Mobile Information Systems 1 (2005) 185–205
IOS Press

Secure route structures for parallel mobile agents based systems using fast binary dispatch

Yan Wang* and Vijay Varadharajan
Department of Computing, Macquarie University, Sydney, NSW 2109, Australia

Abstract. In a distributed environment, where a large number of computers are connected together to enable the large-scale sharing of data and computing resources, agents, especially mobile agents, are the tools for autonomously completing tasks on behalf of their owners. For applications of large-scale mobile agents, security and efficiency are of great concern. In this paper, we present a fast binary dispatch model and corresponding secure route structures for mobile agents dispatched in parallel to protect the dispatch routes of agents while ensuring the dispatch efficiency. The fast binary dispatch model is simple but efficient with a dispatch complexity of $O(\log_2 n)$. The secure route structures adopt the combination of public-key encryption and digital signature schemes and expose minimal route information to hosts. The nested structure can help detect attacks as early as possible. We evaluated the various models both analytically and empirically.

Keywords: Mobile agent, secure route structure, robust route structure

1. Introduction

The use of mobile agents for distributed applications in a distributed environment is gaining increasing attention. Mobile agents are computational entities that are autonomous, mobile and flexible and can facilitate parallel processing. Very often, a mobile agent acts on behalf of its owner to migrate through the distributed network completing the specified tasks and returning results back to the owner [11,12, 16].

For example, in a national scale Grid environment [2,4–7], 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 to that host. Particularly, in a mobile agent based e-commerce environment [27], a pool of mobile agents can be dispatched at the request of a consumer (end-user) to visit remote e-shops asking offers for a specified product, evaluating these offers and negotiating with the shops. In such environments, the efficiency of dispatching a large number of mobile agents is particularly important in terms of performance. Moreover, the initial route information should be protected against potential malicious hosts. Otherwise, some attacks can be easily mounted breaking the deployment of agents. So, for the application of large-scale mobile agents, security and efficiency are of primary concern [8,20].

*Corresponding author. Tel.: +61 2 98509539; Fax: +61 2 98509551; E-mail: yanwang@ics.mq.edu.au.

Tamper-poof devices [23] and secure coprocessors [15] are hardware-based mechanisms that can be used for protecting mobile agents and hosts. Software-based approaches involve more work, such as the encrypted functions [10,19] and digital signatures with proxy certificates [18]. Security enhanced mobile agents providing a range of security services as discussed in [20]. Several secure route structures are presented in [22] for protecting a sequentially migrating agent. But a sequential migrating agent can only satisfy small-scale applications and it may not be adequate for applications of grid computing or e-commerce where parallelism is exploited to achieve high performance and fast response [27]. In particular, agent based collaborative Internet applications requiring computations distributed over multiple resources will involve dispatching multiple agents in parallel and consolidating the results. Such applications are likely to become increasingly significant in the future. Hence there is a clear need to develop suitable models for the operation of parallel mobile agents and to analyze their security and performance characteristics. This is the main focus of this paper, which addresses dispatching mobile agents in parallel in an efficient manner, while protecting their routing information.

In this paper, 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 also 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 employ well-known public-key encryption algorithm, digital signature generating algorithm and X.509 authentication framework [3, 14,21]. In the following, we assume that there exists a secure environment including the generation, certification and distribution of public keys. Based on such an infrastructure, each host enables an execution environment for mobile agents and can know the authentic public key of other hosts.

2. Background and previous work

2.1. Public-key cryptography, hash function and signature

Public-key cryptography uses two different keys [14,21]. One is the *public key* and the other is the secret key. The public key can be distributed publicly to other parties via digital certificates while the secret key is kept by the owner. Suppose Alice wants to send a message m to Bob in a secure manner. Alice can use the public key of Bob, P_B, to encrypt m as $P_B[m]$ and send it to Bob. Upon receiving the ciphertext, Bob can use his secret key S_B to decrypt the message as $m = S_B[P_B[m]]$. RSA is probably the most well known public-key system [17] at present.

Secret key can also be used to generate a digital signature [14,21]. If Bob wants to send Alice a document D, he can generate the signature as $S_B(D)$ and send it to Alice with the document, denoted as $\{D, S_B(D)\}$ or $D||S_B(D)$. Here ',' or '$\|$' means the concatenation of two pieces of messages. With the signature, Alice can use Bob's public key P_B to check the data integrity of the document. Generally, when generating a signature on a long document, a one-way hash function, denoted as $\mathbb{H}$, can be used to generate digital digest, which operates on an arbitrary-length message m and returns a fixed-length hash value h, where $h = \mathbb{H}(m)$. In this way, the signature, denoted as $sig = S_B\ \mathbb{H}(D)$, will be shorter.

2.2. Basic binary dispatch model (*BBD*)*-A previous work*

When there are n mobile agents, in a sequential dispatch model, the agents are dispatched one by one. But it is not efficient since the dispatch complexity is $O(n)$. In this section, we briefly preview the *basic*

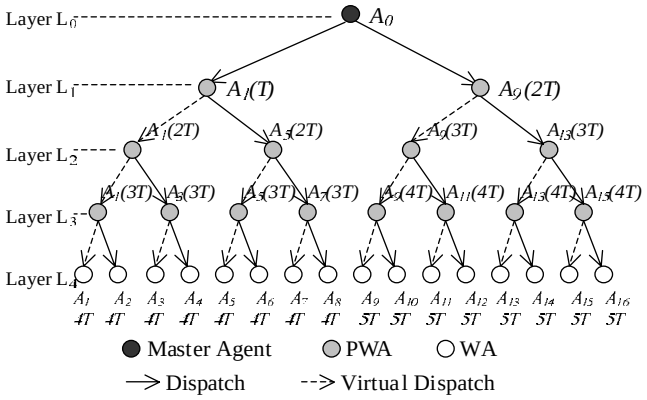


Fig. 1. BBD Model with 16 Mobile Agents.

binary dispatch (BBD) model that is the basis of our previous work in [24–26]. It is a typical parallel dispatch model where each *parent agent* can dispatch two *child agents* resulting in a binary tree structure as shown in Fig. 1.

We term an agent as a *Master Agent* if it is created at the home host 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 such as 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).

As shown in Fig. 1, suppose master agent A_0 has to dispatch 16 agents to 16 hosts (i.e. agent A_i is dispatched to host H_i and H_0 is the home host where A_0 resides). Now, 16 mobile agents are divided into 2 groups led by two PWAs, say A_1 and A_9 respectively. 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. A_5 is a PWA responsible for activating its 4 members in binary. After having dispatched A_5, 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 parallel in another branch. 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 Fig. 1. In fact, to summarize, the tasks of A_1 in Fig. 1 are to act as a PWA and dispatch A_5, A_3 and A_2 in sequence. Then, it becomes a WA.

