Protecting Free-Roaming Mobile Agent against Multiple Colluded Truncation Attacks

S.Venkatesan
Department of CSE,
Anna University
Chennai-600025, India
venkalt_s@yahoo.co.in

C.Chellappan
Professor, Department of CSE
Anna University,
Chennai-600025, India
drcc@annauniv.edu

ABSTRACT

Mobile agent environment is one of the emerging technology to reduce the network traffic. Various security issues are identified and protected. Now the new security issue in this environment is the multiple colluded truncation attacks in the Free-roaming mobile agents. This paper proposes the identity verification mechanism to protect the multiple colluded truncation attacks The identification is verified from the beginning dummy offer of the creator of the agent.

Categories and Subject Descriptors

C.2.0 **[General]** : Security and protection I.2.11 **[Distributed Artificial Intelligence]**: Intelligent Agent D.2.0 **[General]**: Protection Mechanism

General Terms

Security, Algorithms,Theory

Keywords

Free-Roaming mobile agent, colluded attacks, Truncation attacks, encapsulated offer

1.INTRODUCTION

Mobile agents are software programs to perform computation in various hosts and then bring the results to the owner of the agent. Roaming agents are moving from one node to another node by the respective statements of the owner but the free-roaming agents are moving from one host to another host without any respective migration paths by the owner. i.e., depends upon the requirements and the current conditions, the current host will select the next host for agent. In the free-roaming mobile agent environment, security is the main issue. That the malicious host may modify, delete or insert (malicious) data in the results of the mobile agents, which they collected from the previous hosts. For this types of attacks the various methods of prevention protocols are proposed and solved.

MOBIMEDIA 2007, August 27-29, Nafpaktos, Greece

DOI 10.4108/ICST.MOBIMEDIA2007.1759

Nevertheless another one type of attack is raised is the multiple colluded attack. i.e, more than one host colluded together to discard the single data or the stream of data between the hosts.The general cryptographic mechanism is used for the protocols.

2.PREVIOUS WORKS

Yee[1] proposed the Partial Result Authentication Code(PRAC) to protect mobile agents results. Here the agent and its originator maintain a list of secret keys or a key generating function. The agent uses a key to encapsulate the collected offer and then destroys the key. However, a malicious host may keep the key or the key generating function. When the agent revisits the host or visits another host conspiring with it, a previous offer or series of offers would be modifed, without being detected by the originator and also the carrying of list of keys by the free roaming agents for all the system is not possible.

Karjoth et al.[2] extends the above schemes for the efficient security purposes. Each host generates a signing key for its successor and certifies the corresponding verification key. Using the received signature/verification key pair, a host signs its partial result and certifies a new verification key of the next host. This technique will avoid the modification attack in above scheme, but not a two-colluder attack. In this attack two visited hosts can collude to discard the partial results collected between their respective visits. Here it uses digital signatures and hash functions to protect a chain relation

Karnik et al.[3] This protocol uses an encrypted checksum to build a backward chain relation to link an agents previous result with the agents data generated at the currently visited host. It guarantees that only new data can be added to the results of the agent collected and no data can be deleted from them. It does not support two-colluder attacks. It is the compact method of the above scheme.

Corradi et al.[4] Here the protocol uses the backward and forward chaining. At each host, the protocol runs a hash function to compute a cryptographic proof of a result from the previous host, a result generated at the current host, and the identity of next hop. Like other protocols, this protocol cannot defend two-colluder truncation attacks.

Cheng et al.[5] This protocol provide the co-signing mechanism to defend the two-colluder truncation attacks. Here a preceding host co-signs a result generated at the current host. Attackers need their preceding non-attackers to co-sign fake offers when they launch two-colluder truncation attacks, and then their actions can be detected. Here also the publicly verifiable forward integrity propriety, this protocol generates a pair of one time secret private and public keys at each host for its successor.

