list based dynamic data structure. We first convert the results of the agent's computation into a form that can be represented by a graph or list structure. Then we seal (encrypt) the head of this dynamic data structure. The cryptographic algorithm like DES or AES can be used to protect the confidentiality of this head object. The only overhead is the transfer of the secret key used for Symmetric Encryption/Decryption however, by using a stronger key transfer protocol like ElGamal or Diffie-Hellmann this problem can be solved efficiently. Experimental results suggest that the proposed scheme can be used by mobile agents against malicious host tampering in an e-commerce application.*

Key words: mobile agent security, malicious host, object sealing, secure computation, symmetric DES/AES encryption decryption.

1. INTRODUCTION

A Mobile agent (MA) is a program that can act in a computer network on behalf of a user or an application. MA has already been employed in a variety of applications with great effect such as information retrieval, workflow management systems, e-commerce applications and Network Management [1-4]. Despite their application in such diverse areas there are still some issues which need to be addressed before mobile agents can be used widely; security being the most serious and challenging issue among others needs a lot of attention. Security in mobile agent system can be divided into two major types. The first one is security of the platform from malicious mobile agents, and the second one is the security of the mobile agent from a malicious platform. The second type of security is the focus of this paper. The security of the mobile agent from malicious host can further be divided into two types: (1) security of the static code and data (2) security of the dynamic execution state of the agent.

In this paper we present a new idea we call this as "seal head protection". We have combined the ideas of pointer aliasing and object sealing for mobile agent security. Because of the aliasing effect it is difficult to analyze objects that may refer to same memory location. Aliasing occurs when two objects refer to same memory location. Since our dynamic data structure representation will use a lot of reference so we believe that it is difficult to analyze such structure. We use object sealing to seal the head of the head of the structure which represents the computation results of a mobile agent at a host. Additionally the strength of our proposed method can be increased by using some opaque predicates [16, 17]. An opaque predicate is a predicate which has a value only known prior to the obfuscator. Obfuscation is a technique of transforming a program into a form that is more difficult to understand. The behavior of the obfuscated program should be identical to the original program which is being obfuscated.

This paper is organized as follow: in section 2 we discuss related work. Section 3 discusses the proposed idea. Section 4 discusses implantation and performance evaluation. In section 5 a conclusion is given.

2. RELATED WORK

Mobile agent's execution environment is responsible for providing necessary environment and resources in order to successfully execute an agent. Methods that have been devised to protect the computation results of a mobile agent include Partial Results Authentication Code (PRAC) [11], Hash Chaining [10], Set Authentication code [12], and Ring Signature [13]. Yee proposed the idea of PRAC in which the results of an agent's computation at each host is encapsulated using MAC (Message Authentication Code). The result of an agent's execution combined with MAC of the results is called PRAC. This method requires the agent