As a whole, the model benefits from the parallel dispatches by different PWAs at different hosts. When there are $n = 2^h$ mobile agents and T is the average time for dispatching a mobile agent, $(h+1)^T$ will be the time for dispatching n mobile agents. The dispatch complexity will be $O(\log_2 n)$ [27]. This is definitely better than the sequential dispatch model especially when there are large-scale mobile agents. In our previous work [27], we have measured that the BBD model can obtain up to 83%–86% savings, when there are 64 mobile agents and the size of each agent varies from 5 Kbytes to 1 Mbytes, 10 Mbytes and even 100 Mbytes.

3. A new parallel dispatch model and secure route structures

While the BBD model 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

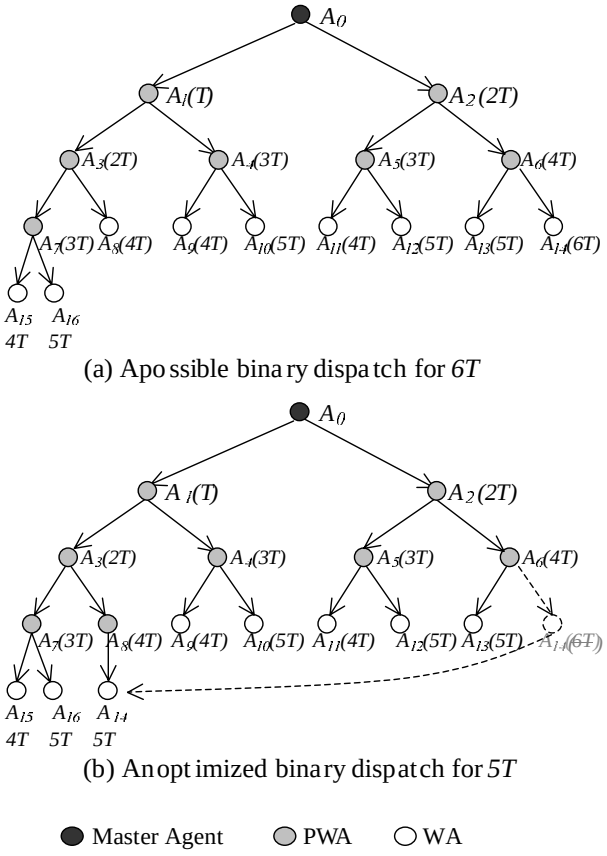


Fig. 2. FBD Dispatch Tree with 16 Mobile Agents.

at $5T$ (see Fig. 1). 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 task at $4T$ since it has to dispatch other agents. The start time of local task 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 $(\log_2 n + 1)T$.

In this section, we propose FBD – a new fast binary dispatch model, and corresponding secure route structures.

3.1. A Fast Binary Dispatch model (*FBD*)

As shown in Fig. 2, 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. That means A_0 dispatches A_1 and A_2. Once arriving the destination host, A_1 dispatches A_3 and A_4 only. After the dispatch, A_1 starts its local accessing task. So does other PWAs. Thus, a PWA can start its local task earlier than BBD. But to obtain the fast dispatch performance, partial adjustment for balance is necessary. As shown in Fig. 2(b), 1 node should be moved to the left branch so that the overall dispatch time is within $(\log_2 n + 1)T$. It is the same with 32 or n (when $n = 2^h$, h is an integer) agents.

We can observe that in Fig. 2(b), 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 last one is $(\log_2 n + 1)T$ when having

n agents altogether. This is same as the BBD model. It means that the starting times of all agents disperse from $3T$ to $(\log_2 n + 1)T$ but the complexity remains $O(\log_2 n)$. As a result, FBD significantly improves the performance when the number of mobile agents is large. In contrast, in BBD, $(\log_2 n)T$ is the starting time of $n/2$ agents for local tasks. $(\log_2 n + 1)T$ is the starting time for the rest $n/2$ agents.

As far as the implementation of both BBD and FBD models is concerned, in Java based agent systems (such as the IBM Aglets Platform [12]), 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 attached with an agent. A new agent instance can be created from this file. In both cases, the common feature is that when a new agent is created, arguments can be encapsulated before it is dispatched. In this paper, we will be focusing on the generic route structure models and will not be discussing our implementation mechanisms.

3.2. Secure route structures

Basically the structure of an agent can be described as follows:

$$\{Cer_0/id_0, S, C, D\}$$

Cer_0 is the certificate of its sender, which should be a registered host in public key based infrastructure (e.g. PKI) environment. With this, a receiver could verify the ownership of a coming agent. Without the loss of generality, for simplicity, Cer_0 can be replaced by the unique identity of the sender. S is the state of an agent represented by a set of arguments. A route is part of this state. C is the code of the agent and D is the results obtained after executions. 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. This will make it vulnerable to several security attacks.

In this section, we propose several secure route structures using the combination of public-key based encryption and signature schemes. In our protocol, all routes are generated by A_0 at H_0 before any dispatch is performed. Routes are encrypted using public keys of the corresponding hosts that will be visited. An encrypted route can be decrypted with the assistance of the destination host. The host also helps dispatch child agents when a PWA arrives there. The agent can verify the validity of plaintext using the 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) 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 in FBD, encapsulating secure routes to them and then waits for the returned results. To simplify, we also suppose that agent A_i should be dispatched to host H_i; upon arrival, A_i should deploy its subsequent child agents if it is a PWA or complete its local task if it is a WA.

After introducing each structure, we will examine whether these secure route structures can be used to detect the following attacks:

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)

All terms and symbols used in this paper are listed in Table 1.

Table 1
Terms and Symbols Used in Our Models

A_0	The master agent at home host H_0.
A_i	A mobile agent residing at host H_i.
A_{RS}	A_{RS} is the right-sibling agent of A, namely, the right child agent of A's parent agent
CH	The current host
$\mathbb{H}$	One-way hash function
H_0	Home host where master agent A_0 resides.
H_i	A host where agent A_i resides.
$ip(H_i)$	The IP address of host H_i.
$isPWA$	A token denoting the current agent is a PWA.
$isWA$	A token denoting the current agent is a WA.
K_{PA}	K_{PA} is a switch variable for parent agent PA that is encrypted by the public key of parent host PH, say P_{PH};
LA	The left child agent
LA'	The substitute agent of left child agent
LH	The left child host of current host CH
LH'	The substitute host of left child host
PA	The parent agent
PH	The parent host of current host CH
P_{Hi}	The public key of host H_i.
$r(A)$	The encrypted route for agent A.
$r'(A)$	The substitute route for agent A.
RA	The right child agent
RH	The right child host of current host CH
SHi	The secret (private) key of host H_i
$S_{Hi}(\mathbb{H}(\ldots))$	The signature of a hash value generated by host H_i
t	The time when a route is generated at H_0.
t_{iR}	The time when agent A_i is received by host H_i.