Yao et al.[6] Songsiri[7] Here the Trusted Third Party is used to store the collected offers. To defend a stemming attack, a special case of two-colluder truncation attacks, the protocol needs to be modified and requires a two-way authentication.

Zhou et al.[8][9] In this protocol, the one time signature key pair is generated by each host rather than the preceding host. The protocol also defends the truncation attack with a special loop. It requires its preceding host to co-sign encrypted data. The co-signers cannot check the encrypted data, malicious hosts may ask their preceding hosts to sign encrypted documents and use these signatures against the co-signers later. This protocol cannot defend multiple-colluder truncation attacks.

Darren et al.[11] This protocol will defend the two-colluder truncation attack with the help of the one hop backward and two-hop forward chaining method. And also it will defend the multiple colluder truncation attacks. It will avoid the fake stem attack and then the interleaving attack. It has the drawback that the two-colluders in the adjacent means it will not defend the attack.

Our new protocol addresses all the issues found in the previously available protocols, especially solutions to the multiple colluder truncation attacks. This paper is discussed as follows. In segment 3, we describe the common mobile agent security properties discussed in the previous papers. In segment 4, we analyze the new protocol and its mobile agent properties. In segment 5 &6, we analyze the protection for various attacks and at last in the segment 7, we conclude the specialty of this protocol**.**

3.NOTATIONS AND SECURITY PROPERTIES

We use the similar notations used in the other schemes [2][5][9][11].

Table. 1.Notations

Notations	**Descriptions**
$S_0{=}S_{n+1}$	Originator or Creator or Owner.
o_0	Offer from S_0 to identify the agent instance on return.
o_i	Offer from S_i.
S_1,S_2… S_n	Hosts and also its identity
r_i	Random number generated by S_i.
Pb_i , Pr_i	Public and Private key of the host S_i.
tPb_i , tPr_i	Temporary Public and Private key of the host S_i.
$Enc_{Pbi}(m)$	Message *m* is encrypted with the public key Pb_i of S_i.
$Sig_{Pri}(m)$	Signature of S_i on message *m* with its private key Pr_i.
$H(m)$	Hash function.
O_i , $1<=i<=n$	Encapsulated offer.
C_i , $1<=i<=n$	Cryptographically encrypted offer of S_i.

Assume that the agent has a chain of encapsulated offers $O_0, O_1, \ldots O_i, \ldots O_m$. The following mobile agent security properties are based on assumption [2][5][11].

Data confidentiality: Each offer o_i is encrypted by S_0's public key Pb_0. Only the originator can decrypt the offer O_i.

Non-repudiability: Each offer o_i is signed by S_i as $Sig_{Pri}(o_i)$. S_i cannot deny its offer o_i after S_0 receives the offer and verifies the signature.

Forward privacy: None of the identities of the creators of offer o_i can be extracted by anyone except the originator S_0.

Strong forward integrity: Assume an attacker S_m holds encapsulation offers $O_0, O_1,\ldots, O_{i-1}, O_i,\ldots, O_{m-1}$, and modifies or replaces O_{i-1} with O_{i-1}'. O_{i-1} is one of the components in the checksum $h_i{=}H(O_{i-1}, r_i, S_{i+1}, S_{i+2})$ in the encapsulated offer O_i. Since O_i is intact, the chain relation $h_i{=}H(O_{i-1}, r_i, S_{i+1}, S_{i+2})$) must be hold true, i.e. $H(O_{i-1}', r_i, S_{i+1}, S_{i+2}){=} H(O_{i-1}, r_i, S_{i+1}, S_{i+2})$This violates the assumption that the hash function H is collision-free. It is impossible for an attacker to modify or replace any offer without changing the next encapsulated offer if a collision free hash function is used in the protocol.[11].

Publicly verifiable forward integrity: Any one can verify the encapsulated offers O_i by checking whether the chain is valid at O_i.

Insertion defense: No offer can be inserted at i unless explicitly allowed; i.e, by the multiple hosts colluded with one another..

