Secure Route Structures for The Fast Dispatch of Large-Scale Mobile Agents

Yan Wang[1], Chi-Hung Chi[2], and Tieyan Li[3]

[1] Department of Computing, Division of Information and Communication Sciences, Macquarie University, NSW 2109, Australia
[2] Department of Computer Science National University of Singapore 3 Science Drive 2, Singapore 117543
{ywang, chich}@comp.nus.edu.sg
[3] Infocomm Security Department, Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613
litieyan@i2r.a-star.edu.sg

Abstract. For the application of large-scale mobile agents in a distributed environment, where a large number of computers are connected together to enable the large-scale sharing of data and computing resources, security and efficiency are of great concern. In this paper, we present secure route structures and corresponding protocols for mobile agents dispatched in binary to protect the dispatch route information of agents. The binary dispatch model is simple but efficient with a dispatch complexity of $O(log_2n)$. The secure route structures adopt the combination of public-key encryption and signature schemes and expose minimal route information to hosts. The nested structure can help to detect attacks as early as possible.

1 Inroduction

Mobile agents are computational entities that are autonomous, mobile and flexible that can facilitate parallel processing. Very often, a mobile agent acts on behalf of its owner to migrate through the distributed network, completes the specified tasks and returns results back to the owner [1–3].

The use of mobile agents in a distributed environment is gaining increasing attention. For example, in a national scale Grid environment [4–8], a large number of computers are loosely coupled together to enable the large-scale sharing of data and computing resources, where agents, especially mobile agents, are naturally the tools for monitoring, managing hosts and deploying jobs. Typically, a mobile agent can carry a computational job and execute at a host after being dispatched there. Likewise, in a mobile agent based E-commerce environment [9], mobile agents can be dispatched as the request of a consumer (end-user) to visit e-shops for asking offers for a specified product, evaluating these offers and negotiating with shops. In the above-mentioned environments, an efficient dispatch model is important and initial dispatch route information should be protected against potential malicious hosts. Otherwise, some attacks may be easily mounted breaking the deployment of agents. So, if the owner needs to dispatch large-scale mobile agents, the security and efficiency are of great concern [10, 11].

Tamper-poof devices [12] and secure coprocessors [13] are hardware-based mechanisms that can be used for protecting mobile agents and hosts. Software-based approaches involve more work such as using Hiding Encrypted Function (HEF) [14], using proxy signature [15] and using delegation certificate [16]. However, these approaches are either limited in certain context or arise other security problems. The secure structure for an individual mobile agent is discussed in [10]. Several secure route structures are presented in [17] for protecting a serially migrating agent. But a serial migrating agent can only satisfy small-scale applications and it is not adequate for Grid computing or E-commerce where parallelism is exploited to ensure high performance and fast response. In such a case, dispatching agents in parallel is essential. However, the secure route structures for mobile agents become more complicated.

In this paper, we focus on the issue of efficiently dispatching mobile agents while protecting their routes. We first present a fast binary dispatch model (FBD), which is able to efficiently dispatch a large number of mobile agents in parallel. Based on this model, we present several secure route structures and security enhanced parallel dispatch protocols, which will expose minimal route information to current hosts. The nested structure of secure route can help to detect attacks as early as possible. In terms of security and robustness, these models are improved one by one targeted at preserving the efficiency of the hierarchical dispatch model while ensuring route security. In this paper, we assume a secure mobile agent environment employing well-known public key cryptography [19] and X.509 certification framework [18–20]. In the following, we assume that there exists a secure environment including the generation, certification and distribution of public keys. Each host enables an execution environment for mobile agents and can know the authentic public key of other hosts.

The rest of this paper is organized as follows: Section 2 first previews the BBD model, a basic binary dispatch model. Then it presents FBD model, a fast binary dispatch model. Two secure route structures based on FBD are presented in Section 3. The security properties of two route structures are also compared in this section. The complexities of the route generation of different structures are analyzed in Section 4. Finally, Section 5 concludes this work.

2 A Fast Binary Dispatch Model (FBD)

When there are n mobile agents, a serial dispatch model is to dispatch them one by one. But it is not efficient since the dispatch complexity is $O(n)$.