3.2.1. Secure route structure (*I*)

During the process of dispatch, 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 (**I**) is as follows:

(i) For a PWA A at current host CH,
$r(A) = P_{CH}[isPWA, ip(LH), r(LA), ip(RH), r(RA), ip(H_0), t, S_{H0}(\mathbb{H}(isPWA, ip(PH), ip(CH), ip(LH), r(LA), ip(RH), r(RA), ip(H_0), id(H_0), t))]$

(ii) For a WA A at current host CH,
$r(A) = P_{CH}[isWA, ip(H_0), S_{H0}(\mathbb{H}(isWA, ip(PH), ip(CH), ip(H_0), id(H_0), t))]$
where

- $r(A)$ denotes the route obtained at host H that is encrypted by the public key of H, 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 the id to represent the ownership;
- t is the 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 Ai has successfully arrived at the current host CH, it should send back a feedback message to confirm the successful dispatch as follows

$$msg = P_{PH}[ip(CH), t_{iR}, S_{H0}(\text{H}(...)), S_{CH}(ip(CH), t_{iR}, S_{H0}(\text{H}(...)))] \quad (msg1)$$

This message is encrypted with the public key of home host including the signature by H_0 included in the dispatched agent's route. t_{iR} is the time when the agent A_i is received.

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

- whether 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., $S_{H0}(\text{H}(isPWA, ip(PH), ip(CH), ip(LH), r(LA), ip(RH), r(RA), ip(H_0), t))$ for a PWA, or $S_{H0}(\text{H}(isWA, ip(PH), ip(CH), ip(H_0), t))$ 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 left child host RH and its route $r(RA)$;

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

We illustrate the dispatch process using the following example.

1. When A_0 is dispatched to H_1, it carries its route $r(A_1)$.
2. After the route is decrypted,
   $r = \{isPWA, ip(H_3), r(A_3), ip(H_4), r(A_4), ip(H_0), t, S_{H0}(\text{H}(...))\}$
   A_1 obtained 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 at H_3, A_3 sends back a confirmation message as follows:
   $msg = P_{H1}[ip(H_3), t_{R3}, S_{H0}(\text{H}(...), S_{H3}(id(H_3), ip(H_3), t_{R3}, S_{H0}(\text{H}(...))))](msg2)$
   where t_{R3} 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.
   Hence, under this model, at any layer, only the addresses of the 2 child hosts are exposed to the current host.

3.2.2. Analysis of secure route structure (*I*)

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

(i) Tampering with the route (ATK1 and ATK2)

Since each encrypted route carried by an agent should be decrypted by the current host where the agent resides, the current host may tamper with the route to impact the actions of the agent. But the tamper attack cannot succeed since the signature by H_0 included in the route cannot be changed or forged. Any changes with the route information can be found after verification (ATK1).

Meanwhile, if a sub-route (say, $r(LA)$ or $r(RA)$) is deleted by the current host, the agent can also check the integrity. Deletion of a route will cause no results returned to the master agent A_0. So a route deletion attack (ATK2) will be found.

(ii) Dispatch skip attack (ATK3)

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 (see Fig. 3). It is the same if there are more hosts between H_j and 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 right child host in the dispatch route. This means that when a PWA is dispatching an agent, normally it does not know what the agent is (a PWA or a WA) and how many members the agent has. So the case that A_i directly dispatches A_k is not likely to take place without the involvement of A_j. This is why the encrypted route uses the nested structure. In the worst case, even if H_i can successfully predict that H_k is its descendent in the dispatch route and makes A_i dispatch a forged agent to H_k, the attack will not be successful, since forging the signature is not possible.

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

There is yet another case. Taking the case shown in Fig. 2 as an example, assuming host H_1 is the malicious one, if A_3 is not dispatched, those agents in the group including A_7, A_8, A_{14}, A_{15} and A_{16} will not be activated. However this attack can be detected because in such a case agent A_0 cannot receive any messages from each agent of A_7, A_8, A_{14}, A_{15} and A_{16}. If this happens, since the five agents belong to the same group led by agent A_3, A_0 will suspect first that A_3 may have not been dispatched. A_0 will ask corresponding hosts to show whether the predefined dispatch has been performed. Apparently, if the dispatch has been carried out, a parent host will receive the confirmation message $msg1$ with a signature from the current host. No party can forge this signature without the current host's secret key. So, no matter what H_1 claims, the attack can be detected.

Even if H_1 and H_3 make collusion attack so that A_3 is dispatched but it is made dead at H_3, the attack can be detected since A_0 cannot get any message from A_3, A_7, A_8, A_{14}, A_{15} and A_{16}, and no confirmation information for a successful dispatch can be shown by H_3. Even if H_3, H_7 and H_8 are also accomplices, the attack may be successful but no honest host is affected.

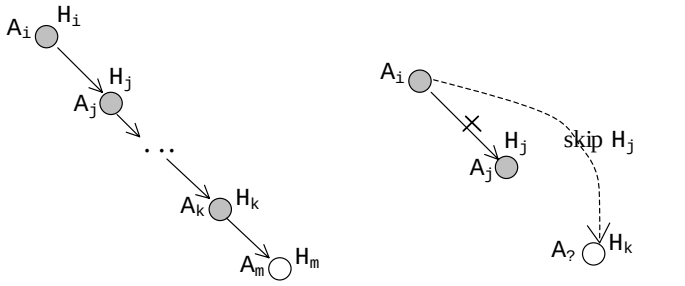
(iii) Replay attack (ATK4)

At a malicious host, replay attack may occur. Consider the following scenario: a malicious H_i who has a PWA residing at its place, dispatches an agent A_j to host H_j. After the normal process has been completed, H_i may replay the dispatch with a forged agent. But the unique timestamp included in the signature by H_0 makes the signature different from others. Therefore, when an agent is dispatched from H_i to H_j as a replay attack, by checking the signature, H_j can easily detect the attack and H_i will face the risk to be reported.

Similarly, another type of replay attack by a host, where a WA has earlier resided, is to repeatedly counterfeit the WA and send messages to agent A_0. But it can be easily detected by A_0 by checking the signatures included in messages, which are generated by H_0 with unique timestamps and are taken as the identification of the agents.

(iv) Wrong dispatch (ATK5)

Since the hosts may be in a competitive situation (such as in e-commerce environments), if a malicious host knows that a child agent will be dispatched to a remote host from its server, and that the remote host may probably give a better offer, it may tamper the address so that the agent can be dispatched to another host without any competitive offer. The tamper process can be done just after the encrypted route is decrypted. Normally, when a host receives an agent, since its address appears in the signature in the agent's route generated by H_0 that cannot be tampered with, it can verify if it is the correct destination. However, when an agent, say A_j, is dispatched to a wrong host, say H_w, its encrypted route $r(A_j)$ will not be correctly decrypted there (see Fig. 4). Without the correct route, the verification process cannot be undertaken. Even if the destination host can get the correctly decrypted route, the route will show



(a) Normal Dispatch (b) Skip the Dispatch to H_j

Fig. 3. Dispatch Skip Attack.

