Mobile Networks and Applications, ACM Press/Kluwer Academics Publisher
accepted for publication, to appear in Vol. 8, No. 4, Aug. 2003

Security and Robustness Enhanced Route Structures for Mobile Agents

Yan Wang, Kina-Lee Tan, Xiaolin Pang
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{ywang, tankl, pangxiao}@comp.nus.edu.sg

Abstract. In this paper, we present a parallel dispatch model with secure route structures for protecting the dispatch route information of mobile agents. This model facilitates efficient dispatching of agents in a hierarchical manner, and ensures route security by exposing minimal route information to hosts. It also provides the capability for detecting attacks. Based on the secure dispatch model, two extensions are further presented. The first integrates parallel dispatch with small-scale serial migration to keep the number of mobile agents manageable while preserving similar dispatch performance. The second is a route robustness enhanced mechanism with substitute routes that allows temporarily unreachable hosts to be bypassed. We evaluated the various models both analytically and empirically, and report our findings here.

1 Introduction

In recent years, there have been increasing interests in deploying mobile agents carrying both code and data for distributed processing in an environment such as the Internet. For example, in electronic commerce (EC), a pool of mobile agents can be dispatched from a host to related e-shops to gather 'fresh' information, such as price, stock status, warranty and delivery services, for the products specified by a customer [Corradi99, Rodrigo00]. Clearly, an efficient strategy is to dispatch a large number of agents to work in parallel [Silva99, Papastavrou00, Panayiotou99]. This will also provide customers with the possibility to find the "best" e-shop for his/her purchases.

However, for mobile agent technologies to be accepted, performance and security issues on their use have to be addressed. First, deploying a large number of agents may cause significant overhead when dispatching them. Novel methods for dispatching agents are desirable. Second, when a mobile agent arrives at a host for execution, the code and data will be exposed to the host and the resources at the host may also be exposed to the mobile agent. Thus, security mechanisms should be set up to protect mobile agents from malicious hosts as well as to protect hosts from malicious agents. Some works have been done to restrict an agent's access to the resources of a remote host [Karjoth97, Varadharajan00]. Protecting the agent is also a difficult task. In particular, in an EC environment, since e-shops are competitive, it is important to protect the routes of a mobile agent if it should visit a list of hosts (e-shops) or if it should dispatch mobile agents to other hosts to accomplish the task. If a malicious host knows the route information, it may tamper with it so that its competitors that may offer better prices or services will not be visited. Several secure route structures are presented in [Westhoff99, Li00] for protecting a serially migrating agent. But a serial migrating agent can only satisfy small-scale applications. This calls for novel methods to be designed.

In this paper, we present a secure dispatch route structure for a binary dispatch model that can hierarchically and efficiently dispatch mobile agents in parallel. The model ensures that minimal route information is revealed to a host that dispatches mobile agents. We further propose two extensions. First, we extend the model to facilitate both parallel dispatch and small-scale serial migration to reduce the number of dispatched mobile agents. In this model, a mobile agent can visit several e-shops within the Intranet of a marketplace. Second, we extend the model with encrypted substitute routes to facilitate robustness to allow temporarily unreachable routes to be bypassed to substitute hosts. We report results on both analytical and empirical evaluations of the various models.

In this paper, we employ well-known public-key encryption algorithm, signature generating algorithm and X.509 authentication framework [Wayner97, CCITT]. In the following, we assume that there exists a secure environment including the generation, certification and distribution of public keys and each host can know the authentic public keys of other hosts.

The rest of this paper is organized as follows. Section 2 presents the secure route structure for binary dispatch model. In Sections 3 and 4, we present the two extensions to the model respectively. The complexities of dispatch and route generation of all parallel models are analyzed in Section 5 and the results of experimental study are illustrated in Section 6. In Sections 5 and 6, our parallel models are also compared with 2 existing serial models. Finally, in Section 7, we conclude our work.

2 A Basic Binary Dispatch Model with Secure Route Structure

In this paper, we assume an infrastructure where a set of marketplaces is connected to the Internet. Within each marketplace, there exist multiple e-shops. Requests by users go through the *master customer-agent* A_0 running at the server (say H_0) of Mobile Agent Service Provider (MASP). At H_0, a customer agent can be created as a request from an end-user. 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 on a host. If an agent also dispatches other agents besides performing the task of local data accessing, it is called a *Primary Worker Agent* (PWA). At the stage of price gathering, since a large number of e-shops may sell the same kind of products, efficient dispatch model is needed. Due to limited space, we focus on dispatch issues in this paper. An in-depth discussion on the infrastructure can be found in [Wang01].

2.1 A Binary Dispatch Model

Here, we briefly introduce the binary dispatch model, which is a typical parallel dispatch model where each *parent agent* can dispatch two *child agents* resulting in a binary tree structure. As discussed in [Wang02a], the model can be easily generalized to dispatch *m (m≥2)* agents in parallel.

As shown in Figure 1, master agent A_0 has to dispatch 16 agents to 16 hosts (e.g. agent A_i to host H_i). Now, 16 mobile agents can be divided into 2 groups led by two PWAs, say A_1 and A_9. When agents A_1 and A_9 are dispatched to H_1 and H_9 respectively, each of them has 8 members including itself. For A_1 at layer L_1, it will dispatch its *right child agent* A_5 and distribute 4 members to it. After that A_1 will transfer to layer L_2, which is called a *virtual dispatch* costing no time. Now A_1 has 4 members only. Following the same process, A_1 dispatches A_3 and A_2 successively. During all these processes, A_1 always resides at H_1 without any migration. At the same time when A_1 dispatches A_5, A_0 dispatches A_9 to H_9 to activate all agents in another branch in parallel. At last, after all dispatch tasks have been completed, A_1 becomes a WA and starts its local data-accessing task at H_1. The whole dispatch process can be illustrated by a *dispatch tree*, as shown in Figure 1. As expected, the dispatch complexity is $O(\log_2 n)$ for dispatching n mobile agents.

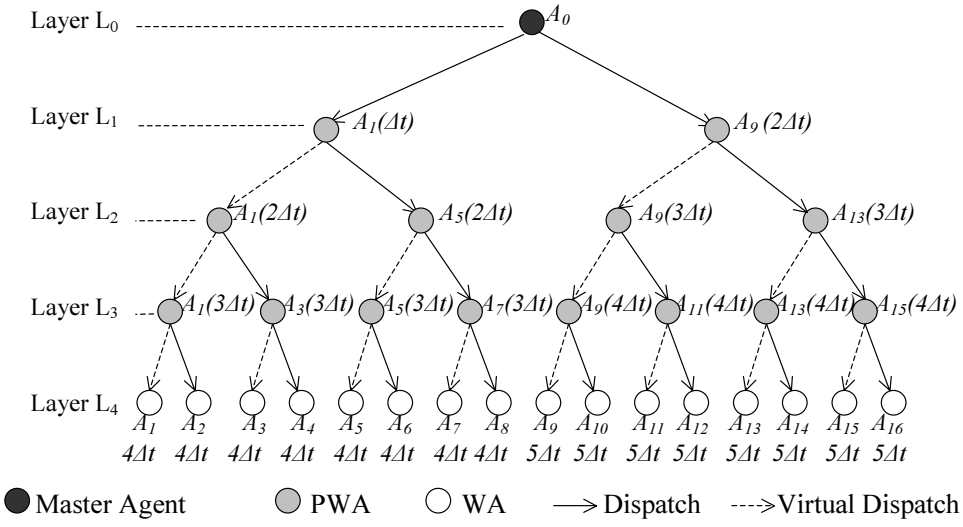


Figure 1 Dispatch Tree with 16 Mobile Agents

2.2 A Secure Route Structure

To ensure route security, we applied cryptographic technique to the basic binary dispatch model. For protecting the routes, we should expose addresses to a host only when necessary. For example, if an agent is at host A, and it has to dispatch an agent to host B, then the address of B must (obviously) be exposed to the host A; however, no other addresses should be exposed.