to produce a secret key for each host, using one way function, from the initial secret key given by the originator. This method makes sure to provide forward integrity which means "None of the results calculated prior to a malicious host can be tampered". In Hash Chaining method the partial results are chained to the identity of the next host in itinerary. This method allows the originator to determine where exactly the chaining is broken if a host behaves maliciously by tampering with the partial results. Although this method provides stronger security, it is not flexible enough. Hohl [7] gave the idea of a time limited black box security using obfuscation as a protection technique. The problem with this technique was the unknown length of the protection time interval. The idea of using watermarking as a protection mechanism was presented by [8, 9]. This method attempts to detect manipulation attacks performed during agent's execution. The agent's execution creates marked results. When the agent returns to the origin host, these results are examined in order to locate the watermark. If the mark has been changed this proves a manipulation of results and malicious behavior of the host. But according to their work manipulation attacks performed by a malicious host are difficult to detect and expensive [6]. In [18] the idea of cryptographic traces was presented based on the execution tracing and cryptography. It allows the "Detection of attacks against the code, state and execution flow for mobile agents. These facts can be used to punish the attackers. However this method has some limitation e.g. mobile agent code is executed again only in case of suspicion although there is suspicion detection protocol but its cost is too high. In [19] Roth proposed the idea of transferring commitments to other cooperating agent. This agent can do tasks like storing, gathering and verifying the information. The underlying principle is the generalization of the trusted third party. These cooperating agents share secrets and decisions and have a disjoint itinerary. Each cooperating agent record the itinerary of other cooperating agent. This makes collusion attacks difficult but not impossible. However this technique is only effective if this requirement can be realistically met. [20] Proposed the idea of reference state. A reference state is a state that is produced by a non-attacking host or reference host. In this model the execution on one host is checked unconditionally and immediately on the next host, regardless of whether this host is trusted or untrusted. [21] Proposed the notion of data lockers. It is a service provided for mobile users to keep their data in secure and safe locations. In [14] the idea of Ring Signature was proposed by Rivest. A Ring Signature, in which no prior setup process and no group manager are necessary. It is a special form of generalized group signature. In [12] an original cryptographic technique called Set Authentication Code is proposed. In this technique each host exchanges a secret key with the agent owner. This key is used to calculate MAC on its results. When the mobile agent returns to the home platform this integrity proof can be verified. Roth [22] pointed out some flows in some of the proposed protocols of [23, 24, 10]. According to Roth these protocols failed because they were unable to bind the collected data by agent with its static code. He proposed fixing these protocols by binding confidential data and acquired data via constructing an agent kernel and ciphertexts. This allows authorized hosts to detect whether a ciphertext brought by an agent actually belongs to the agent. In [25] the idea of using multi-agent architecture was proposed. They used different classes of agents like task agent, data collection agent and data computation agent. The task agents are responsible for the completion of job the user wants to complete. Computation agents perform the desired computation in single hop or multi-hop fashion. Data collection agents responsible for data state collection.

2.1 Security Properties

In [10] Karjoth have defined a set of security properties which are considered as the basic guidelines for the data integrity mechanism. Here we took the liberty of modifying the original text slightly. While defining these properties Karjoth assumed that a malicious host has captured the agent containing a set of encapsulated offers $O_1, O_2, ..., O_m$ where $m \leq n$ and O_m is the last host visited by the agent before being captured.

Forward Integrity: According to Yee [11], "None of the partial results collected prior to a malicious host can be modified without detection". If a mobile agent visits a number of hosts $S_1, S_2, ..., S_n$ and the first malicious host that it encounter is S_m where $1 \leq m \leq n-1$ then none of the partial results collected at hosts $S_i (i \leq m)$ can be undetectably modified by a malicious host.

Strong Forward Integrity: If a mobile agent visits a number of hosts $S_1, S_2, ..., S_n$ and the first malicious host it encounters is S_m then none of the encapsulated offer O_k where $k > m$ can be modified.

Insertion Resilience: Only hosts that are authorized to insert offers can add the offers.

Truncation Resilience: None of the encapsulated offer can be removed from the chain $O_1, O_2, ..., O_n$ without being detected.

Non-Repudiation: No hosts can deny the offers that it made to a visiting host.

3. PROPOSED IDEA

We have combined the idea of object sealing with pointer aliasing for mobile agent's computation results protection. When a mobile agent visits a host it performs some. computation on the host and as a result it produces some data. The results produced by a mobile agent at host can be represented by a dynamic data structure like a graph or link list or a PPCT (Planted Planer Cubic Tree). After representing the results in the form of a data structure we seal (encrypt) the head of Link List or graph or PPCT using