In [21, 22], we proposed the basic binary dispatch (BBD) model. It is a typical parallel dispatch model where each *parent agent* can dispatch two *child agents* resulting in a binary *dispatch tree* structure with the dispatch complexity of $O(log_2 n)$. We term an agent as a *Master Agent* (e.g. A_0 in Figure 1) if it is created at the home host (e.g. H_0) and is responsible for dispatching a pool of mobile agents to remote hosts. We call an agent a *Worker Agent (WA)* if its sole responsibility is to perform simple tasks assigned to it, e.g. accessing local data. If a *WA* also dispatches other worker agents besides performing the task of local data accessing, it is called a *Primary Worker Agent (PWA)*.

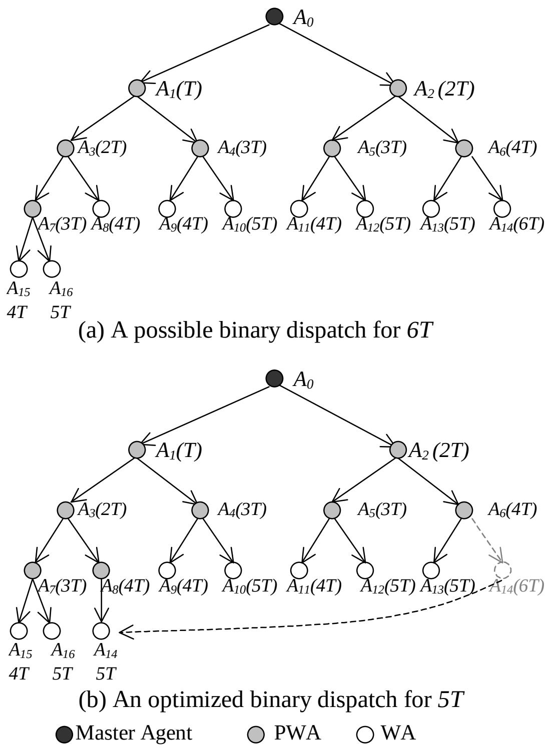
A0
A1(T)
A2 (2T)
A3(2T)
A4(3T)
A5(3T)
A6(4T)
A7(3T) A8(4T) A9(4T) A10(5T) A11(4T) A12(5T) A13(5T) A14(6T)
A15 4T
A16 5T
(a) A possible binary dispatch for 6T
A0
A1(T)
A2 (2T)
A3(2T)
A4(3T)
A5(3T)
A6(4T)
A7(3T) A8(4T) A9(4T) A10(5T) A11(4T) A12(5T) A13(5T) A14(6T)
A15 4T
A16 5T
A14 5T
(b) An optimized binary dispatch for 5T
Master Agent
PWA
WA

Fig. 1. FBD dispatch tree with 16 mobile agents

While the BBD model [21, 22] is efficient, it has a drawback. For example, if there are 16 mobile agents, 8 mobile agents arrive at their destinations and start their local tasks at *4T* and other 8 mobile agents do at *5T*. Here we distinguish the tasks of a *PWA* by dispatch tasks and local tasks. Agent A_1 arrives at its destination at *1T* but it can only start its local data access task at *4T* since it has to dispatch other agents. The start time is the same with agents A_2 to A_8. So do other *PWAs*. In other words, half of the n agents can start their tasks at time $(log_2 n)T$ and the other half at time $(log_2 n + 1)T$.

As shown in Figure 1, in the FBD model, a *PWA* is only responsible for dispatching 1 or 2 child agents before starting its local task. No virtual dispatch is necessary. But to obtain fast dispatch performance, partial adjustment is necessary. As shown in Figure 1, one node should be moved to the left branch so that the overall dispatch time is within $(log_2 n + 1)T$ (see Figure 1b). It is the same with 32 or n (when $n = 2^h$, h is an integer) agents.

We can observe in Figure 1b that A_1 starts its local task at *3T* no matter how many descendent agents it has. It is *4T* for A_2 and A_3, and *5T* for A_4 and A_5 etc. The latest one is $(log_2 n + 1)T$ when having n agents altogether. The final one is the same with BBD model. That means that the starting times of all agents disperse equally from *3T* to $(log_2 n + 1)T$ but the dispatch complexity remains $O(log_2 n)$. This significantly benefits the efficiency when the number of mobile agents is large.