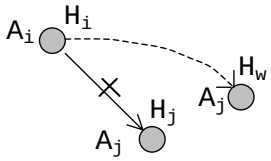


Fig. 4. Wrong Dispatch Attack.

that it is a wrong destination. 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.

(v) Breaking the dispatch sequence (ATK6)

However with route structure (I), a PWA could dispatch its right child agent first or dispatch agents after the local task is completed. This means that the dispatch order will not be strictly followed (ATK6). Thus the overall dispatch performance will be worsened. If all agent/hosts break the dispatch sequence, the whole dispatch performance will be as worse as a sequential migration model. The reason of this weakness is that two sub-routes for child agents are obtained simultaneously when a route is decrypted. And there is no sequence dependency between these two dispatches.

3.3. 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 (parent) agent including the route for the right dispatch. Then the current agent can dispatch the right child agent to right child host. 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:
$r(A) = P_{CH}[isPWA, ip(LH), r(LA), ip(RH), ip(H_0), r(A_{RS}), t, S_{H0}(\mathbb{H}(isPWA, ip(PH), ip(CH), ip(LH), r(LA), ip(RH), ip(H_0), r(A_{RS}), id(H_0), t))]$
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:
$r(A) = P_{CH}[isPWA, K_{PA}, ip(LH), r(LA), ip(RH), ip(H_0), t, S_{H0}(\mathbb{H}(isPWA, K_{PA}, ip(PH), ip(CH), ip(LH), r(LA), ip(RH), ip(H_0), id(H_0), t))]$
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
$r(A) = P_{CH}[isWA, r(A_{RS}), ip(H_0), t, S_{H0}(\mathbb{H}(isWA, ip(PH), ip(CH), r(A_{RS}), ip(H_0), id(H_0), t))]$
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 of A is
$r(A) = P_{CH}[isWA, K_{PA}, ip(H_0), t, S_{H0}(\mathbb{H}(isWA, K_{PA}, ip(PH), ip(CH), ip(H_0), id(H_0), t))]$

In protocol (II), if a PWA has only one child agent, the structure of the child agent is the same as (iv).

In route structure (II), a PWA arriving at the destination knows that it has to dispatch two 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 the right dispatch can be performed. Similar to structure (I), the route for the right agent is encrypted with 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 with the public key of its destination host. Only after the right agent is dispatched can current agent obtain it to start its local task.

Now we illustrate the dispatch process of agent A_1 (see Fig. 5).

1. When A_1 arrives H_1, its decrypted route is
$r = \{isPWA, ip(H_3), r(A_3), ip(H_4), ip(H_0), t, S_{H0}(\mathbb{H}(...))\}$
2. A_1 will know it is a PWA and its left child agent is going to H_3 with $r(A_3)$ and its right child agent is going to H_4 but there is no route for it now.

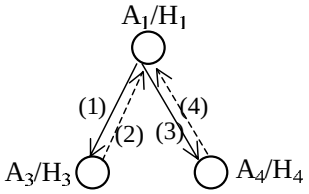


Fig. 5. Dispatch Process of Structure (II).

3. After A_3 is dispatched to H_3, A_1 obtains $r(A_4)$ from a message as follows:
   $msg = P_{H1}[ip(H_3), r(A_4), t_{3R}, S_{H0}(\text{H}(...)), S_{H3}(ip(H_3), ip(H_3), r(A_4), t_{3R}, S_{H0}(\text{H}(...)))]$
   where t_{3R} is the time when H_3 received A_3.
4. Now A_4 could be dispatched.
5. From the successful dispatch of A_4, A_1 gets the switch variable T_{A1} 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 Fig. 5. But the returned message is simpler. As structure (II) adopts the same nested structure and signatures from H_0 are included in all routes, it is easy to see structure (II) has similar properties to structure (I) against attacks ATK1 to ATK5. From the above example, we can observe that agent A_1 must dispatch A_3 and A_4 in the predefined order before executing its local task. Therefore, due to the special arrangement of the route $r(RA)$, the sequences of a PWA's actions are dependant on each other. The dispatch order has to be strictly followed. So this dispatch protocol can prevent the dispatch disorder attack (ATK6).

3.4. Robustness extension – secure route structure (*III*)

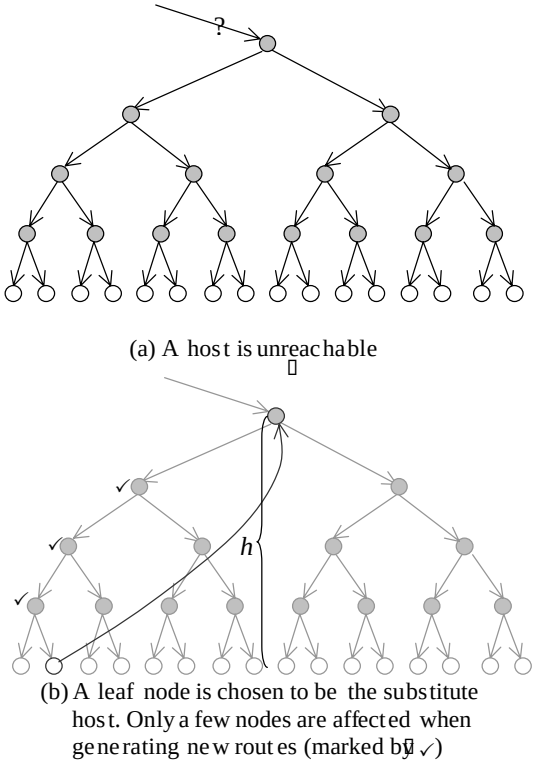
Dispatch protocol (II) could ensure the dispatch order to be strictly followed. But, when a predefined destination host is not reachable, the predefined dispatch cannot be performed any more. To address this issue, the failure can be reported to A_0 so that a new package is generated excluding the unreachable host. But it is costly. An available solution is to prepare substitute routes for substitute hosts. Once a predefined host is not reachable, the agent is dispatched to the substitute host with the substitute route to deploy the rest agents.

Hence the robust route structure extension is as follows:

Robust Route Structure (**III**)

(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:
$r(A) = P_{CH}[isPWA, ip(LH), r(LA), r'(LA'), ip(RH), ip(H_0), r(A_{RS}), t, S_{H0}(\text{H}(isPWA, ip(PH), ip(CH), ip(LH), r(LA), r'(LA'), ip(RH), ip(H_0), r(A_{RS}), id(H_0), t))]$
where $r'(LA') = P_{AH}[ip(LA'), r(LA'), t, S_{H0}(ip(LA'), r(LA'), t)]$

(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:
$r(A) = P_{CH}[isPWA, K_{PA}, ip(LH), r(LA), r(LA'), ip(RH), ip(H_0), t, S_{H0}(\text{H}(isPWA, K_{PA}, ip(PH), ip(CH), ip(LH), r(LA), r'(LA'), ip(RH), ip(H_0), id(H_0), t))]$
where LH' is the substitute host and $r'(LA')$ is the substitute route;



(a) A host is unreachable

(b) A leaf node is chosen to be the substitute host. Only a few nodes are affected when generating new routes (marked by ✓)

Fig. 6. The Selection of a Substitute Host.