For the binary dispatch model, it is more complicated than the traditional serial migration model since a PWA has different dispatch tasks at different layers. Only the operations for a WA are simple. For the binary dispatch model, a basic definition of route structure, is as follows:

Route Structure (I)

$$\begin{cases} (1)\text{For a PWA at current host CH,} \\ \quad r(CH)=P_{CH}[isPWA, ip(RH), r_L(CH), r_R(CH), \\ \qquad S_{H0}(H(isPWA, ip(PH), ip(CH), ip(RH), r_L(CH), r_R(CH), t))] \\ (2)\text{For a WA at current host CH,} \\ \quad r(CH)=P_{CH}[isWA, ip(H_0), S_{H0}(H(isWA, ip(PH), ip(CH), ip(H_0), t))] \end{cases}$$

Where

- $r(CH)$ denotes the route at the current host CH, where the agent should reside;
- isPWA or isWA is the token showing the current state of the agent;
- $ip(H_i)$ denotes the IP address of host H_i;
- RH denotes the right child host of current host;
- PH denotes the parent host of current host;
- $r_L(CH)$ and $r_R(CH)$ denote the encrypted route for the left and right children respectively;
- $P_{CH}[M]$ denotes the message M is encrypted by the public key P_{CH} of current host CH;
- $H(isPWA, ip(PH), ip(CH), ip(RH), r_L(CH), r_R(CH), t)$ is the digest with fix-length (e.g. 128 bytes by MD5) returned by a hash function H (e.g. MD5);
- $S_{H0}(D)$ denotes the signature signed on digest D by host H_0 using its secret key S_{H0};
- and t is the timestamp at which the route is generated at host H_0. t is unique for all routes within a dispatch tree.

In route structure (I), $ip(PH)$ and $ip(CH)$ only appear in the signature for verification.

Starting the binary dispatch process with secure routes, the 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, the carried route $r(CH)$ can be decrypted with the secret key of CH so that the agent can know:

(1) it is a PWA or a WA. It is used to determine the next task of the agent;

(2) the signature signed at host H_0: $S_{H0}(H(isPWA, ip(PH), ip(CH), ip(RH), r_L(CH), r_R(CH), t))$ for a PWA, or $S_{H0}(H(isWA, ip(PH), ip(CH), ip(H_0), t))$ for a WA.

If it is a PWA, it will also know

(1) the address $ip(RH)$ of the right child host RH;

(2) the encrypted route $r_R(CH)$ for the right child agent, which can only be decrypted by the right child host;

(3) the encrypted route $r_L(CH)$ for the left dispatch (virtual dispatch).

If it is a WA, it will know the address of H_0, $ip(H_0)$, the home host where A_0 is residing. With this address, the WA can send its result to A_0.

Clearly, in this model, at any layer, only the address of the right child agent is exposed to the current host so that the right dispatch can be completed. For a PWA, if it has $m=2^k$ members, only k addresses of its members are exposed to the current host.

The algorithm for dispatching agents is described as follows:

Algorithm 1: Binary dispatch with secure routes

Step 1: when an agent A is successfully dispatched to a host CH, the host will use the secret key S_{CH}, to decrypt the carried route r(CH), obtaining the route r as:

$$r=S_{CH}[r(CH)]$$

Step 2: if A is a WA, go to step 6; otherwise, A is a PWA, it will dispatch another agent to the host at ip(RH), encapsulating the right dispatch route $r_R(CH)$ to it.

Step 3: if the dispatch is successful, host RH will send back a message including its signature to CH.

$$msg=S_{RH}(H(Entity_{RS}, ip(RH), t_r)) \quad (1)$$

where $Entity_{RS}$ is the full entity of the received agent including its code, state and data. t_r is the timestamp when receiving the agent successfully. H is the hash function.

Once getting such a message, host CH will keep $S_{RH}(H(Entity_{RS}, ip(RH), t_r))$ in its database as a successful dispatch record.

Step 4: Now A should try to complete its left dispatch after decrypting the left dispatch route. Let $r=S_{CH}[r_L(CH)]$

Step 5: if A is still a PWA, go to step 2; otherwise go to step 6.

Step 6: A is a WA and starts its task for local data accessing.

Step 7: when the data access task is completed, A will be disposed after successfully sending a message to agent A_0,

$$msg=P_{H0}[ip(PH),ip(CH),Result_{CH},S_{H0}(isWA,ip(PH),ip(CH),ip(H_0),t),$$
$$S_{CH}(H(ip(PH),ip(CH),Result_{CH},t_{Result}))] \quad (2)$$

where $S_{H0}(H(isWA,ip(PH),ip(CH),ip(H_0),t))$ is the signature from H_0, which is included in the decrypted route of the agent. Here it is used to show the identification of the agent. $S_{CH}(H(ip(PH),ip(CH),Result_{CH},t_{Result}))$ is the signature generated by current host CH. $Result_{CH}$ is the result obtained at CH. PH is the parent host of CH and t_{Result} is the time when getting the result ($t_{Result}>t$).

2.3 Resolving Security Threats

We shall examine several security issues that will be encountered when dispatching mobile agents and show how our model resolves them.

2.3.1 Preventing a PWA from Dispatching a Child Agent

When an agent is dispatching a child agent, a malicious host may peek into the code and modify it to stop the dispatch process in certain layer after the route is decrypted. In the worst case, assuming host H_1 is the malicious one, taking the case shown in Figure 1 as an example, if A_5 is not dispatched, those agents in the group including A_5 to A_8 will not be activated. However this attack can be detected in our model because in such a case agent A_0 cannot receive any messages from each agent of A_5, A_6, A_7 or A_8. If this happens, since the four agents belong to the same group led by agent A_5, A_0 will suspect first that A_5 may have not been dispatched. A_0 will ask hosts H_1 and H_5 to show whether the predefined dispatch has been performed. Apparently, if the dispatch has been carried out, H_1 will receive the confirmation message with the signature $S_{H5}(H(Entity_{A5},ip(H_5),t_r))$ from H_5. H_1 cannot forge this signature without H_5's secret key. So, no matter what H_1 claims, the attack can be detected.