For the implementation strategy of both BBD and FBD models, in IBM Java-based Aglets system [1], if all agents have the same type of tasks with different arguments, a clone-based strategy can be adopted. This can reduce the network bandwidth. Otherwise, all agent classes can be packaged in a JAR file that can be attached with a dispatched agent. A new agent instance can be created from it. For both strategies, the common feature is that when a new agent is created, arguments can be encapsulated before it is dispatched. Here in this paper, we focus on the generic route structures and ignore implementation details.

3 Two Secure Route Structures

In this section, we will discuss possible solutions of secure route structure and dispatch protocol based on the FBD model.

The structure of an agent can be described as follows:

{*Cer0/id0, S, C, D*}

Cer0 is the certificate of its sender, which should be a registered host in a PKI (Public Key Infrastructure) environment. With it, a receiver could verify the ownership of a coming agent. Without loss of generality, for simplicity, *Cer0* can be replaced by the unique id of the sender. S is the state of an agent represented by a set of arguments. A route is part of it. C is the code of the agent and D is the results obtained after execution. It can be sent back through messages.

In the FBD model, if no secure route structure is provided, a host where a PWA resides can know all addresses of the hosts where the PWA's descendant agents should go. Attacks can be easily mounted without being detected.

In this section, to propose several secure route structures, we adopted the combination of public-key encryption and signature schemes. In our protocol, all routes are generated by master agent A_0 at home host H_0 before any dispatch is performed. Routes are encrypted by public keys of corresponding hosts that will be visited. A carried encrypted route can be decrypted with the assistance of the destination host. The host also helps to dispatch child agents when a PWA arrives there. The agent can verify the validity of plaintext using included signature. The host can delete a used route after the corresponding dispatch is successful.

In the following context, we assume the following scenario. A host (say, home host H_0 here) needs to dispatch a pool of mobile agents to other hosts for execution. After generating corresponding secure routes, the master agent A_0

dispatches 2 *PWAs* by FBD, encapsulating secure routes to them and then waits for returned results. To simplify, we also suppose that agent A_i should be dispatched to host H_i where once arriving, A_i should deploy its subsequent child agents if it is a *PWA* or complete its local task if it is a *WA*. In our description, $\bar{h}$ denotes the one-way hash function. P_A denotes the public key of participant A while S_A denotes A's secret key.

Also we will examine if these secure route structures can be used to detect the attacks as follows.

ATK1: route forging attack (forge a route)

ATK2: route delete attack (delete a unused route)

ATK3: dispatch skip attack (skip a predefined dispatch)

ATK4: replay attack (dispatch a forged agent to a visited host)

ATK5: wrong dispatch attack (dispatch an agent to a wrong host)

ATK6: dispatch disorder attack (break the predefined dispatch order)

3.1 Secure Route Structure (I)

During the process of dispatching, a *PWA* resides at the same host without any migration. Its task is to dispatch one or two child agents and then complete its local task.

The secure route structure is as follows:

Secure Route Structure (I)

(i) For a *PWA* A at current host *CH*,

$r(A)=P_{CH}[isPWA, ip(LH), r(LA), ip(RH), r(RA), ip(H_0), t, S_{H_0}(\bar{h}(isPWA, ip(PH), ip(CH), ip(LH), r(LA), ip(RH), r(RA), ip(H_0), id(H_0), t))]$

(ii) For a *WA* A at current host *CH*,

$r(A)=P_{CH}[isWA, ip(H_0), S_{H_0}(\bar{h}(isWA, ip(PH), ip(CH), ip(H_0), id(H_0), t))]$

where

- *r(A)* denotes the route obtained at host H that is encrypted by the public key of H, say P_H;

- *isPWA* or *isWA* is the token showing the current agent is a *PWA* or a *WA*;

- *ip(H)* denotes the address of host H;

- *CH* is the current host; *LH* and *RH* are the left child host and right child host and *PH* is the parent host of *CH*; H_0 is the home host;

- *LA* is the left child agent of A and *RA* is the right one;

- if current agent has only one child agent, *ip(RH)* and *r(RH)* are *NULL*;

- $id(H_0)$ denotes the unique identification of H_0; here for simplification, we use it to represent the ownership;

- t is the unique timestamp when the route is generated at H_0 and it is unique in all routes;