(iii) The route for a WA is the same as structure (II).

Note that $r'(LA')$ here has a different structure from $r(LA)$. $r(LA')$ is the same as presented in structure (II-i) for a left child agent. But it is included in $r'(LA')$, which is encrypted with the public key of a Assistant Host (AH).

In Fig. 1, H_1 is the left AH for all agents rooted by A_9 and H_9 is the AH for all agents rooted by A_1. Here we simply assume that the address of the AH is public. When a predefined host is not reachable, the current host will report it to its AH attaching the route $r'(LA')$. After confirming the fault, AH will decrypt $r'(LA')$ and send $r(LA')$ back, with which the dispatch could continue.

If a substitute host is chosen to be one of the original n hosts, a strategy (strategy (1)) illustrated in Fig. 6 can be adopted. A leaf node, which is originally a right child host, is chosen to replace the unreachable host (see Fig. 6(b)). The benefit is that the height of the branch rooted by the unreachable host is h; only $h-1$ nodes are needed to re-generate the routes. This is important to reduce the complexity of route generation. But if the substitute host can be chosen out of original n hosts, an extra host can be specified in advance when generating the routes (strategy (2)). This will make the route generation simpler.

4. Complexity analysis

A comparison of the security properties of three models is listed in Table 2.

In this section, we analyze the complexity of route generation of three models and compare them with existing models. To simplify, we assume that the time to encrypt a message of arbitrary given length is

Table 2
Security Properties of Three Models

	Route forge attack	Route delete attack	Dispatch skip attack	Replay attack	Wrong dispatch attack	Dispatch disorder attack	Robust route structure
Route (I)	√	√	√	√	√	X	No
Route (II)	√	√	√	√	√	√	No
Route (III)	√	√	√	√	√	√	Yes

√: the attack can be prevented or detected;
X: the attack cannot be prevented or detected.

a constant, say C.

The model presented in [22] adopted a fully sequential migration providing secure route structure without any robustness mechanism. Let us suppose that the visited hosts are $H_1, H_2, \ldots, H_n$. Then the route is:

$$r(H_i) = P_{Hi}[ip(H_{i+1}), r(H_{i+1}), S_{H0}(ip(H_i), ip(H_{i+1}), r(H_{i+1}), t)] \quad (1 \leqslant i < n)$$
$$r(H_n) = P_{Hn}[EoR, S_{H0}(ip(H_{n-1}), ip(H_n), t]$$

where S_{H0} is the secret key of home host H_0 and E_oR is the token meaning the end of the route. The migration complexity is $O(n)$ if there are n hosts to be visited.

A robust sequential model proposed in [13] ensures both security and robustness. In the robust sequential 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:

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

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 $O(n)$. For both sequential models the complexity for route generation is $O(n)$ [24]. For BBD model, as we analyzed in [24], the complexity for route generation is $O(n)$.

In the following context, we analyze the complexity of the proposed secure route structures.

Theorem 1. Assuming that the time to encrypt a message of arbitrarily given length is a constant, the time complexity for route generation in structure (I) is $O(n)$.

Proof: With 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.

Thus, we have

$$\begin{cases} T(n) = 2T(n/2) \\ T(m) = T(m/2) + T(m/2 - 1) + C \quad (2 \leqslant m \leqslant n/2) \\ T(1) = C \end{cases} \tag{1}$$

$$\begin{aligned} T(m) &= T(m/2) + T(m/2 - 1) + C < T(m) = 2T(m/2) + C < T(m) = 2(2T(m/4) + C) \\ &+C = 4T(m/4) + (1 + 2)C < 4(2T(m/8) + c) + (1 + 2)C = 8T(m/8) + (1 + 2 + 4)C \\ &= 2^k T(m/2^k) + (2^0 + 2^1 + \ldots + 2^{k-1})C \\ &= 2^k T(m/2^k) + (2^k - 1)C \quad (2 \leqslant m \leqslant n/2) \end{aligned}$$

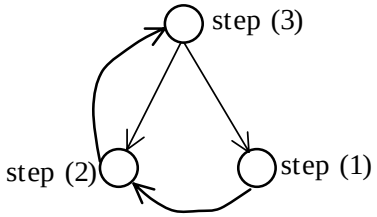


Fig. 7. Steps in Route Generation of Structure (II).

If there are n agents and each branch has m agents, namely, when $n = 2m = 2^h$, $T(n) < 2^h T(1) + (n-1)C < (2n-1)C$.

Therefore $T(n)$ is $O(n)$. □

In route structure (II), the route of the right child agent is generated first (step 1 in Fig. 7). Then it is included in the route of the left child agent (step 2 in Fig. 7), which is included in the route of the parent agent (step 3 in Fig. 7).

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

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

Similar to Eq. (1), $T(m) = 2T(m/2) + C$ yields $T(m) = 2^k T(m/2^k) + (2^k - 1)C$ $(2 \leqslant m \leqslant n/2)$. When $n = 2m = 2^h$, $T(n) = 2(n-1)C$. So $T(n)$ is $O(n)$. Therefore, we have the following theorem.

Theorem 2. Assuming that the time to encrypt a message of arbitrarily given length is a constant, the time complexity for route generation in structure (II) is $O(n)$.

For structure (III), we have the following theorem.

Theorem 3. Assuming that the time to encrypt a message of arbitrarily given length is a constant, the time complexity for route generation in structure (III) is $O(n)$ or $O(n \log_2 n)$.

Proof: In structure (III), a substitute route is added. To generate the substitute route in strategy (1), as most sub-branches remain unchanged, only h nodes are needed to re-generate their routes, where k is the height of the branch rooted by the substitute host (see Fig. 6(b)). So if the branch has m ($m = 2^k$) nodes, the complexity is

$$\begin{cases} T(n) = 2T(n/2) \\ T(m) = 2T(m/2) + kC(m = 2^k) \quad (2 \leqslant m \leqslant n/2) \\ T(1) = C \end{cases} \tag{3}$$

Otherwise, if strategy (2) is adopted, then

$$\begin{cases} T(n) = 2T(n/2) \\ T(m) = 2T(m/2) + 2C \quad (2 \leqslant m \leqslant n/2) \\ T(1) = C \end{cases} \tag{4}$$

Table 3
Complexity Comparison of Three Models

	Complexity of Route Generation	Complexity of Dispatch
Route (I)	$O(n)$	$O(\log_2 n)$
Route (II)	$O(n)$	$O(\log_2 n)$
Route (III)	$O(n \log_2 n)$ *or* $O(n)$	$O(\log_2 n)$