Truncation defense: Assume an attacker S_m truncates all encapsulated offers after O_{i+1}, and then appends its own offer O_m. The new chain of encapsulated offers is now $O_0, O_1,\ldots, O_i, O_{i+1}, O_m$. Since O_i is intact, the chain relation $h_i{=}H(O_{i-1}, r_i, S_{i+1}, S_{i+2})$ must be hold true, i.e. $h_i{=}H(O_{i-1}, r_i, S_{i+1}, S_{i+2}) = H(O_{i-1}, r_i, S_{i+1}, S_m)$.This violates the assumption that the hash function H is collision-free. [11].

4. PROPOSED PROTOCOL

Our protocol builds a chain relation to all the previous encapsulated offers and forwards it to the next host. The encapsulated offers consist of the hash function and then the encrypted offer which consist of the generated offer and random number by the visited hosts $S_0,S_1,S_2,S_3,S_4\ldots$. The protocol is discussed in below.

4.1 Agent at the Creator (S_0)

The creator S_0 creates and starts the agent with the dummy offer o_0 and random number r_0. And then signs the offer $Sig_{pr0}(o_0)$ for Non- repudiability and encrypt those data Enc_{Pb0} ($Sig_{pr0}(o_0)$, r_0) with the help of the public key Pb_0 and selects the next host S_1 . Also it generated the temporary public and private key (tPb_i , tPr_i) for sign in the identity of the next host $Sig_{tpr0}(S_1)$.

S_0: Generate offer o_0
Compute $C_0 = Enc_{Pb0}(Sig_{pr0}(o_0), r_0)$
Generate tPr_0 ,tPb_0
Decide next host S_1
$h_0 = H(Sig_{pr0}(S_0), Sig_{tpr0}(S_1), tPb_0, r_0)$
$O_0 = SigPr_0(C_0, h_0)$
$S_0 \rightarrow S_1$: O_0

At last S_0 signs the final encapsulated offer with the its private key Pr_0 and sends its offer to the next selected host S_1.

4.2 Agent at S_1

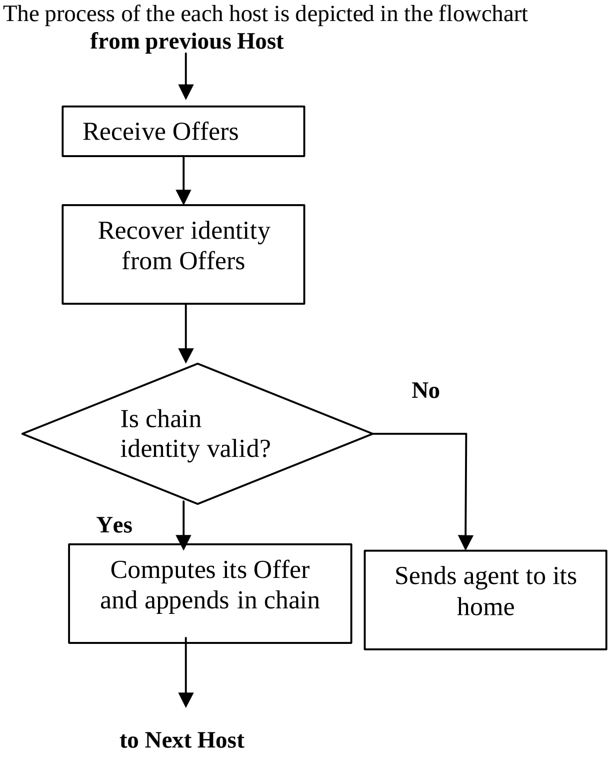
Agent migrates from S_0 to S_1 and carries the encapsulated offer O_0. After receiving the encapsulated offer, S_1 can verify the encapsulated offer for the integrity that is to check whether he is the recipient or not and also the offer from the previous host is reliable. This is by the public key Pb_0 of the sender.