In route structure (I), the route of an agent is encrypted by the public key of its destination host. The route is encapsulated when it is dispatched by its parent agent. Starting the binary dispatch process with secure routes, the master agent A_0 dispatches two *PWAs* to different hosts, each being encapsulated with an encrypted route for future dispatch task. When an agent has successfully arrived at the current host *CH*, it should send back a feedback message to its parent host *PH* confirming the successful dispatch as follows:

$P_{PH}[ip(CH),\ t_R,\ S_{H_0}(\bar{h}(\ldots),\ S_{CH}(ip(CH),\ t_R,\ S_{H_0}(\bar{h}(\ldots))]$

This message is encrypted by the public key of home host including the signature by H_0 included in the dispatched agent's route. t_R is the time when the agent is received.

The carried route $r(A)$ can be decrypted with the secret key of CH so that the agent can know:

- it is a PWA or a WA. This is used to determine if it needs to dispatch child agents;

- the signature signed at host H_0, i.e., $S_{H_0}(\bar{h}(isPWA,\ ip(PH),\ ip(CH),\ ip(LH),\ r(LA),\ ip(RH),\ r(RA),\ ip(H_0),\ t))$ for a PWA, or $S_{H_0}(\bar{h}(isWA,\ ip(PH),\ ip(CH),\ ip(H_0),\ t))$ for a WA.

If it is a PWA, it will also know

- the address *ip(LH)* of the left child host LH and its route *r(LA)*;

- the address *ip(RH)* of the right child host RH and its route *r(RA)*;

For any PWA or WA, the route includes the address of H_0 (i.e. $ip(H_0)$), the home host where A_0 is residing. With this address, the agent can send its result back to A_0.

Next, we illustrate the dispatch process through an example.

1 When A_0 is dispatched to H_1, it carries its route $r(A_1)$.

2 After the route is decrypted, namely
$r=\{isPWA,\ ip(H_3),\ r(A_3),\ ip(H_4),\ r(A_4),\ ip(H_0),\ t,\ S_{H_0}(\bar{h}(\ldots))\}$
A_1 obtains addresses $ip(H_3)$ $ip(H_4)$ and $ip(H_0)$, routes $r(A_3)$ and $r(A_4)$.

3 Then A_1 dispatches agent A_3 to host H_3, encapsulating route $r(A_3)$ to it.

4 Once arriving H_3, A_3 sends back a confirmation message as follows:
$msg = P_{H_1}[ip(H_3), t_{R_3}, S_{H_0}(\bar{h}(\ldots), S_{H_3}(id(H_3), ip(H_3), t_{R_3}, S_{H_0}(\bar{h}(\ldots))]$
where t_{R_3} is the time when H_3 received A_3

5 After that A_1 dispatches agent A_4 to H_4 and receives a message from A_4.

6 Hereafter A_1 will start to complete its local task and return the result to A_0 at H_0.

Clearly, under this model, at any layer, only the addresses of the 2 child hosts are exposed to the current host.

Next, we will examine if route structure (I) and its dispatch protocol can detect the above-mentioned attacks.

Fist, route structure (I) adopts a nested structure. Each route is encrypted by the pubic key of the destination host. It does not need the agent to carry any key.

Second, in each route, a signature by H_0 is included which includes the information of the rest of the route. At a destination, the host could use the public key of H_0 and the public hash function $\bar{h}$ to check the signature and verify the data integrity of the route. Since no party knows the private key of H_0, the signature cannot be forged. That means a forged route can be found by the destination host (*ATK1*). Even if a sub-route (say, *r(LA)* or *r(RA)*) is deleted by current host, the agent can also check the integrity via a trust third party (TTP). And deleting a route will cause no results returned to master agent A_0. So a route deletion attack (*ATK2*) or a dispatch skip attack (*ATK3*) will be found.

Meanwhile since t is unique in all routes and signatures, and signatures cannot be forged, a replay attack can be found by the destination host (*ATK4*). In a signature, the dispatch route, i.e. the path from parent host *PH* to current host *CH* and to child host *LH* or RH, is included also. This can reduce the redundancy of the route (*ip(PH)* and *ip(CH)* appear in the signature only) and detect a wrong dispatch (*ATK5*).

But with route structure (I), a *PWA* could dispatch its right agent first or dispatch agents after the local task is completed. That means the dispatch order may not be strictly followed (*ATK6*). Thus the overall dispatch performance will be worsened. The reason is that two sub-routes for child agents are obtained simultaneously when a route is decrypted. Moreover there is no dependency between two dispatches.