From Eq. (3), we have

$$\begin{aligned} T(m) &= 2T(m/2) + kC = 2(2T(m/4) + kC) + kC = 4T(m/4) + (1+2)kC \\ &= 4(2T(m/8) + kC) + (1+2)kC = 8T(m/8) + (1+2+4)kC \\ &= 2^k T(m/2^k) + (2^k - 1)kC \end{aligned}$$

When $m = 2^k$, $T(m) = mC + (m-1)(\log_2 m)C$ ($2 \leqslant m \leqslant n/2$).
As $T(n) = 2T(n/2)$, so $T(n) = 2(n/2 \cdot C + (n/2-1)(\log_2 n/2)C) = nC + (n-2)(\log_2 n - 1)C$
Therefore $T(n) = O(n \log_2 n)$.
From Eq. (4), we have

$$\begin{aligned} T(m) &= 2T(m/2) + 2C = 2(2(m/4) + 2C) + 2C \\ &= 4T(m/4) + (2+4)C = 4(2T(m/8) + 2C) + (2+4)C \\ &= 8T(m/8) + (2+4+8)C \\ &= 2^k T(m/2^k) + 2(1 + 2 + ... + 2^{k-1})C \\ &= 2^k T(m/2^k) + 2(2^k - 1)C \end{aligned}$$

When $m = 2^k$, $T(m) = mC + 2(m-1)C = (3m-2)C$ ($2 \leqslant m \leqslant n/2$).
As $T(n) = 2T(n/2)$, so $T(n) = 2(3n/2 - 2)C = (3n-4)C$.
Therefore $T(n)$ is $O(n)$. □

The complexities of three models are illustrated in Table 3.

5. Experimental results

To further study the performance of the different models proposed above, we conducted experiments on a cluster of PCs. These PCs are connected to a LAN with 100 Mbytes/s network cards running Window NT, JDK, IBM Aglets 1.0.3 [1,9]. For route generations, the experiments are based on a PC of Pentium IV 1.8 GHz CPU and 512 Mbytes RAM. For sequential migration and binary dispatch, the experiments are put on a cluster of PCs of Pentium 200MMX CPU and 64 Mbytes RAM. All programs run on the top of the Tahiti servers from the ASDK [1,12] and JDK from Sun Microsystems [9].

To encrypt a route, we used the RSA algorithm [17] and the length of each key used is 1024 bits. Before generating a signature, hash function MD5 [14] is used to generate a hash value with a fixed-length of 128 bits. For the third experiment, since all PCs have the same configuration, the performance differences arise totally due to the differences in the sequential versus parallel dispatch models. In our experiments, we also compare our models with the secure route structures of sequential migration [13, 22] in which the complexities for route generation are all $O(n)$. The results are illustrated in Fig. 8 to Fig. 14. Each result is the average of four independent executions

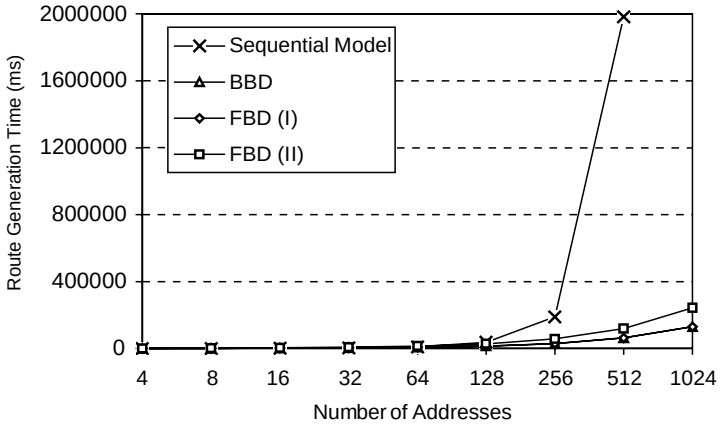


Fig. 8. Route Generation Time for Sequential Model and Binary Dispatch Model.

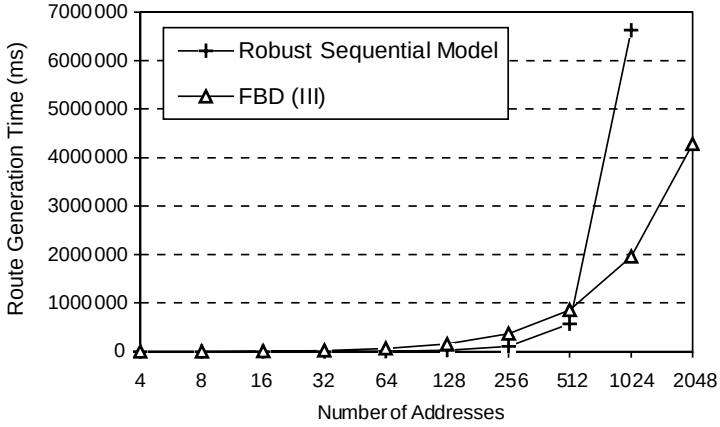


Fig. 9. Comparison of the Time for Generating a Route with One Substitute Route.

5.1. Experiment 1: Route generation-sequential model vs. binary models

In this experiment, we first compare the route generation time of different models. Here four secure structures are compared. Results are shown in Fig. 8. When the number of addresses is fewer than 128, all models deliver similar performances. When the number becomes 128 or more, the binary dispatch model begins to outperform the sequential model.

The route generation performances of the four secure structures are pretty close to each other. The time for FBD (II) is longer than for BBD and FBD (I). For BBD and FBD (I), the performances are very close to each other. With the increase of the number of addresses, the time for sequential model increases very fast. When generating the route with 1024 addresses, the program of the sequential model ran out of memory after the 771st address is added, where the heap size was set to 1200 Mbytes and the maximum had been reached. But for FBD (II), it takes 243 seconds for 1024 addresses. It can even generate routes with up to 2048 addresses for 483 seconds. Generally, the time for structures (I) to (V) increases fairly slowly.

Theoretically, when there are n addresses, the binary dispatch model should do the encryption for $2n - 2$ times. For the sequential model, it is n times only. The time complexities are both $O(n)$. If

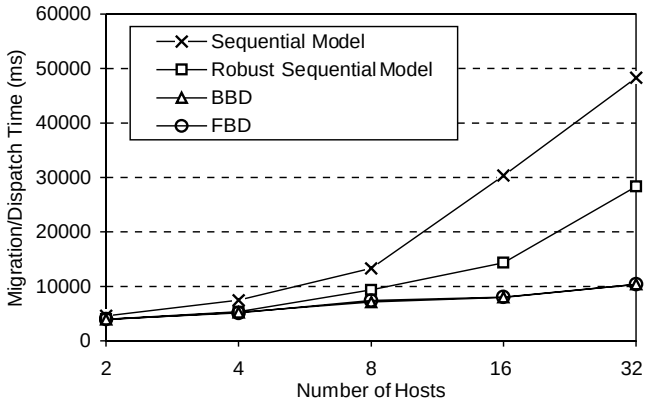


Fig. 10. Comparison of The Migration/Dispatch Time.