Even if H_1 and H_5 make a collusion attack, the attack can be detected since A_0 cannot receive any message from A_6, A_7 and A_8 and no confirmation information for dispatch can be produced by H_5. If H_6, H_7 and H_8 are also accomplices, the attack may be successful but no honest host is affected.

2.3.2 Dispatch Skip Attack

There is yet another case that can be handled in this model. Let us consider a partial dispatch route: PWA A_i at host H_i dispatches A_j to H_j and A_j dispatches A_k to H_k. In our model, the encrypted route encapsulated to a PWA includes the encrypted route for its right child agent, which can only be decrypted at the child host in the dispatch route. This means when a PWA is dispatching an agent, normally it does not know what the child agent is, a PWA or a WA, and how many members it has. So the case that A_i directly dispatches A_k is not likely to take place without the involvement of A_j. This explains why the encrypted route uses the nested structure. In the worst case, even if H_i can successfully predict that H_k is its descendant in the dispatch route and makes A_i dispatch a forged agent to H_k, the attack will not be successful either since forging the signature is impossible.

The skip attack can be successful only when H_i, H_j and H_k are accomplices. But no honest is affected.

2.3.3 Dispatching an Agent to a Wrong Host

Since the hosts (e-shops) may be competitors, a malicious host may tamper with the addresses so that agents are routed to other hosts instead of the predetermined hosts. The tampering can be done just after the encrypted route is decrypted. However, when an agent is dispatched to a wrong host, its encrypted route will not be correctly decrypted there. Without the correct route, the verification process cannot be undertaken. Even if the destination host can get the correctly decrypted route, the verification will show that is a wrong destination since the address of the destination host is included in the signature in the route generated by H_0 that cannot be tampered with. Thus, in both situations, the attack can be detected by the destination host and the agent will be returned to the sender. Meanwhile, this error will be recorded by the destination host for future investigation.

2.3.4 Sending the Result of a WA to A_0 Directly or Not

In our model, when a WA has fulfilled its data access task, it will send a message to A_0 directly by encrypting the result, the signature by current host as well as the signature by the H_0 originally included in the agent's route. The structure is shown as message (2) in Section 2.2. The whole message is encrypted with the public key of H_0. We choose this strategy in this model with regard to both security and performance issues. An alternative is that a PWA should be responsible for dispatching agents and collecting data from them. But this solution increases the workload of a PWA and the possibility of attacks in the result-returning path.

In comparison, in our model, since a WA only visits one host, the host would not delete the result or prevent its offer from being returned once the agent has been successfully dispatched there. In case the attack occurs, based on the detection of successful dispatch, the problem should be with the host where the agent has arrived. In terms of performance, since each WA has different starting time and ending time for the data-accessing task and the size of each offer will be small, the returned results will not lead to a bottleneck at A_0.

2.3.5 Replay Attack

In a malicious host, the replay attack may occur. Consider the following scenario, that a malicious H_i who has a PWA residing at it and it dispatched agent A_j to host H_j. After the normal process has been completed, H_i may replay the dispatch with a forged agent. However, when an agent is dispatched from H_i to H_j as a replay attack, the timestamp included in the signature from H_0 cannot be tampered with. By verifying the signature, H_j can easily detect the replay attack and H_i will face the risk to be reported.

Similarly, another type of replay attack is for a host, where a WA had earlier resided, to repeatedly counterfeit the WA and send messages to the agent A_0. But it can be easily detected by A_0 by checking the signatures included in messages.

2.3.6 Collusion Attack

If in a normal sequence, host H_a should dispatch an agent to H_b. Assuming H_a and H_c are in a collusion tie, the agent is dispatched to H_c. In this way H_a and H_c make an attempt to skip the visit to H_b who is their competitor and send their own offers instead. However H_c can hardly forge the signature by H_b that should be included in the message returned to A_0. In such a case, the counterfeited message can be detected when it is returned and this will cause the investigation against H_c and H_a. Since H_b will report that no such agent has ever been dispatched to it and H_a cannot show the correct dispatch record which should include the signature by H_b, the attack can be identified. The attack can be successful only when H_a, H_b and H_c make a collusion attack sending a result from H_b

encapsulating the price from H_c. However, in a healthy competitive environment, the probability is fairly low. Even if it can take place, the buying agents will visit H_b not H_c. If H_b cannot offer the product with the provided price, it will result in a commercial cheating, which is the same as a merchant's giving abnormally low price and causing the abortion of the purchase. This will cause the deduction of the merchant's credit standing and little agents will be dispatched later to such merchants.

3 Serial Migration Extension for Parallel Dispatch (PD-SM)

In binary dispatch model, *n* WAs should be dispatched to visit *n* e-shops. This may overload the network traffic. Since several e-shops to be visited may locate in the same intranet of a marketplace, if the number of these e-shops is very limited (e.g. 3 or 4), one agent can be dispatched to the marketplace to visit several e-shops one by one. If many e-shops within a marketplace should be visited, a few agents can be dispatched in parallel and each agent serially visits a set of e-shops. The number of e-shops to be visited by one agent can be determined by a threshold. This will help to reduce the inter-marketplace network load caused by a fully parallel dispatch and will not significantly affect the whole efficiency. Figure 2 illustrates an example with 8 WAs visiting 16 hosts.

Suppose the threshold for serial migration is 3, the secure route structure is described as follows:

Secure Route Structure (II):

(1) For a PWA at CH, $r(CH)=P_{CH}[isPWA,ip(RH),r_L(CH),r_R(CH),$
$S_{H0}[H(isPWA,ip(PH),ip(CH),ip(RH),r_L(CH),r_R(CH),t)]]$

(2) For a WA, assuming the threshold is 3 and the e-shop servers are H_1, H_2 and H_3 in sequence,

$$r(CH)=P_{CH}\big[isWA,ip(H_1),MIG,S_{H0}[H(isWA,ip(CH),ip(H_1),MIG,t)],$$
$$P_{H1}\big[isWA,ip(H_2),MIG,S_{H0}[H(isWA,ip(H_1),ip(H_2),MIG,t)],$$
$$P_{H2}\big[isWA,\ ip(H_3),MIG,S_{H0}[H(isWA,ip(H_2),ip(H_3),MIG,t)],$$
$$P_{H3}[isWA,EoR,ip(H_0),S_{H0}[H(isWA,ip(H_3),ip(H_0),EoR,t)]]]]\big]$$

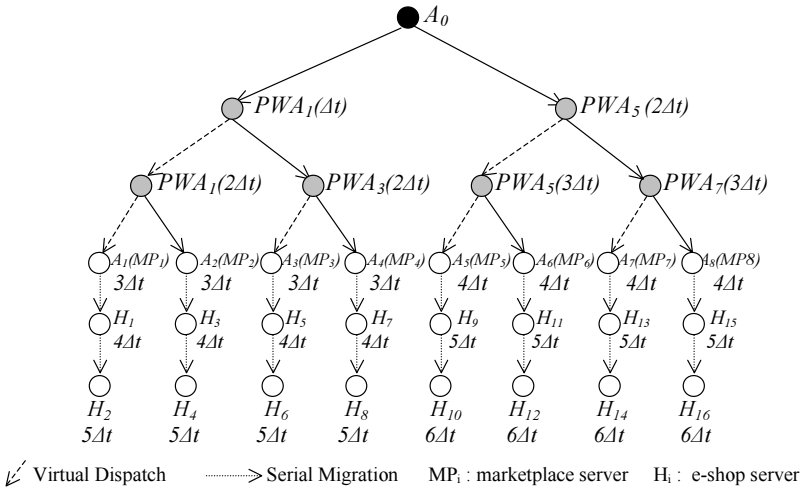


Figure 2 Dispatch Tree with 8 WAs for PD-SM$^+$2

In route structure (II), the route for a PWA is the same as route structure (I).