java's object sealing ability. Given any Serializable object one can create a SealedObject that encapsulates the original object in a serialized format and seals its serialized contents using a cryptographic algorithm like AES or DES to protect its confidentiality [5]. Dynamic data structures have a special property that we can traverse the whole structure from the head of structure e.g. all the nodes in a link list can be traverse by a single node called the head of the link list. So if we are able to represent the results of a mobile agent by a data structure like a Link list or graph or PPCT we can use java ability of sealing objects to protect the results of a mobile agent. Sealing or Encryption of objects in java is achieved by the existence of special purpose classes like SealedObject. Once the head is encrypted it is not available to anyone who does not posses the correct decryption key. One important step is to be able to represent the results with a dynamic data structure. Since mobile agent results can be converted into ASCII values and later these ASCII values can be converted to binary values. We use this idea of first converting the results into ASCII values and then from ASCII values to binary values. Once we have the binary values we can use any data structure to represent it by a dynamic data structure. For that purpose we first convert the results of the agent's computation into a form that can be represented by a dynamic graph structure. After this all we need is to seal (encrypt) the head of that graph. The cryptographic algorithm like DES or AES can be used to protect the confidentiality of this head object. The only overhead is the transfer of the secret key used for Symmetric Encryption/Decryption. Figure 1 & 2 illustrate our idea. As we can see at each host the head of the dynamic structure is sealed using symmetric encryption algorithm. We can recover the whole graph by just having the head of the graph. This is the property of any data structure that you can recover the whole structure just from head. But if the head is deleted or modified you can not traverse the graph. So the head of the graph must be protected from any modifications. For that reason we need to seal the head of the graph. One of the shortcomings of this idea is the exchange of the secret key used by each host for sealing the results representation head. For key transfer there are many efficient key transfer algorithms available like ElGamal or Diffie-Hellmann Key Exchange algorithm [15]. Here we don't consider how actually the key is being transferred rather assumed that home platform knows the secret key of each host.

3.1 Sealing an Object

The encrypted content can later be unsealed/decrypted and de-serialized, giving us the original object. The decryption process requires the corresponding algorithm with the correct decryption key. First of all the Cipher object must be initialized with the correct algorithm used for sealing object like AES/DES, key and the padding scheme, etc. After this we can apply this to seal the object. The original object that was sealed can be recovered in two different ways [5]. (1) By using the get Object method that takes a Cipher object. This method requires a fully initialized Cipher object, initialized with the exact same algorithm, key, padding scheme, etc., that were used to seal the object. This approach has the advantage that the party who unseals the sealed object does not require knowledge of the decryption key. (2) By using one of the get Object methods that take a Key object. In this approach, the getObject method creates a cipher object for the appropriate decryption algorithm and initializes it with the given decryption key and the algorithm parameters (if any) that were stored in the sealed object.

```
public SealedObject(Serializable object, Cipher
c);
public final Object getContent(Cipher c);
KeyGeneratorKeyGen=KeyGenerator.getInstance("AES");
SecretKey aesKey=keyGen.generateKey();
Cipher cipher= Cipher.getInstance("AES");
cipher.init (Cipher.ENCRPT_MODE, aesKey);
```

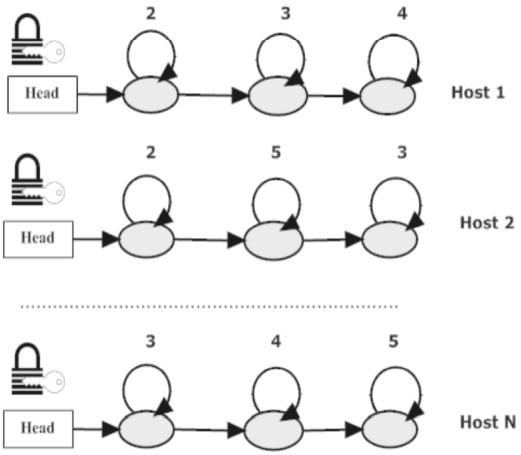

Figure 1: Computation Results Represented as a Graph.

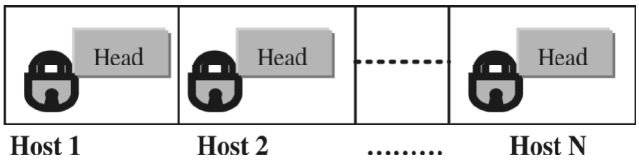

Figure 2: Seal Head Per Host.

3.2 Graph Representation

The data structure shown in the figure.3 is a very simple. It can be used to represent the results collected by a mobile agent at a remote host. This data structure has three fields. The first field self Reference is of type integer and it is used to represent the number of '0'. For every '1' in the result we draw a new node. The nextNode is a reference to the next node in the graph structure. The nextNode Null value represent that we have reached to the end of the input. The third field is the head and it used to keep track of the head of the graph. The figure 3 shows how such graph will look like.