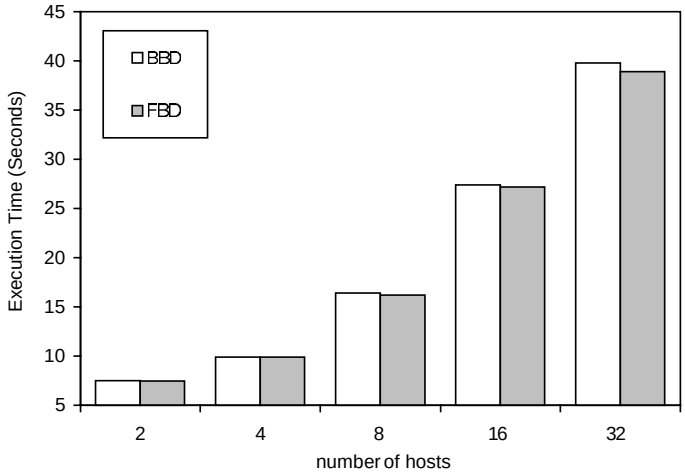


Fig. 11. Results of Reading 1 Kbytes Data from the 1 Mbytes XML File of Every Host.

the encryption time for a message is a constant, the route generation time for 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, for sequential model, each time after encryption, the route's length increased with a length of a network address and a signature. So the encryption time gradually increases with the increasing route length. When the number of addresses is large, the total encryption time will become very long.

For example, when there are 512 addresses, the sequential model performs 512 encryptions. It took some 190 seconds (about 9.6% of overall time) to complete the first 256 encryptions and some 1793 seconds (about 90.4% of overall time) for the last 256 encryptions. The total time is 1983 seconds. For the binary dispatch model (structure FBD (II)), it completed all encryptions in 118 seconds for 512 nodes with 57.6 seconds (about 48.7% of overall time) for first 256 leaf nodes.

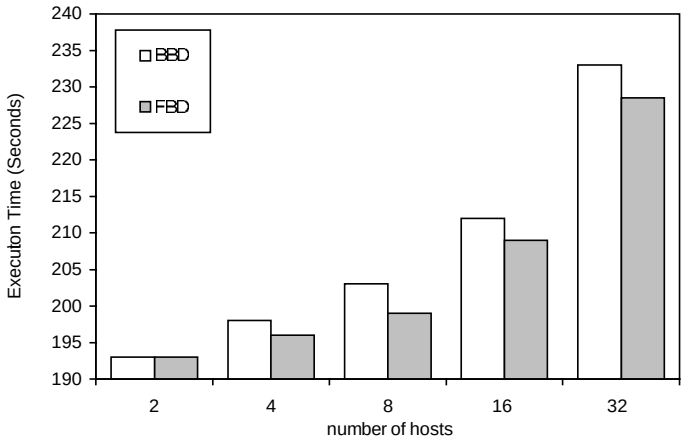


Fig. 12. Results of Reading 1 Kbytes Data from the 100 Mbytes XML File of Every Host.

5.2. Experiment 2: Route generation: Sequential robust model vs. FBD (II)

In this experiment, we compare the route generation time for models with one substitute route.

The complexity of sequential robust model is $O(n)$. The results shown in Fig. 9 illustrates that the difference in performance is not very significant. The robust sequential model can outperform this a bit in most cases. But when the there are 1024 addresses, the robust sequential model becomes inferior. With 2048 addresses, the program of robust sequential model runs out of memory after running several hours. The reason is the same as mentioned in experiment 1. For FBD (III), we tested it with up to 2048 addresses. The time was 4277 seconds.

5.3. Experiment 3: Sequential migration vs. binary dispatch

In this experiment, we tested up to 32 hosts to compare the migration and dispatch time of different models neglecting any robustness mechanism. In Fig. 10, in the implementation, a mobile agent will not access any local data so that the measured time is used for migration or dispatch only. In order to obtain independent result each time, we rebooted the Tahiti server to prevent the affect from the cache.

When the number of visited hosts is no more than 8, the differences in performance were not significant. With the increase in the number of hosts, the migration time of any sequential migration model increases very fast. In comparison, the dispatch time for binary dispatch model increases fairly slowly.

Meanwhile, the migration time for sequential robust model is always shorter than the sequential model since in the sequential robust model, two mobile agents are dispatched and each one only visits $n/2$ hosts. Nevertheless, its performance is not comparable to the binary dispatch model. For the two binary dispatch models, since no time for local data access is measured, their performances are almost the same.

The performances of two binary dispatch models are also compared when the time for local data access is measured. The size of read local XML documents is set to 1 Mbytes and 100 Mbytes. The returned data set size is set to 1 Kbytes and 100 Kbytes.

From the data illustrated in Figs 11–14, we could observe that when the data set size of the XML document and the result size are small (e.g., 1 Mbytes and 1 Kbytes respectively in Fig. 11), the performance difference is not significant.

But when the XML document size is large (e.g., 1 Mbytes) but the result size is small (e.g., 100 Kbytes), the performance difference is the most significant. In this case, the overall time is relatively short. FBD

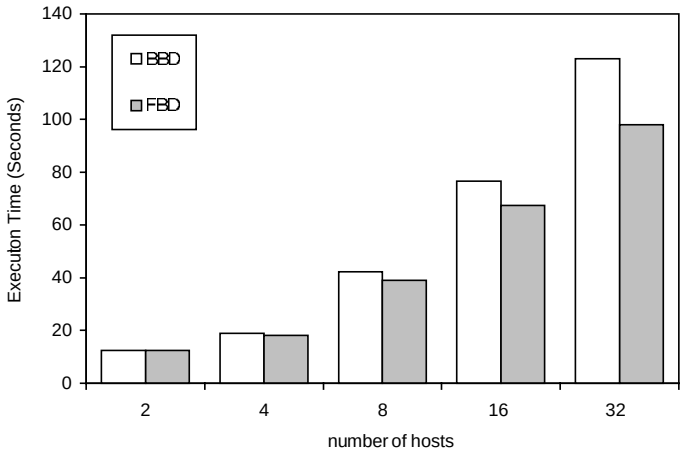


Fig. 13. Results of Reading 100 Kbytes Data from the 1 Mbytes XML File of Every Host.