For a WA, the route is a bit complicated, where MIG is the token meaning that the WA should migrate to the next host after completing the task at the current host and EoR is the token showing the end of the route. When a PWA has completed all its dispatch tasks, it will become a WA and migrate within a marketplace following the predefined sequence as H_1, H_2 and H_3. When having obtained the information from H_3, a WA will send the whole results back to agent A_0 at H_0. The message is as follows:

$$msg=P_{H0}[Result(H_3),ER(H_2),SIG_{H0}(H_3),S_{H3}(H(Result(H_3),ER(H_2),SIG_{H0}(H_3),t_3))](3)$$

where

- $ER(H_i)$ is the encrypted result obtained at H_i;
- $SIG_{H0}(H_i)$ denotes the signature generated by H_0, which is included in the route decrypted by H_i. Here it shows the identification of the agent;
- $ER(H_2)=P_{H0}[Result(H_2),ER(H_1),SIG_{H0}(H_2),S_{H2}(H(Result(H_2),ER(H_1),SIG_{H0}(H_2),t_2))]$;
- and $ER(H_1)=P_{H0}[Result(H_1),SIG_{H0}(H_1),t_3,S_{H1}(H(Result(H_1),SIG_{H0}(H_1),t_1))]$.

In message (3), the results are encrypted by the public key of H_0 in a nested structure. This helps to prevent deletion attack. The signature of current agent is included in the message preventing forged results. And the signature generated by current host shows all are from the correct host.

Figure 2 illustrates an example for PD-SM model with the threshold of 2, which is termed as PD-SM$^+$2.

4 Robustness Enhanced Extension

So far we have presented a security enhanced dispatch model for mobile agents and an extension combining parallel dispatch and serial migration. However, in terms of secure dispatch model, each PWA only knows the right child host RH where its right child agent is to be dispatched at a certain layer. As such, should the right host be unreachable, the right dispatch branch cannot be deployed and all the members grouped in this agent will thereby not be activated. A straightforward solution is for a PWA to have an alternative route for dispatching its right child agent so that if the predefined right child agent cannot be successfully dispatched due to some reasons from the destination host, the PWA can have another route for the right dispatch.

In [Li00] Li proposed a robust model for serial migrating agents and the route robustness is enhanced by equally dividing a route, say $\{ip(H_1),ip(H_2),\ldots,ip(H_n)\}$, into two parts, say $\{ip(H_1),\ldots,ip(H_i)\}$ and $\{ip(H_{i+1}),\ldots,ip(H_n)\}$. They are distributed to two agents A_1 and A_2 respectively. A_1 and A_2 are in partner relationship. Each agent residing at any host en route knows the addresses of the next destination and an alternative host. But the latter is encrypted by the public key of its partner agent. In case the migration cannot be performed, the encrypted address will be sent to the partner agent for decrypting. With its assistance, the agent can continue its migration. The problem with the model is that since both A_1 and A_2 are dynamically migrating, when one needs the other's assistance, locating each other will be costly for both time and system resources. Meanwhile, the model is serial so it is not efficient. Additionally, using the secret key of a dynamically migrating agent is not secure. But the idea of using the mutual assistance of two agents to enhance the robustness is good and can be easily used in our model, where the two first PWAs (e.g. A_1 and A_9 in Figure 1)in the left and right branches can do it better. Since they do not need to migrate, sending messages to them is fairly simple and fast. Encrypting and decrypting a route using the keys of the host where the first PWA resides is more secure.

If we were to provide one substitute route, the route structure in equation (I) can be extended as follows:

Route Structure (III):

(1) For a PWA at current host CH,

$$r(CH)=P_{CH}[isPWA,ip(RH),r_L(CH),r_R(CH),r_R'(CH),$$
$$S_{H0}(H(isPWA,ip(PH),ip(CH),ip(RH),r_L(CH),r_R(CH),r_R'(CH),t))],$$

where $r_R'(CH)=P_{APWA}[ip(SH),r(SH),S_{H0}(H(ip(SH),r(SH),t))]$ is the *substitute route* for the right branch of host CH, SH is the *substitute host*.

(2) For a WA at CH, $r(CH)=P_{CH}[isWA,ip(PH),ip(H_0),S_{H0}(isWA,ip(PH),ip(CH),ip(H_0),t)]$

$r_R'(CH)$ is encrypted by the public key of the first PWA in another branch of the whole dispatch tree, which here is termed as *Assistant PWA* (APWA). For example, in Figure 1, A_1 is the first PWA in left branch so it is the APWA for the right branch following A_9. A_9 is the APWA for the left branch following A_1.

Now suppose A_1 is the first PWA in the left dispatch sub-tree. A_m is the right one. If current host CH is the descendant of A_1, then $r_R'(CH)$ is encrypted by the public key of A_m, say P_{Am}. Otherwise, if CH is in the right dispatch sub-tree from the root node, $r_R'(CH)$ is encrypted by P_{A1}.

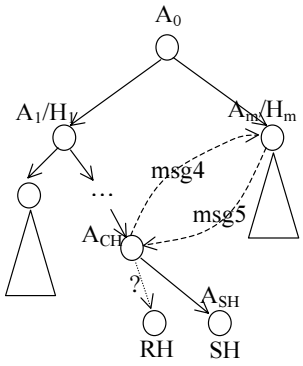
If the dispatch failure occurs when A_{CH} is dispatching its right child agent A_{RH} to right host RH, and A_{CH} is in the left sub-tree, A_{CH} should report it to A_m attaching the substitute route $r_R'(CH)$ (see Figure 3).

$$msg=P_{Hm}[ip(CH),ip(RH),r_R'(CH),S_{CH}(H(ip(CH),ip(RH),r_R'(CH),t_1))] \quad (4)$$

where t_1 is the time when message (1) is generated.

There are two reasons for A_{CH} to send a request to A_m. One is that host RH has a temporary failure when A_{CH} is trying to dispatch an agent there. Another reason is that host CH is malicious and attempts to know more addresses by sending a cheating request. However, the failure report will be confirmed by A_m before it responds to any decrypted routes. And the request is saved by A_m for future investigation.

In this way by route structure (III), a PWA will have a substitute route for the dispatch of its right child agent. Once the original dispatch is not successful, with the assistance of its APWA, it can have another destination to perform the right dispatch.