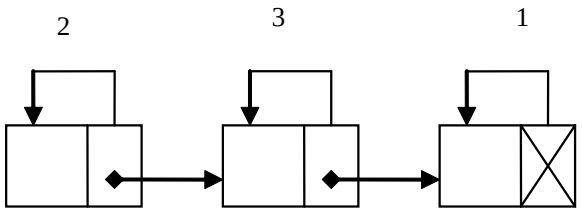

Figure 3: Graph Representation of the Results

The graph above encodes the binary value of 100100010

3.3 Mathematical Model

$R_i \rightarrow$ Result at host S_i

$G_i \rightarrow$ Graph representation of results at host S_i

$G_i = \Psi(R_i)$

$\Psi \rightarrow$ Function for converting results to graph representation

$\mathcal{H}_i \rightarrow$ head of the dynamic structure at S_i

$\mathcal{E}_i \rightarrow$ Encrypted Head after applying SEAL Function at host S_i

$\mathcal{E}_i = SEAL\ (\mathcal{H}_i)$

$\Omega_i = \Omega_{i-1} \cup \{\mathcal{E}_i\}$

$\mathcal{H}_i = Decrypt\ (\mathcal{E}_i)$

3.4 Security Model

In order to test the proposed idea we have implemented a simple Ticket Booking Agent. The primary goal is to make sure that the partial results collected by mobile agent are in contact and any modification or tampering can be detected by the originator. The owner of the agent wants to take a flight say from Beijing to Hong Kong on next weekend. The owner of the agent wants to buy the cheapest ticket. The owner sends his mobile agent to various airline agencies which are providing the ticket service. The owner sends his agent with his desired preferences to various airlines servers to check the availability of the ticket, the price as well as other relevant information e.g. departure time, arrival time, flight is direct or number of stops etc. The agent queries the servers and return to the originator with the results of its computation. In order to get the best suitable price of tickets for its owner the agent must keep the prices/data collected at all the hosts. When a mobile agent collects price from one host say S_i and move to the next host S_{i+1}, the price of the former host S_i must be kept secret from the later sever otherwise the later host may take advantage e.g. offering a false offer or modifying previously stored offers. The host S_{i+1} may offer a price that is not the actual price offered by it or in worst case it can modify the price collected prior to visiting S_{i+1} in order to get unfair advantage. Mobile agent must be protected against such attacks. Our main objectives are confidentiality and integrity of the data carried by mobile agent. Confidentiality here means to reveal cleartext only at trusted hosts. Integrity means the agent must be protected such that it can collect new data set at each host they visit but also any tampering with the pre-existing collected data set must be detected by any trusted host.

4. IMPLEMENTATION AND PERFORMANCE EVALUATION

We have implemented the proposed idea using IBM Aglets frameworks. For cryptographic primitives we have used the bouncy castle provider. Our experimental results were performed in a window based environment with 3 PCs each having Pentium IV 3.01GHZ processor, 512MB RAM for agent home and remote hosts. There are several factors in performance evaluation of the proposed scheme such as the size of the dynamic graph, how fast the underlying symmetric encryption algorithm runs and how fast object serialization is performed. Object serialization gives us the ability to read or write a whole object to or from a byte stream. As we can see that the object serialization time is very small even for large objects. The time for symmetric encryption for both AES and DES are also listed. The table 1 below suggests that symmetric encryption of dynamic structure head does not introduce very serious performance issue.

Table 1
Performance with AES/DES Symmetric Encryption

Object Size	*Serialization*	*DES Sealing*	*AES Sealing*		
			128bits	*192bits*	*256bits*
10 Bytes	0ms	40ms	8ms	10ms	14ms
100Bytes	0ms	45ms	21ms	23ms	26ms
10KB	5ms	120ms	48ms	50ms	54ms
100KB	10ms	900ms	100m	105ms	120ms