3.2 Secure Route Structure (II)

In the following, an alternative route structure is presented where the route of the right child agent is included in the route of left child agent. When the left child agent is dispatched to the left child host, a feedback is returned to the current agent including the route for the right dispatch. With it, the current agent can dispatch the right child agent to right child host. Hereby, the dispatch order could not be broken (*ATK6*) while the properties against other attacks remain the same.

Obviously in this route, the structures for left dispatch and right dispatch are different since a left dispatch should return a predefined route that is included ahead. For the right dispatch, there is no such a sub-route.

Secure Route Structure (II)

(i) For a *PWA* A at current host *CH*, if *A* is a left child agent of its parent agent at host *PH*, the route for *A* is:

$r(A)=P_{CH}[isPWA,\ ip(LH),\ r(LA),\ ip(RH),\ ip(H_0),\ r(A_{RS}),\ t,\ S_{H_0}(\bar{h}(isPWA,\ ip(PH),\ ip(CH),\ ip(LH),\ r(LA),\ ip(RH),\ ip(H_0),\ r(A_{RS}),\ id(H_0),\ t))]$

where

- A_{RS} is the right-sibling agent of A, namely, the right child agent of A's parent agent;

- *r(RA)* is not included in *r(A)*.

(ii) For a *PWA* *A* at current host *CH*, if *A* is a right child agent of its parent agent at host *PH*, the route for *A* is:

$r(A)=P_{CH}[isPWA,\ K_{PA},\ ip(LH),\ r(LA),\ ip(RH),\ ip(H_0),\ t,\ S_{H_0}(\bar{h}(isPWA,\ K_{PA},\ ip(PH),\ ip(CH),\ ip(LH),\ r(LA),\ ip(RH),\ ip(H_0),\ id(H_0),\ t))]$

where

- K_{PA} is a switch variable for parent agent *PA* that is encrypted by the public key of parent host *PH*, say P_{PH};

(iii) For a *WA* *A* at current host *CH*, if *A* is a left child agent of its parent agent at host *PH*, the route for *A* is

$r(A)=P_{CH}[isWA,\ r(A_{RS}),\ ip(H_0),\ t,\ S_{H_0}(\bar{h}(isWA,\ ip(PH),\ ip(CH),\ r(A_{RS}),\ ip(H_0),\ id(H_0),\ t))]$

where

- A_{RS} is the right-sibling agent of A, namely, the right child agent of A's parent agent;

(iv) For a *WA* A at current host *CH*, if A is a right child agent of its parent agent at host *PH*, the route for A is

$r(A)=P_{CH}[isWA,\ K_{PA},\ ip(H_0),\ t,\ S_{H_0}(\bar{h}(isWA,\ K_{PA},\ ip(PH),\ ip(CH),\ ip(H_0),\ id(H_0),\ t))]$

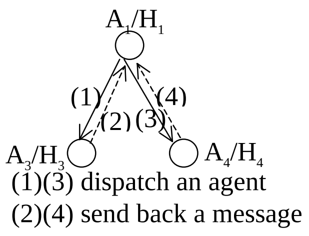

Fig. 2. Dispatch process of structure (II)

In route structure (II), a *PWA* arriving at the destination knows that it has to dispatch 2 child agents and where they should go. But it does not have the route for the right child agent. Only after its left child agent is dispatched can the route for the right child agent be returned and hereafter the right dispatch can be performed. Similar to structure (I), the route for the right agent is encrypted by the public key of the right child host. So the left child host cannot decrypt it and don't know the address where the corresponding agent should go. This could prevent a forged agent to be dispatched to the right child host by the left child agent. In terms of the route structure, the route for the right child agent, say *r(RA)*, is moved from *r(A)* to the route of left child agent *r(LA)* (hereby *r(RA)* is denoted as $r(A_{RS})$). Likewise, in structure (II), a switch variable for current host *CH* is included in the route of its right child agent. Here we assume that each agent has its unique switch variable encrypted by the public key of its destination host. Only after the right child agent is dispatched can current agent obtain it to start its local task.

Next, we will illustrate the dispatch process of agent A_1 (see Figure 2).