RH: right child host SH: substitute host

Figure 3 Decrypting A Substitute Route

When A_m gets such a message, it will

Step 1: Detect whether RH is unreachable. If it is true, then go to step 2, otherwise go to step 3

Step 2: A_m will decrypt $r_R'(CH)$, $r=S_{Hm}[r_R'(CH)]=[ip(SH),r(SH),S_{H0}(H(ip(SH),r(SH),t))]$, and send r to A_{CH} through a message
$msg=P_{CH}[ip(SH),r(SH),S_{H0}(H(ip(SH),r(SH),t)),$
$S_{Hm}(H(ip(SH),r(SH),S_{H0}(H(ip(SH),r(SH),t),t_2))](5)$
Stop.

Step 3: If RH is in the correct state, A_m will tell A_{CH} about it and record the request in a database. Stop.

In message (5), the second signature is generated by H_m and t_2 is the corresponding timestamp; SH is the substitute host.

What we should address is that the substitute host is originally included in the members for the right dispatch branch. Taking the dispatch tree in Figure 4 as an example, if the dispatch failure occurs when A_1 at host H_1 is dispatching A_{17} to H_{17}, A_1 can get a substitute route with the assistance PWA A_{33} at H_{33}. To generate the substitute route, choosing H_{18} to be the substitute host is better. By exchanging the positions of H_{17} and H_{18} as shown in Figure 4(b), though H_{18} becomes the root of the branch with H_{17} to H_{32}, most sub-branches under H_{18} is kept unchanged. This is very important and can reduce the complexity to generate a new substitute route.

Following the same idea, the second and the third substitute routes can be generated as shown in Figure 4(c) and 4(d). If H_{18} is not reachable, H_{20} can be the second substitute. If H_{20} is not reachable either, H_{22} can be the new root for this branch and the positions of H_{20} and H_{22} are exchanged (see Figure 4(d)). An originally unreachable host should be put to be a leaf node so that the failure of the second dispatch attempt can be made without increasing more load of the APWA for route decryption. For a detail discussion on generalizing the robust route structure for *k* substitute routes and *k+1* dispatch branches, see [Wang02b].

5 Complexity Analysis

In this section we present some analytical results of our models and compare them with two existing secure models, Westhoff's model [Westhoff99] and Li's model [Li00]. Due to space constraint, we shall not present the proofs of the various results. We refer the readers to [Wang02b].

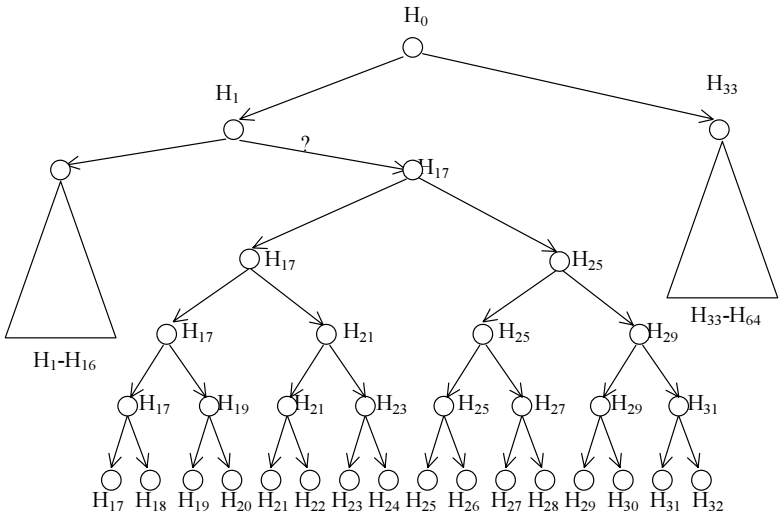
Westhoff's model [Westhoff99] adopted a fully serial migration providing secure route structure without any robustness mechanism. Suppose the visited hosts are H_1, H_2, …, H_n, the route is:

$r(H_i)=P_{Hi}[ip(H_{i+1}),r(H_{i+1}),S_{H0}(H(ip(_{Hi}),ip(H_{i+1}),r(H_{i+1}),t))](1\le i<n)$

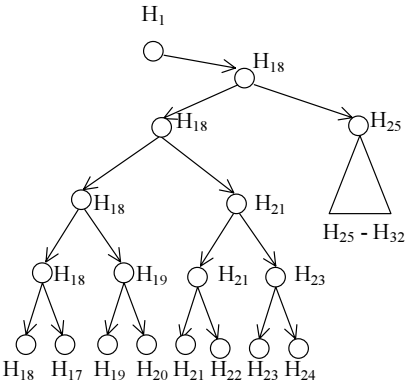
$r(H_n)=P_{Hn}[EoR,S_{H0}(H(ip(_{Hn-1}),ip(H_n),t)]$ (i)

where S_{H0} is the secret key of home host H_0 and EoR is the token meaning the end of the route.

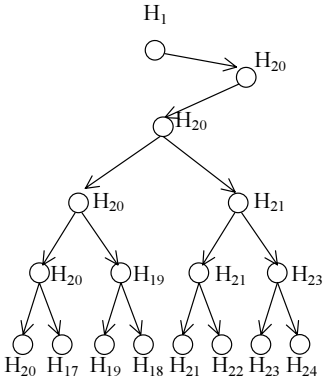
Obviously the migration complexity is *O(n)* if there are *n* hosts to be visited.



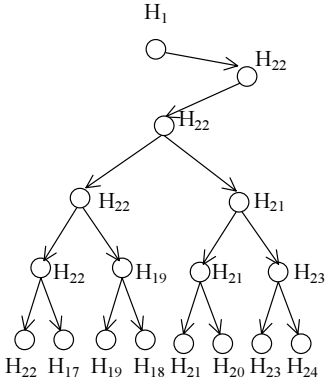
(a) In the original sequence, A_{33} is the APWA for left branch including A_1 to A_{32}



(b) if H_{17} is not reachable, H_{18} becomes a substitute.



(c) H_{20} is the second substitute.



(d) H_{22} is the third substitute.

Figure 4 Examples of Substitute Routes

Li's model [Li00] mentioned in Section 4 ensures both security and robustness. In Li's model, as the addresses of *n* hosts are distributed to two agents, say $\{ip(H_1),\ldots,ip(H_m)\}$ and $\{ip(H_{m+1}),\ldots,ip(H_n)\}$, the nested route structure is:

$r(H_i)=P_{Hi}[ip(H_{i+1}),r(H_{i+1}),r(H_i)',S_{H0}(H(ip(H_{i+1}),r(H_{i+1}),r(H_i)',t))]$ (ii)

where $r(H_i)'=P_{AA}[ip(H_{i+2}),r(H_{i+2}),r(H_{i+2})',S_{H0}(ip(H_{i+1}),r(H_{i+2}),r(H_{i+2})',t)]$ is the substitute route where H_{i+2} is the new destination if H_{i+1} is not reachable. P_{AA} is the public key of the assistant agent.

The whole migration time can be theoretically half of the first model. However the time complexity is *O(n)*.