S_1: Receive O_0
Recover C_0, h_0 by Pb_0
Ver($Sig_{pr0}(S_0)$,$Sig_{tpr0}(S_1)$,tPb_0) recover S_1 ,S_0

Where S_1(identity) is recovered by tPb_0 and the S_0(identity) is recovered by Pb_0 and verify the identity sequence from the beginning.

Generate offer o_1
Compute $C_1 = Enc_{Pb0}(Sig_{pr0}(o_1), r_1)$
Generate tPr_1 ,tPb_1
Decide next host S_2
$h_1 = H(Sig_{pr0}(S_0), Sig_{tpr0}(S_1),$
$Sig_{tpr1}(S_2, tPb_0), tPb_1, r_0)$
$O_1 = Sig_{tPr1}(C_1, h_1$
$S_1 \rightarrow S_2$: O_0, O_1, tPb_1

The process of the each host is depicted in the flowchart



After that it will generates its offer o_1 and random number r_1 and then computes encrypted C_1 with the public key of creator Pb_0.

4.3 Agent at S_2

After receiving the encapsulated offers, host S_2 recovers the encapsulated offers O_1 by the help of the tPb_1 and O_0 by the help of Pb_0. After that the identity of the two encapsulated offers O_0 ,O_1 are recovered. The recovered identities of O_0 *is* S_0 , S_1 and the identities of O_1 is S_0 , S_1 ,S_2 . Now the last two identities of the O_0 and first two identities of O_1 is compared O_0 (S_0 , S_1) = O_1 (S_0 ,S_1). If both are same the next process will proceeds otherwise it sends the agent back to its home.

S_2: Receive O_0, O_1, tPb_1
Recover C_1, h_1 by tPb_1
Recover C_0, h_0 by Pb_0
Ver($Sig_{pr0}(S_0)$, $Sig_{tpr0}(S_1)$,$Sig_{tpr1}(S_2, tPb_0)$,
tPb_1 ,r_0)) recover S_0 ,S_1, S_2
Ver($Sig_{pr0}(S_0)$,$Sig_{tpr0}(S_1)$,tPb_0)recoverS_0 ,S_1

After that it will generates its offer o_1 and random number r_1 and then computes encrypted C_2 with the public key of creator Pb_0.

Generate offer o_2
Compute $C_2 = Enc_{Pb0}(Sig_{pr2}(o_2), r_2)$
Generate tPr_2 ,tPb_2
Decide next host S_3
$h_2 = H(Sig_{tpr0}(S_1), Sig_{tpr1}(S_2, tPb_0),$
$Sig_{tpr2}(S_3, tPb_1), tPb_2, r_3)$
$O_2 = Sig_{tPr2}(C_2, h_2)$
$S_1 \rightarrow S_2$: O_0, O_1, O_2, tPb_2

4.4 Agent at S_i

S_i receives the all previous encapsulated offers and verify the identities.

S_i: Receive $O_0, O_1, , O_2, , O_3, \ldots O_{i-1}, tPb_{i-1}$
Recover C_{i-1} ,h_{i-1} by tPb_{i-1}

⋮

Recover C_0, h_0 by Pb_0
Ver($Sig_{tpri-3}(S_{i-2}, tPb_{i-4})$,$Sig_{tpri-2}(S_{i-1}, tPb_{i-3})$,
$Sig_{tpri-1}(S_i, tPb_{i-2})$, tPb_{i-1} ,r_0))
recover S_{i-2} ,S_{i-1}, S_i

⋮

Ver($Sig_{pr0}(S_0)$, $Sig_{tpr0}(S_1)$,$Sig_{tpr1}(S_2, tPb_0)$,
tPb_1 ,r_0)) recover S_0 ,S_1, S_2
Ver($Sig_{pr0}(S_0)$,$Sig_{tpr0}(S_1)$,tPb_0)recoverS_0 ,S_1

It will recover the identities in the following format and compares the last two and first two identities. If both are same, the host decides no attack on that data else attack is identified and then the host sends the agent to its home. It will match from O_0 *to* O_{i-1} for the reliability of the chain.