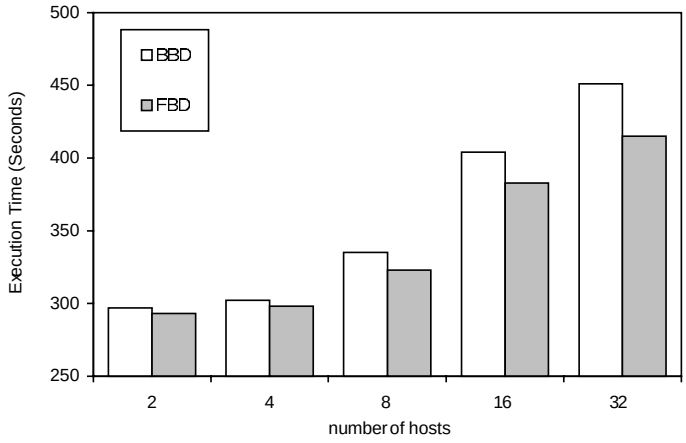


Fig. 14. Results of Reading 100 Kbytes Data from the 100 Mbytes XML File of Every Host.

model can avoid the congestion at the side of the master agent. When there are 32 hosts (see Fig. 11), FBD model can obtain 20.3% saving percentage. The performance difference becomes more and more significant with the increase of the number of hosts.

6. Conclusions

This paper presents several secure route structures and corresponding dispatch protocols based on a fast binary dispatch model ensuring both security and efficiency. They expose only minimal addresses to a host to perform dispatches. With the improvement of security performances, the computational overhead for route generation may also increase. However, with respect to security, which is the most important issue for mobile agents, the sacrifice on performance is worthy, while the dispatch complexity remains $O(\log_2 n)$.

For practical applications, mobile agents having the same type tasks and having physically close destinations can be put in the same group encapsulated with pre-encrypted routes. For verifying the

integrity of an incoming agent, the pure code can be included in the signature of a route after being hashed to a fixed length (e.g. 128 bits by MD5 algorithm) when it is generated at the home host. And the length of the signature remains unchanged.

References

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

[2] M. Baker, R. Buyya and D. Laforenza, Grids and Grid Technologies for Wide-Area Distributed Computing, *International Journal of Software: Practice and Experience* **32** (2002), Wiley Press, USA.

[3] CCITT Recommendation X. 509-1989, The Directory-Authentication Framework. Consultation Committee, International Telephone and Telegraph, International Telecommunication Union, Geneva, 1989.

[4] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and 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** (2001), 187–200.

[5] I. Foster, The Grid: A New Infrastructure for 21st Century Science, *Physics Today* **55** (2002), 42–47.

[6] I. Foster and C. Kesselman, *Computational Grids*, Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufman, 1999.

[7] I. Foster, C. Kesselman, J. Nick and 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.

[8] I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, *A Security Architecture for Computational Grids*, Proc. 5th ACM Conference on Computer and Communications Security Conference, 1998, 83–92.

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

[10] P. Kotzanikolaou, M. Burmester and V. Chrissikopoulos, *Secure Transactions with Mobile Agents in Hostile Environments*, ACISP 2000, LNCS 1841, Springer-Verlag, 2000, 289–297.

[11] D.B. Lange and M. Oshima, Mobile Agents with Java: The Aglet API, appears, in: *Mobility: Process, Computers, and Agents*, D. Milojicic, F. Douglis and R. Wheeler, eds, Addison-Wesley Press, Reading, Massachusetts, USA, 1999, pp. 495–512.

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

[13] 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 (PRICAI00), 2000, 101–114.

[14] A. Menezes, P. Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[15] E. Palmer, *An Introduction to Citadel-a Secure Crypto Coprocessor for Workstations*, in Proceedings of IFIP SEC'94, 1994.

[16] S. Papastavrou, G. Samaras and E. Pitoura, Mobile agents for World Wide Web distributed database access, *IEEE Transactions on Knowledge and Data Engineering* **12** (2000), 802–820.

[17] R.L. Rivest, A. Shamir and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM* (21) (1978), 120–126.

[18] A. Romao and M.M. Sliva, *Secure Mobile Agent Digital Signatures with Proxy Certificates*, E-Commerce Agents, LNAI 2033, Springer-Verlag, 2001, 206–220.

[19] T. Sander and C.F. Tschdin, Protecting Mobile Agents Against Malicious Hosts, Mobile Agents and Security, *LNCS* **1419** (1998), Springer-Verlag, 44–60.

[20] V. Varadharajan, *Security Enhanced Mobile Agents*, in Proceedings of the 7th ACM conference on Computer and Communications Security, 2000, 200–209.

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

[22] 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, 57–71.

[23] U.G. Wilhelm, *Cryptographically Protected Objects*, Technical Report, Ecole Polytechnique Federale de Lausanne, Switzerland, 1997.

[24] Y. Wang and X. Pang, Security and Robustness Enhanced Route Structures for Mobile Agents, *Mobile Networks and Applications* **8** (2003), 413–423.

[25] Y. Wang and K.L. Tan, A Secure Model for the Parallel Dispatch of Mobile Agents, Proc. of Third International Conference on Information and Communications Security (ICICS2001), Springer-Verlag, *LNCS* **2229** (2001), 386–397.

[26] Y. Wang, K.L. Tan and X. Pang, *Several Structures for Protecting the Routes of Mobile Agents Dispatched in Parallel*, in Proceedings of the 5th International Workshop on Mobility in Databases and Distributed Systems, 2002, 61–68.

[27] 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: Internet and Web Information Systems* **5** (2002), 41–66.

Yan Wang is currently a lecturer in the Department of Computing, Macquarie University, Australia. He is also a member of Information and Networked System Security Research. He received his Doctorate Degree of Engineering in computer science and technology in 1996 from the Harbin Institute of Technology (HIT), P.R. China. He was a visiting scholar at the Department of Computer Science, City University of Hong Kong in 1997 and 1999 respectively. From Nov. 1999 to June 2003, he was a Postdoctoral Fellow/Research Fellow in the Department of Computer Science, School of Computing, National University of Singapore (NUS). He is a member of IEEE. He has published a number of research papers in international journals and conferences. His research interests cover mobile agent, computer security, trustworthy computing and electronic commerce.

Vijay Varadharajan is the Microsoft Chair and Professor of Computing at Macquarie University. He is also the Director of Information and Networked System Security Research. He is also the Technical Board Director of Computer Science, Australian Computer Society. Previous to this, he was the Foundation Chair Professor and Head of School of School of Computing and IT at UWS Nepean. Prior to taking up this appointment, he was responsible for worldwide Security Research at Corporate Hewlett-Packard Labs based at HP Labs Europe in Bristol, UK, for a number of years. He has published more than 220 papers in International Journals and Conferences and has co-authored and edited 8 books on Security, Networks and Distributed Systems. His current research interests are in distributed system security, network security, secure electronic commerce, mobile agent security and trustworthy computing. He is on the Editorial Board of several journals including the ACM Transactions on Information System Security and The International Journal of Information Security (Springer-Verlag). He has been a member of the Board of Advisors in Trusted Computing Platform Association (TCPA) (USA) and is on the Microsoft Trustworthy Computing Advisory Board (USA). He is a Fellow of the British Computer Society (FBCS), a Fellow of the IEE (FIEE), a Fellow of the IMA (FIMA), a Fellow of the Australian Institute of Engineers (FIEAust), a Fellow of the Australian Computer Society and Senior Member of IEEE.