Theorem 1: Disregarding the time spent on local data access, the time complexity of migration of Westhoff's model and Li's model for visiting *n* hosts is *O(n)*.

In comparison, in our model the efficiency is greatly improved while the security and robustness are ensured in our model. With the binary dispatch model the dispatch complexity is $O(\log_2 n)$.

Theorem 2: If n $(n\geq 2)$ WAs are dispatched by binary dispatch model, $h=\log_2 n$ $(h\geq 1)$ is an integer and the height of the dispatch tree, Δt is the time for dispatching a PWA or a WA, then the total dispatch time for n WAs is

$$T=(h+1)\Delta t.$$

Corollary 1: When all n WAs are dispatched by binary dispatch model, the dispatch time is $T=(\log_2 n+1)\Delta t$ and the time complexity is $O(\log_2 n)$.

Theorem 3: If n $(n\geq 4)$ e-shops should be visited by PD-SM model, m is the threshold for serial migration $(n\geq m)$, H $(H\geq 1)$ is an integer and the height of the dispatch tree before serial migration ($m\cdot 2^H=n$), Δt is the time for dispatching a PWA or a WA including the time for decryption, then the total dispatch and migration time with n WAs is

$$T=(H+1+m)\cdot\Delta t$$

With regard to the complexity for generating routes, the two serial models and the secure binary dispatch model have different performances. Based on the nested secure structure, which helps to prevent route tampering or deleting attacks and detects them as early as possible, assuming that the time to encrypt a route of arbitrary-length is a constant, we have the following results on the complexity for generating routes.

Theorem 4: Assuming that the time to encrypt a route is a constant, the time complexity for generating routes of Westhoff's model is $O(n)$.

Theorem 5: In the secure binary dispatch model, the complexity for generating routes without substitute route is $O(n)$.

Table 1 summarizes and compares the features of the various models without any substitute route.

Table 1 Comparison of Models without Substitute Routes

Features / Models	Nested Secure Route	Dispatch/ Migration Complexity	Route Generating Complexity
Westhoff's Model	*Yes*	$O(n)$	$O(n)$
Binary Dispatch with Secure Routes	*Yes*	$O(\log_2 n)$	$O(n)$
PD-SM	*Yes*	$O(\log_2 n+m)$	$O(n)$

Table 2 Comparison of Models with Substitute Routes

Features / Models	Nested Secure Route	Robust Route	Dispatch/ Migration Complexity	Route Generating Complexity With 1 Substitute Route	Route Generating Complexity With 3 Substitute Routes	Try Failed Hosts Latter
Li's Model [Li00]	*Yes*	*Yes*	$O(n)$	$O(n)$or$O(2^n)$	$O(n)$or$O(4^n)$	*No*
Binary Dispatch with 1 Substitute Route	*Yes*	*Yes*	$O(\log_2 n)$	$O(n\log_2 n)$	*N/A*	*Yes*
Binary Dispatch with 4 Branches and 3 Substitute Routes	*Yes*	*Yes*	$O(\log_2 n)$	*N/A*	$O(n\log_2 n)$	*Yes*

"N/A" means the model does not match corresponding feature.
"No" means the model does not have corresponding feature.

Theorem 6: Assuming that the time to encrypt a route is a constant, the time complexity for generating a route with 1 substitute route in Li's model is $O(n)$.

However, if a failed host is used for a second attempt in Li's model, the complexity for generating routes will become extremely bad since the sequence of hosts in a substitute route has been changed and the route should be generated and encrypted again.

Theorem 7: The time complexity for generating routes with 1 substitute routes of Li's model making the 2nd attempt to the failed hosts is $O(2^n)$.

Theorem 8: In the secure binary dispatch model, the complexity for generating routes with 1 substitute route is $O(n \log_2 n)$.

Table 2 summarizes and compares the features of the various models with substitute routes.

6 Experiments

In Section 5, for simplicity, the analysis is based on the assumption that the encryption time of a message of any length is a constant. To further study the performance of the different models, we conducted some experiments on a cluster of PCs. These PCs are connected to a LAN with 10Mbytes/s network cards PCs running Window NT, JDK, IBM Aglets 1.0.3 [Lange98, ASDK]. For route generations, the experiment is based on a PC of Pentium III 700 MHz CPU and 128 Mbytes RAM and the number of addresses is set up to 1024. For serial migration and binary dispatch models, the experiments are put on a cluster of PCs. Each PC has a Pentium 200MMX CPU and 64 Mbytes RAM. The number of e-shops is set up to 64. All programs run on the top of the Tahiti servers from the ASDK [Lange98, ASDK] and JDK from Sun Microsystems [JDK].

Note that all encrypted routes adopt nested structure so that the performance variations will be totally from the differences in the route structures. To encrypt a route, we use the RSA algorithm [RSA78] and the length of each key is 1024 bits. Before generating a signature, hash function MD5 [Wayner97] is used to generate a hash value with fixed-length of 128 bytes. For the third experiment, the dispatched mobile agent has no task of local data-access. And since all PCs have the same configuration, the performance differences are totally from the difference of serial migration and parallel dispatch.

All results are illustrated in Figures 5 to 10. Each result is the average of four independent executions.

6.1 Experiment 1: Comparison of Route Generation of Models without Substitute Routes

In this experiment, we first compare the route generation time of Westhoff's model and our secure binary dispatch model. All results are shown in Figure 5. When the number of addresses is fewer than 128, the 2 models deliver similar performances. When the number becomes 256 or more, the binary dispatch model begins to outperform the serial model.

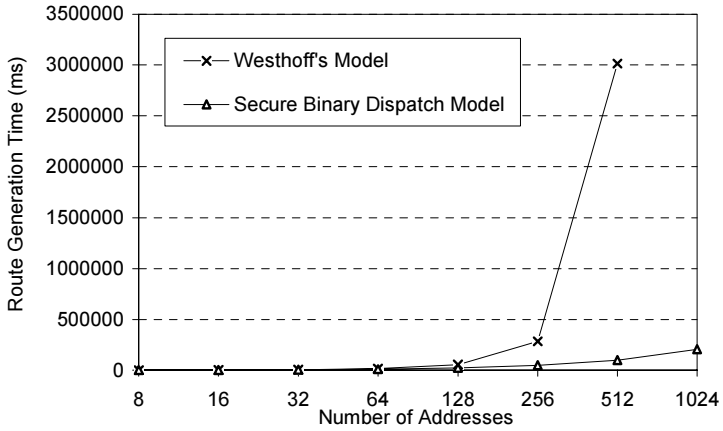


Figure 5 Route Generation Time for Westhoff's Model and Binary Dispatch Model

Theoretically, when there are *n* addresses, the binary dispatch model should do the encryption for *2n-2* times. For the serial model, it is *n* times only. The time complexity of both is *O(n).* If the encryption time for a message is a constant, the route generation time of the binary dispatch model is obviously longer. Nevertheless, the encryption time varies with the length of the encrypted message. For the binary dispatch model, *n* times' encryptions are spent on all leaf nodes in the dispatch tree where the length of each route is only about 200 bytes. Unfortunately, as shown in Figure 6, for Westhoff's model, each time after encryption, the route's length is increased at least with a length of an IP address and a signature. So, the encryption time will gradually increase with the increase of the route length. When the number of addresses is large, the total encryption time will become very long.