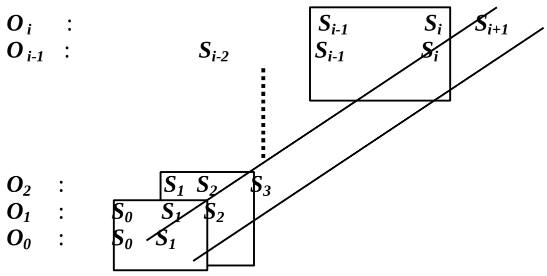


Fig.1. Comparison Representation

Fig.1 represents the comparison function done in the each visited host. Each host will have the three identities except the host S_0 . The Host identity represented in the diagonal is the next visiting host, which selected by the current host. Now the identity of the hosts are compared one by one with the help of first two identity and then the last two identity of the current and the previous host. Which is contained in the hash function. After the verification process, the host S_i generates its offer and made the hash function and forwards the agent to its next host $S_{i+1.}$

Generate offer o_i
Compute $C_i = Enc_{Pb0}(Sig_{pri}(o_i), r_i)$
Generate tPr_i ,tPb_i
Decide next host S_{i+1}
$h_i = H(Sig_{tpri-2}(S_{i-1}, tPb_{i-3}), Sig_{tpri-1}(S_i, tPb_{i-2}), Sig_{tpri}(S_{i+1}, tPb_{i-1}), tPb_i, r_i)$
$O_i = Sig_{tPri}(C_i, h_i)$
$S_i \rightarrow S_{i+1}$: $O_0, O_1, , O_2, , O_3, \ldots O_i, tPb_i$

4.5 Agent return to home S_0

At last agent returns to its home and give the collected offers to its owner. The owner will recover all the encapsulated offers $O_0, O_1, , O_2, , O_3, \ldots O_{i-1}, O_i , O_{i+1}, \ldots O_n$ with the help of the public key. where O_n is recovered by the tPb_n and then O_{n-1} is recovered by the help of the tPb_{n-1} which is already available in the hast function of O_n. After that the offer $o_0, o_1, , o_2, , o_3, \ldots o_{i-1}, o_i , o_{i+1}, \ldots o_n$ are recovered by the public key of the originator $Pb_{0.}$

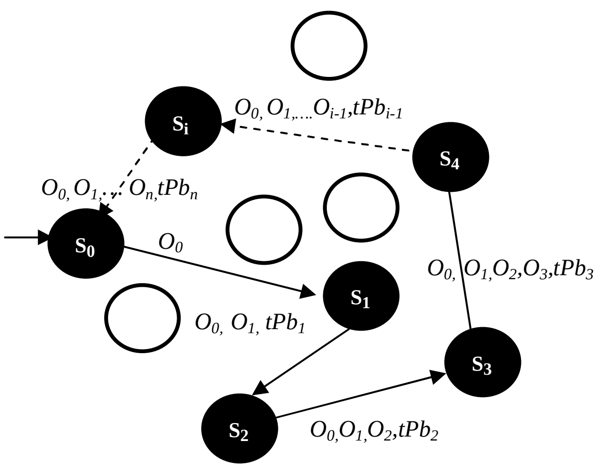


Fig.2. Agent Migration

Fig.2. shows the agent migration from the owner to the other host to collect the offer and return back to the owner.

4.6 Security properties

Data Confidentiality: Only the originator can decrypt the offer o_i because it is encrypted by the public key Pb_0 of the host S_0

Non-repudiability: Each offer o_i is signed by its host S_i as $Sig_{pri}(o_i)$. So S_i cannot deny its offer o_i after S_0 receives.

Forward privacy: Even though the identities of the creators of offer o_i extracted by anyone but they are not able read or update the offer except the originator S_0. Because of the encryption mechanism $Enc_{Pb0}(Sig_{pri}(o_i), r_i)$.