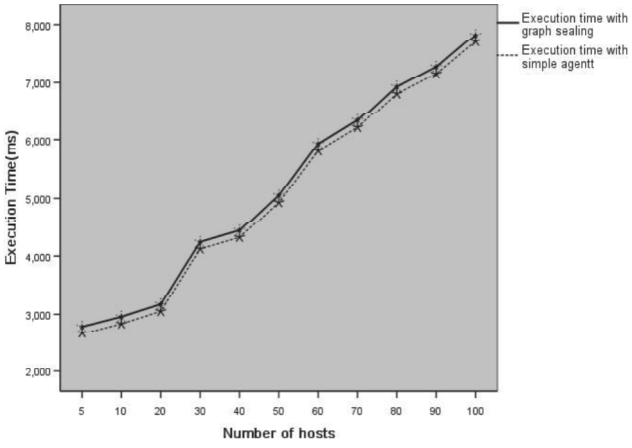

Figure 4: Comparison of Execution Time of Graph Head Sealing with Simple Agent Execution

We have also compared execution of graph head sealing with the simple agent execution. This is shown in figure 4 above. Although graph head sealing put some extra execution burden on the executing platform as we can see this burden is not very large. At the cost of some extra burden we are providing more security to the agent. We believe that this

overhead is not a serious concern when compared with the critical nature of application in which mobile agents are used.

5. CONCLUSION

Mobile agent technology offers many advantages but their use is limited mainly because of the security problems associated with them. Security in mobile agent system can be divided into two types (1) security of host from malicious mobile agent (2) security of mobile agent from malicious host. The second type of security can further be divided into two subtypes (1) Security of static code (2) Security of dynamic data state. The security of the dynamic data state is still an open problem of research. In this paper we have proposed a new method for protection of mobile agent's computation results. First we represent the computation results of a mobile agent with a dynamic data structure and then we used object sealing to protect the computation results of the mobile agent. A symmetric encryption algorithm like AES or DES can be used to seal the objects. Later we can unseal the object provided we have the secret key. The ability to represent the results an agent computation in a graph shape which is hard to analyze for an attacker thus providing agent owner with enough protection. Our experimental results suggest that the method can be implemented efficiently by mobile agent programmer. At the cost of some extra execution time we can make the computation of the agent more secure and the owner of the agent can be more certain about the results collected by the agent.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (60703011), the Chinese national 863 high-tech projects (2007AA01Z458, the Foundation for the Author of National Excellent Doctoral Dissertation of China (FANEDD-200238), the Scientific Research Foundation of Harbin Institute of Technology (HIT.2003.52), the Research Fund for the Doctoral Program of Higher Education (RFDP): 2007 021307, and the Program for New Century Excellent Talents in University (NCET-04-0330).The author of the paper is also grateful to COMSATS Institute of Information Technology(CIIT), Pakistan for their generous support.

REFERENCES

[1] Dag Johansen, "Mobile agent Applicability", *In Proceedings of the Second Workshop on Mobile Agents* 1998, Berlin, Springer-Verlag LNCS; Vol. 1477, ISBN 3-540-64959-X, (1998), pp. 9-11 September, 1998.

[2] Pattie Maes, Robert H.Guttman and Alexandros G. Moukas, "Agents That Buy and Sell", *Communication of the ACM*, Vol, 42, No. 3, pp. 81-91, March 1999.

[3] Stavros Papastavrou, George Samaras and Evaggelia Pitoura, "Mobile Agent for WWW Distributed Database Access", In Proceedings of IEEE International Conference on Data Engineering (*ICDE99*), 1999.

[4] Gian Pietro Picco and Mario Baldi, "Evaluating Tradeoffs of Mobile Code Design Paradigms in Network Management Applications", In Proceedings of 20th *ICSE98*, Kyoto, Japan IEEE CS Press, 1998.

[5] Gong, L., and Schemers, "Signing, Sealing, and Guarding Java Objects". *Mobile Agents and Security* LNCS, Vol. 1419. pp. 206-216, 1998.

[6] O. Esparza, M. Soriano, J. L. Munoz, and J. Forn¡äe, "Host Revocation Authority: A Way of Protecting Mobile Agents from Malicious Hosts", *International Conference on Web Engineering* (*ICWE 2003*), LNCS. Springer-Verlag, 2003.

[7] Fritz Hohl, "Time Limited Black box Security: Protecting Mobile Agents from Malicious Hosts" *Mobile Agents and Security*, LNCS, Vol. 1419, Springer-Verlag London, UK 1998.