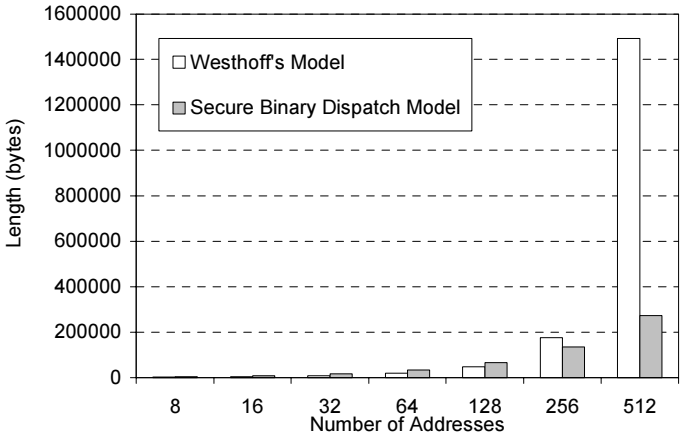


Figure 6 Comparison of Route Length of 2 Models

For example, when there are 512 addresses, the Westhoff's model performs 512 encryptions. As we measure, it uses 284 seconds to complete the first 256 encryptions and 2731 more seconds for the last 256 encryptions. The total time is 3015 seconds. For the binary dispatch model, it completes all encryptions in 101 seconds, and takes 37 seconds for 512 leaf nodes. But when generating the route with 1024 addresses, the program of the Westhoff's model ran out of memory after the 771th address is added where the heap size is set up to 1200 Mbytes and it has reached the maximum.

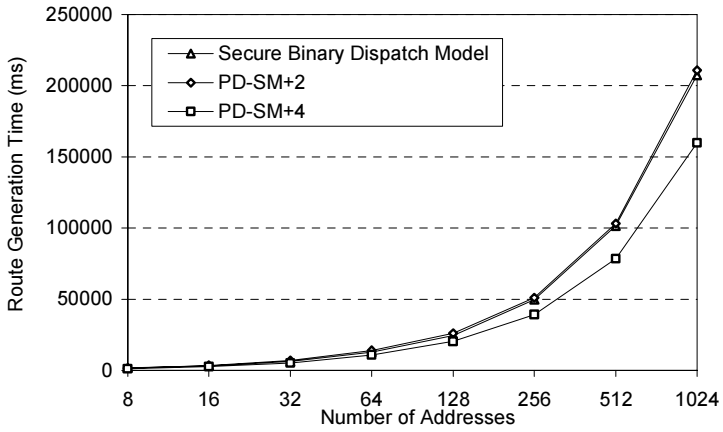


Figure 7 Route Generation Time for 3 Parallel Dispatch Models without Substitute Routes

The results of PD-SM model are shown in Figure 7. We observe that when the threshold is 2, PD-SM$^+$2 delivers almost the same performance as the binary dispatch model. When the threshold is 4, the route generation time from PD-SM$^+$4 is decreased. The reason can be found from Figure 8, where we observe that the route of PD-SM$^+$ model with threshold=4 is shorter than other two cases. That explains why its route generation time is shorter.

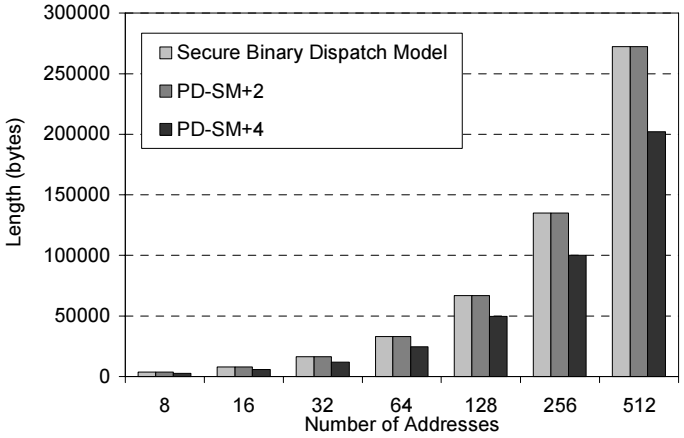


Figure 8 Comparison of Route Length of 3 Models

6.2 Experiment 2: Comparison of Route Generation of Models with One Substitute Route

In this experiment, we compare the route generation time for models with one substitute route. For Li's model, we implemented the case of skipping a failed host.

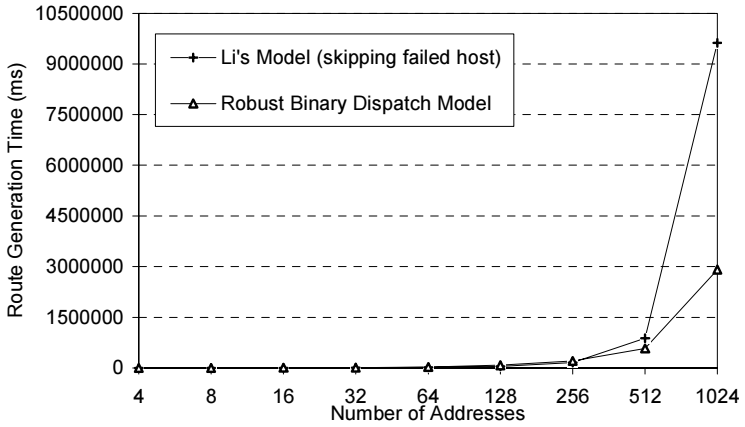


Figure 9 Comparison of the Time for Generating a Route
with 1 Substitute Route

The results shown in Figure 9 illustrates that though the time complexities of the two models analyzed in Section 5 are different (i.e. $O(n)$ vs. $O(n \log_2 n)$), their performances are very close to each other when the number of addresses is not greater than 256. But when the there are 512 addresses, the binary dispatch model begins to outperform. When the address number is 1024, 2 routes are generated and each has 512 addresses. Herein the route generation time of Li's model is too long. The reason is the same as we analyzed in experiment 1.

6.3 Experiment 3: Comparison of Serial Migration Models and Binary Dispatch Models

In this experiment, we tested up to 64 hosts to compare the migration/dispatch time of different models ignoring any robustness mechanisms. In the implementation, a mobile agent will not access any local data so that the measured time is spent for migration or dispatch only. The results are shown in Figure 10 and Table 3.

When the number of visited hosts is 8, the performance differences are not significant. With the increase of the number of hosts, the migration time of any serial migration model increases very fast. In comparison, the dispatch time for binary dispatch model or PD-SM model increases fairly slowly. The performances from PD-SM$^+$2 and PD-

SM^+4 are fairly acceptable (see Table 3). Moreover, the number of dispatched agents is different in the 3 parallel models. They are compared in Table 4.