Strong Forward Integrity: None of the encapsulated offers can be modified because the each and every encapsulated offers includes the chain relation S_0, S_1, S_2 and S_1, S_2, S_3 . If any results get modified, the chain relation will not followed and then the current host identifies the attack and sends the agent to its home. Random number r_i also generated by the host for the purpose of the integrity. It is available in both the hash function and the encrypted function. The random number r_i in the both will be checked for the integrity. There may be chance to attack the part of the encapsulated offer. $O_i = Sig_{tPri}(C_i, h_i)$. There is the possibility to change the C_i and make the new offer O_i '= $Sig_{tPr2}(C_1', h_i)$. Which will be identified during the decryption, with the help of r_i

Insertion Resilience: No offer can be inserted in the middle and also no more than one offers can be inserted into the chain (**Revisiting attack[8]**) by the host, because the identities will repeat more than once in the chain. For example S_0, S_1, S_1 and S_1, S_1, S_2, now the host S_2 compares the chain and identifies the dual insertion by the single host with its identity. Same way the modification $H(Sig_{tpri-2}(S_{i-1}, tPb_{i-3}), Sig_{tpri-1}(S_i', tPb_{i-2}), Sig_{tpri}(S_{i+1}, tPb_{i-1}), tPb_i, r_i) = H(Sig_{tpri-2}(S_{i-1}, tPb_{i-3}), Sig_{tpri-1}(S_i, tPb_{i-2}), Sig_{tpri}(S_{i+1}, tPb_{i-1}), tPb_i, r_i)$ also identified because it violates the hash function.

Truncation Resilience: No offer o_i or sequence of offers from o_i , $o_{i+1}, \ldots o_m$ can be truncated and then the append the new offer. Because the comparison function is ended with the owner encapsulated offer O_0, so it can be easily identified by the following function.

$Ver(Sig_{tpri-3}(S_{i-2}, tPb_{i-4}), Sig_{tpri-2}(S_{i-1}, tPb_{i-3}), Sig_{tpri-1}(S_i, tPb_{i-2}), tPb_{i-1}, r_0))$
recover S_{i-2}, S_{i-1}, S_i

⋮

$Ver(Sig_{pr0}(S_0), Sig_{tpr0}(S_1), Sig_{tpr1}(S_2, tPb_0), tPb_1, r_0))$ recover S_0, S_1, S_2
$Ver(Sig_{pr0}(S_0), Sig_{tpr0}(S_1), tPb_0)$recover$S_0, S_1$

Public Verifiable Forward Integrity: Any host can recover the encapsulated offer $O_0 \ldots\ldots O_i$ to verify the h_i by the help of the temporary public key generated by the previous host**.** The offer $O_i = Sig_{tPri}(C_i, h_i)$ is recovered by the help of tPb_i which is received with the chain $O_0, O_1, , O_2, , O_3, \ldots O_{i-1}, O_i$, tPb_i. Then the O_i contains the temporary public key for O_{i-1}. Likewise it will recover the chain.

5. MULTIPLE COLLUDED TRUNCATION ATTACKS

Various mechanism is provided to defend the truncation attacks. Darren et al.[11], pointed to protect the two-colluder truncation attacks which will also avoid the attacks by the single host[2]. And also the they provide the protection for Multiple (three or more) colluded attacks while they are not in the adjacent place. But our proposed protocol, can protect the multiple colluded attackers when they are in the adjacent places also because the chain relation is maintained in the form of the previous, current and then the next host identities. In the chain $S_0,\ldots,S_{i-1}$, S_i, $S_{i+1},\ldots$, S_x, $S_{x+1},\ldots$, S_{m-1} , S_m if S_i, S_{i+1}, and S_m are colluded to attack the sequence of offers from $O_{i,}$, $O_{i+1,}$ …… O_m. Now the offers will be $O_{0,}$ $O_{1,}$, $O_{2,}$, $O_{3,}$ … $O_{i-1,}$ O_i‘,O_{i+1}‘, … O_m‘. Now this chain is forwarded to the host S_{m+1}. This host will recover this offers and verify the identifies as represented in the Fig.1.