[8] O. Esparza, M. Fernandez, M. Soriano, J. L. Munoz, and J. Forne, "Mobile Agent Watermarking and Fingerprinting: Tracing Malicious Hosts", In (*DEXA 2003*), Vol. 2736 of LNCS, Springer-Verlag, 2003.

[9] O. Esparza, M. Soriano, etc. "Detecting and Proving Manipulation attacks in Mobile Agent System". In First International Workshop, *MATA 2004*, Florianopolis, Brazil, In Proceedings, Vol. 3284 of LNCS. Springer Verlag, 2004.

[10] G. Karjoth, N. Asokan, and C. Gulcu, "Protecting the Computation Results of Free-Roaming Agents", *In Proceedings of 2nd International Workshop on Mobile Agents*, Vol. 1477 of LNCS, p. 195–207. Springer-Verlag, 1998.

[11] B. Yee. "A sanctuary for Mobile Agents", In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, Vol. 1603 of LNCS, p. 261–273, Springer-Verlag, 1999.

[12] S. Loureiro, "Mobile Code Protection", *PhD Thesis*, ENST Paris / Institut Eurecom, 2001.

[13] His-Chung Lin, Sung-Ming Yen, and Her-Shu Chen. "Protection of Mobile Agent Data Collection by Using Ring Signature", In Proceedings of 2004 IEEE *ICNSC 2004* Taipei, Taiwan, March 21-23, 2004.

[14] Ronal L.Rivest, Adi.Shamir, and Yael. Tauman "How to leak a secret", Advances in Cryptology-*ASIACRYPT 2001*, LNCS, Vol. 2248. Springer-Verlag, pp. 552-565, 2001.

[15] Rolf Oppliger, *Contemporary Cryptography*, Artech House Computer Security, April 30, 2005.

[16] Anirban Majumdar and Clark Thomborson, "Securing Mobile Agents Control Flow using Opaque Predicates", *Intelligence and Security Informatics* LNCS, Springer Berlin / Heidelberg, Vol. 3495/2005.

[17] Yusuke Sakabe, Masakazu Soshi and Atsuko Miyaji, "Java Obfuscation with a Theoretical Basis for Building

Secure Mobile Agents", Communication and Multimedia Security (*CMS 2003*) LNCS, Springer Berlin / Heidelberg, Vol. 2828/2003.

[18] Vigna J. Vigna J., "Cryptographic Traces for Mobile Agents", *Mobile Agent and Security*, LNCS 1419, 1998, Springer, pp. 137-153.

[19] V. Roth. "Mutual Protection of Co-operating Agents" *Secure Internet programming: Security Issues for Mobile and Distributed Objects*, LNCS 1603, New York, NY, USA: Springer-Verlag, pp. 275-285, 1999.

[20] F. Hohl, "A Framework to Protect Mobile Agents by Using Reference States". In Proceedings of the 20th International Conference on Distributed Computing Systems (*ICDCS 2000*), 10-13 April 2000 Taipei, TAIWAN.

[21] Y. Villate, A. Illarramendi, and E. Pitoura, "Data Lockers: Mobile-Agent Based Middleware for the Security and Availability of Roaming Users Data". In the 7th International Conference on Cooperative Information Systems (*CoopIS 2000*), Eilat, Israel, September 6-8, 2000, LNCS 1901, pp 275-286, Springer, 2000.

[22] V. Roth, "On the Robustness of Some Cryptographic Protocols for Mobile Agent Protection", In Proc of MA 2001**,** Vol. 2240 of LNCS. Springer Verlag, December 2001.

[23] N. M. Karnik and A. R. Tripathi, "Security in the Ajanta Mobile Agent System," *Technical Report* TR-5-99, University of Minnesota, Minneapolis, MN 55455, U. S. A., May 1999.

[24] A. Corradi, R. Montanari, and C. Stefanelli, "Mobile Agents Protection in the Internet Environment", In *COMPSAC '99*, pp. 80-85, 1999.

[25] J. T. McDonald, Alec Yasinsac, W. C. Thompson. "Mobile Agent Data Integrity Using Multi-agent Architecture", In Proceedings of the International Workshop on Security in Parallel and Distributed Systems (*PDCS 2004*), San Francisco, CA, 14-17 September 2004.