1 When A_1 arrives H_1, its decrypted route is

$r=\{isPWA,\ ip(H_3),\ r(A_3),\ ip(H_4),\ ip(H_0),\ t,\ S_{H_0}(\bar{h}(\ldots))\}$

2 A_1 will know it is a *PWA*. Its left child agent is going to H_3 with $r(A_3)$ while its right child agent is going to H_4 but there is no route for it now.

After A_3 is dispatched to H_3, A_1 obtains $r(A_4)$ from a message as follows:

$msg=P_{H_1}[ip(H_3),\ r(A_4),\ t_{R_3},\ S_{H_0}(\bar{h}(\ldots)),\ S_{H_3}(ip(H_3),\ ip(H_3),\ r(A_4),\ t_{R_3},\ S_{H_0}(\bar{h}(\ldots)))]$

where t_{R_3} is the time when H_3 received A_3.

3 Hereby A_4 could be dispatched.

4 From the successful dispatch of A_4, A_1 gets the switch variable K_{A_1} to start its task and return the result to A_0 at H_0.

In fact structure (I) has the same dispatch process as shown in Figure 2. But the returned message is simpler.

Moreover, it is easy to see structure (II) remains the same properties as structure (I) against attacks *ATK1* to *ATK5*. Due to the special arrangement of the route *r(RA)*, the dispatch order will be strictly followed so that the dispatch protocol can prevent dispatch disorder attack (*ATK6*).

The comparison of the security properties of two structures is listed in Table 1.

Table 1. Security Properties of Two Structures

	ATK1	ATK2	ATK3	ATK4	ATK5	ATK6
Route (I)	*Y*	*Y, by* A_0	*Y, by* A_0	*Y*	*Y*	*N*
Route (II)	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*

Y: the attack can be prevented or detected;
N: the attack cannot be prevented or detected.

4 Complexity Comparison of Route Structures

In this section, we analyze the complexity of route generation of different models. To simplify, we assume that the time to encrypt a message of arbitrary-length is a constant, say *C*.

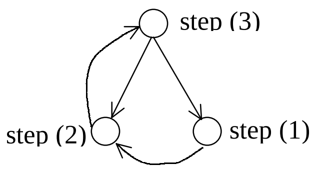

Fig. 3. Steps in the route generation of structure (II)

In structure (I), when a branch has m nodes, the route of the root is generated after two sub-routes are ready, which have *m/2-1* and *m/2* nodes respectively.

$$\begin{cases} T(n) = 2T(n/2) \\ T(m) = T(m/2) + T(m/2-1) + C \qquad (2 \le m \le n/2) \\ T(1) = C \end{cases} \qquad (1)$$

$T(m) = T(m/2) + T(m/2-1) + C) < 2T(m/2) + C$ yields *T(m)=O(m)*. So *T(n)* is *O(n)*.

In route structure (II), the route of the right child agent is generated first (step 1 in Figure 3). Then it is included in the route of the left child agent (step 2 in Figure 3), which is included in the route of the parent agent (step 3 in Figure 3).

If each sub-branch has *m/2* nodes, the complexity is

$$\begin{cases} T(n) = 2T(n/2) \\ T(m) = 2T(m/2) + C \qquad (2 \leq m \leq n/2) \\ T(1) = C \end{cases} \tag{2}$$

So *T(n)* is *O(n)*.

Though structure (II) seems more complex than structure (I), their route generation complexities are the same. The complexity comparison of two structures is listed in Table 2.

Table 2. Complexity Comparison of Two Structures

	Route Generation Complexity	Dispatch Complexity
Route (I)	$O(n)$	$O(log_2 n)$
Route (II)	$O(n)$	$O(log_2 n)$

5 Conclusions

This paper presented two secure route structures and corresponding dispatch protocols based on a fast binary dispatch (FBD) model ensuring both security and efficiency. They expose only minimal addresses to a host to perform dispatches. With the improvement of security performance in structure (II), the complexity of route generation remains unchanged.

For practical applications, mobile agents with the same type tasks and physically close destinations can be put in the same group encapsulated with pre-encrypted routes. For verifying the integrity of a coming agent, the pure code can be included in the signature of a route after being hashed to a fixed length (e.g. 128 bytes by MD5 algorithm) when it is generated at the home host. And the length of the signature remains unchanged.