6.GROWING A FAKE STEM ATTACK & INTERLEAVING ATTACK

Attacker truncates the offers O_i and appends fake offers is referred as growing a fake stem attack[2]. Attackers breaks the chain and provide the space for inserting the fake offers is interleaving attack[10]. In this protocol, there is no opportunities for this type of attack because each will maintain the identity, by the identity the fake stem and Interleaving attack will easily identified as described in the Insertion resilience.

7.CONCLUSION

Our protocol uses the identity chain relation with the collected offer. So the free roaming agent will be protected from the two-colluded or multiple-colluded attacks. It can also avoid the Growing of fake stem attack[2], Revisiting attack[8] and Interleaving attack[10]. In this method we can avoid the concentration of the previous host while we reach the next host in the previous protocol[11]. This method will avoid the network traffic and also very much useful in the Distributed environments (which will be used to help the free roaming mobile agent in the E-Commerce & E-learning environments to collect the results.

8.ACKNOWLEDGMENT

This Work is supported by the NTRO, Government of India. NTRO provides the fund for collaborative project “smart and secure environment” and this paper is modeled for this project. Authors would like to thanks the project coordinators and the NTRO members.

9. REFERENCES

[1] B.S. Yee. “A sanctuary for mobile agents”. *Technical Report CS97-537*, UC San Diego, Department of Computer Science and Engineering, April 1997.

[2] G. Karjoth, N. Asokan, and C. Gülcü. “Protecting the computation results of freeroaming agents”. In *Proc. Second International Workshop on Mobile Agents (MA’98), K. Rothermel and F. Hohl, editors, LNCS 1477*, pp.195 - 207, Springer-Verlag, 1998.

[3] 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.

[4] A. Corradi, R. Montanari, and C. Stefanelli. “Mobile agents Protection in the Internet Environment”. In *The 23rd Annual International Computer Software and Applications Conference (COMPSAC ’99)*, pages pp. 80–85, 1999.

[5] J. Cheng and V. Wei. “Defenses against the truncation of computation results of free-roaming agents”. In *4th International Conference on Information and Communications Security*, volume LNCS 2513, pages 1– 12, December 2002.

[6] M. Yao, E. Foo, E. P. Dawson and K. Peng. “An Improved Forward Integrity Protocol for Mobile Agents”. In *proceedings of 4th International Workshop on Information Security Applications (WISA 2003)*, volume 2908 of Lecture Notes in Computer Science, pages 272-- 285. Springer-Verlag, 2004. ISBN: 3-540-20827-5.

[7] Suphithat Songsiri. "A New Approach for Computation Result Protection in the Mobile Agent Paradigm". In *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)*, 27-30 June 2005, Murcia, Cartagena, Spain.

[8] J. Zhou, J. Onieva, and J. Lopez. "*Analysis of a Free Roaming Agent Result-Truncation DefenseScheme*". In *Proceedings of 2004 IEEE Conference on Electronic Commerce*, pages 221--226, San Diego, USA, July 2004, IEEE Computer Society Press.

[9] J. Zhou, J. Onieva, and J. Lopez. "Protecting Free Roaming Agents against Result-Truncation Attack". In *Proceedings of 60th IEEE Vehicular Technology Conference*, pages 3271--3274, Los Angles, USA, September 2004, IEEE Vehicular Technology Society Press.

[10] V. Roth. “On the robustness of some cryptographic protocols for mobile agent protection.” In *Proceedings of the 5th International Conference on Mobile Agents (MA 2001)*, volume 2240 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 2001.

[11] D.Xu, L.Harn, M.Narasimhan, J.Luo. “An Improved Free-Roaming Mobile Agent Security Protocol against Colluded Truncation Attacks.” *In Proceedings of the 30th Annual international Computer Software and Applications Conference(COMPSAC,06)*, Volume 2, Page(s):309 – 314 ,Chigago,Sept. 2006,IEEE Computer Society Press.