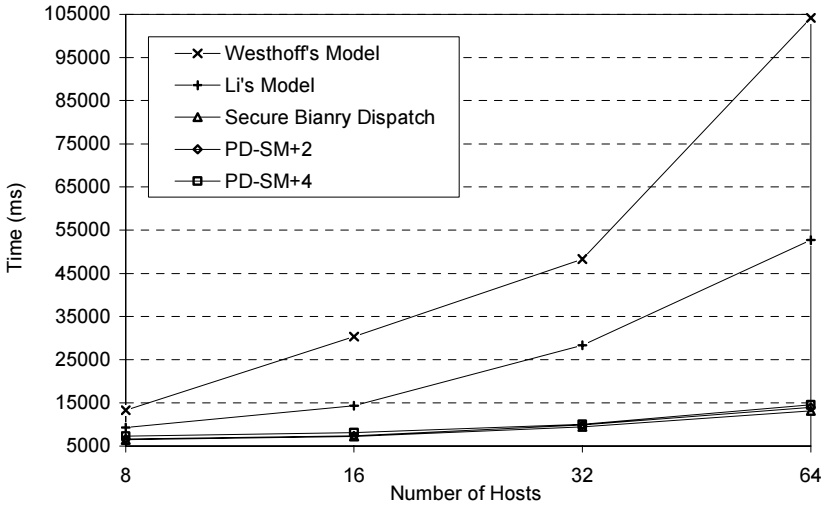


Figure 10 Comparison of Dispatch/Migration Time

Meanwhile, the migration time for Li's model is always shorter than that of Westhoff's model since in Li's model, 2 mobile agents are dispatched and each only visits *n/2* hosts. Nevertheless, its performance is not comparable to the parallel dispatch models. When having 64 hosts, the binary dispatch model can get 73.6% and 86% savings respectively in comparison to Li's model and Westhoff's model.

Table 3 Dispatch/Migration Time in Milliseconds for 5 Models

Number of Hosts / Models	8	16	32	64
Westhoff's Model	*13279*	*30323*	*48300*	*104190*
Li's Model	*9324*	*14321*	*28321*	*52696*
Binary Dispatch	*6467*	*7254*	*9435*	*13158*
PD-SM^+2	*6689*	*7341*	*9897*	*13988*
PD-SM^+4	*7313*	*8124*	*10005*	*14565*

Table 4 The Number of Dispatched Mobile Agents in 3 Parallel Models

Number of Hosts / Models	8	16	32	64	n
Binary Dispatch	*8*	*16*	*32*	*64*	*n*
PD-SM^+2	*4*	*8*	*16*	*32*	*n/2*
PD-SM^+4	*2*	*4*	*8*	*16*	*n/4*

Of course, what should be pointed out is that the time for local tasks is not measured in this experiment. Otherwise, the response time for 2 serial models will become inferior. For the PD-SM model, if the threshold is small, the performance will not be worsened significantly.

7 Conclusions

In this paper we have proposed and discussed binary dispatch models of mobile agents with secure routes and robustness mechanisms. These models utilize the automation and autonomy of mobile agents and the corresponding code is simple. Besides the high efficiency from binary dispatch, the secure mechanism provides the capability to protect mobile agents from malicious hosts and vise versa. Meanwhile, the robustness mechanism enables the fault-tolerance without any loss on security. The experiments show that the binary dispatch models can

not only benefit from route generation, but also can significantly benefit from the parallel dispatch and parallel execution of mobile agents.

In this paper for simplicity we only present and discuss some typical cases of different models that the depth of each leave node is the same. When the number of WAs is not just the power of 2, it will cause the changes of leaf nodes but the performance differences will be similar. For practical applications, mobile agents having tasks of the same type and having physically close destinations can be put in the same group encapsulated with pre-encrypted route structures.

Acknowledgement

This work is supported by the NSTB/MOE funded project on Strategic Program on Computer Security (R-252-000-015-112/303).

References:

[ASDK] http://www.trl.ibm.co.jp/aglets/

[CCITT] CCITT, Recommendation X. 509-1989, "The directory-authentication framework", Consultation Committee, International Telephone and Telegraph, International Telecommunication Union, Geneva, 1989

[Corradi99] A. Corradi, R. Montanari, and C. Stefanelli, "Mobile agents in e-commerce applications", *Proceedings of 19th IEEE International Conference on Distributed Computing Systems, Workshops on Electronic Commerce and Web-based Applications*, Austin, Texas, USA, May 30-June 4, 1999, pp. 59-64

[JDK] http://java.sun.com/products/

[Karjoth97] G. Karjoth, D. B. Lange and M. Oshima, "A security model for Aglets", *IEEE Internet Computing*, July-August 1997, pp. 68-77

[Lange98] D. B. Lange and M. Oshima. *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley Press, Massachusetts, USA, 1998

[Li00] T. Li, C.K. Seng and K. Y. Lam, "A secure route structure for information gathering", *Proceedings of 2000 Pacific Rim International Conference on AI*, 2000

[Panayionou99] C. Panayionou, G. Samaras, E. Pitoura and P. Evripidou, "Parallel computing using Java mobile agents", *Proceedings of 25th EUROMICRO Conference*, Milan, Italy, September 8-10, 1999, Volume 2, pp. 430-437

[Papastavrou00] 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

[Rodrigo00] T. D. Rodrigo and A. Stanski, "The evolving future of agent-based electronic commerce", in *Electronic Commerce: Opportunity and Challenges* (Edited by Rahman S. M. and Raisinghani M. S.), Idea Group Publishing, Hershey, USA, 2000, pp. 337-351

[Silva99] L. M. Silva, V. Batista, P. Martins and G. Soares. "Using mobile agents for parallel processing", *Proceeding of International Symposium on Distributed Objects and Applications (DOA'99)*, Edinburgh, Scotland, September 1999, pp. 34-42

[Varadharajan00] V. Varadharajan, "Security enhanced mobile agents", *Proceedings of the 7th ACM conference on Computer and Communications Security*, November 1 - 4, 2000, Athens, Greece, Pages 200 - 209

[Wang01] Y. Wang, K. L. Tan, J. Ren and X. Pang, "An agent-mediated, secure and efficient Internet marketplace", *Proceedings of the 4th International Conference on Electronic Commerce Research (ICECR-4),* Dallas, TX, USA, November 8 - 11, 2001, pp. 649-641

[Wang02a] Y. Wang, "Dispatching multiple mobile agents in parallel for visiting e-shops", in *Proceedings. of 3rd International Conference on Mobile Data Management (MDM2002)*, IEEE Computer Society Press, Jan. 8-11 2002, Singapore, pp. 61-68

[Wang02b] Y. Wang, K. L. Tan, X. Pang, "Enabling the parallel dispatch of mobile agents with secure and robust Routes ", unpublished manuscript available upon request from authors.

[Wayner97] P. Wayner, *Digital Copyright Protection*, SP Professional, 1997, Boston, USA

[Westhoff99] D. Westhoff, M. Schneider, C. Unger and F. Kenderali, "Methods for protecting a mobile agent's route", *Proceedings of the Second International Information Security Workshop (ISW'99),* 1999, Springer Verlag, LNCS 1729, pp. 57-71