Though structure (II) has better properties, once a predefined host is not reachable, all members predefined in a branch will not be activated. To resolve this problem, a robustness mechanism should be designed. Furthermore, in our future work, we will conduct experiments comparing the performance differences of different protocols.

6 Acknowledgement

This work was partly supported by National University of Singapore. The authors would like to thank the anonymous reviewers for their valuable comments.

References

1. D. B. Lange, and M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley Press, Massachusetts, USA, 1998
2. S. Papastavrou, G. Samaras, and E. Pitoura, Mobile Agents for World Wide Web Distributed Database Access, IEEE Transactions on Knowledge and Data Engineering, Vol. 12, Issue 5, Sept.-Oct. 2000, pp 802-820
3. D. B. Lange, and M. Oshima, Mobile Agents with Java: The Aglet API, appears in Mobility: Process, Computers, and Agents (edited by Milojicic, D., Douglis, F. and Wheeler, R.), Addison-Wesley Press, Reading, Massachusetts, USA, 1999, pp 495-512
4. I. Foster, C. Kesselman, J. Nick, S. Tuecke,The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002
5. I. Foster, The Grid: A New Infrastructure for 21st Century Science. Physics Today, 55(2):42-47, 2002.
6. I. Foster, C. Kesselman, Computational Grids, Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1999.
7. M. Baker, R. Buyya and D. Laforenza, Grids and Grid Technologies for Wide-Area Distributed Computing, International Journal of Software: Practice and Experience, Volume 32, Issue 15, Wiley Press, USA, 2002.
8. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications, 23:187-200, 2001
9. Y. Wang, K.-L. Tan and J. Ren, A Study of Building Internet Marketplaces on the Basis of Mobile Agents for Parallel Processing, World Wide Web Journal, Kluwer Academics Publisher, Vol. 5, Issue 1, 2002, pp 41-66
10. V. Varadharajan, Security Enhanced Mobile Agents, in Proceedings of the 7th ACM conference on Computer and Communications Security, November 1 - 4, 2000, Athens, Greece, ACM Press, pp 200 - 209
11. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, A Security Architecture for Computational Grids, Proc. 5th ACM Conference on Computer and Communications Security Conference, 1998, pp. 83-92
12. U. G. Wilhelm, Cryptographically Protected Objects, Technical Report, Ecole Polytechnique Federale de Lausanne, Switzerland, 1997
13. E. Palmer, An Introduction to Citadel-a Secure Crypto Coprocessor for Workstations, in Proceedings of IFIP SEC'94 (Curacao, 1994)
14. T. Sander and C.F. Tschdin, Protecting Mobile Agents Against Malicious Hosts, Mobile Agents and Security, LNCS Vol. 1419, Springer-Verlag, 1998, pp 44-60
15. P. Kotzanikolaou, M. Burmester, and V.Chrissikopoulos, Secure Transactions with Mobile Agents in Hostile Environments, ACISP 2000, LNCS 1841, Springer-Verlag, 2000, pp 289-297

16. A. Romao, and M.M. Sliva, Secure Mobile Agent Digital Signatures with Proxy Certificates, E-Commerce Agents, LNAI 2033, Springer-Verlag, 2001, pp 206-220
17. D. Westhoff, M. Schneider, C. Unger and F. Kenderali, Methods for Protecting a Mobile Agent's Route, in Proceedings of the Second International Information Security Workshop (ISW'99), Springer Verlag, LNCS 1729, 1999, pp 57-71
18. P. Wayner, Digital Copyright Protection, SP Professional, Boston, USA, 1997
19. A. Menezes, P. Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996
20. CCITT Recommendation X. 509-1989. The Directory-Authentication Framework. Consultation Committee, International Telephone and Telegraph, International Telecommunication Union, Geneva, 1989
21. Y. Wang and J. Ren, Building Internet Marketplaces on the Basis of Mobile Agents for Parallel Processing, in the Procs. of 3rd International Conference on Mobile Data Management (MDM2002), IEEE Computer Society Press, Jan. 8-11 2002, Singapore, pp 61-68
22. Y. Wang, Dispatching Multiple Mobile Agents in Parallel for Visiting E-Shops, in the Proc. of 3rd International Conference on Mobile Data Management (MDM2002), IEEE Computer Society Press, Jan. 8-11 2002, Singapore, pp53-60