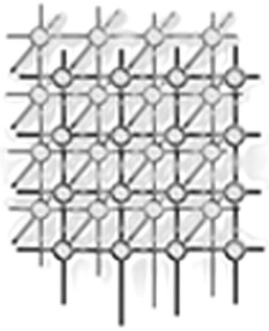
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2005; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*



G-PASS: An Instance-oriented Security Infrastructure for Grid Travelers[‡]

Tianchi Ma ,Lin Chen ,Cho-Li Wang ,Francis C. M. Lau[†]

Department of Computer Science
The University of Hong Kong, Hong Kong

SUMMARY

Grid computing unifies distributed resources via its support for the creation and use of virtual organizations (VOs), where a VO represents a collection of distributed resources to be accessed through predefined resource sharing and coordination policies. We consider a special type of mobile processes, named Grid travelers, which can travel across boundaries of virtual organizations for the detection of resource availability, to negotiate for the approval of access privileges, and to conduct remote execution. A new security infrastructure named G-PASS is proposed to guarantee the validity and integrity of the travelers and the critical security knowledge they collect while traveling, especially while crossing some VOs. G-PASS borrows the idea of passport and custom, as well as the procedures for people's travel in real life, to provide role-based delegation mapping and access control. We demonstrate the power and feasibility of G-PASS with a simulated mobile agent environment and a distributed ray-tracing application running on multiple VOs. Various security overheads coming from migration decisions and actual agent or process migration are reported. G-PASS can be installed with GSI as the base, which makes it compatible with the existing Grid middleware.

KEY WORDS: security, delegation, trust, mobile agent, process migration

*Correspondence to: Tianchi Ma, Systems Research Group, Department of Computer Science, The University of Hong Kong, Hong Kong
[†]E-mail: tcma@cs.hku.hk
[‡]This research is supported in part by HKU Foundation Seed Grant 28506002, the China 863 National Grid project, and a HKU grant for the HKU Grid Point.

Received 15 March 2005
Revised 19 March 2005

1. INTRODUCTION

In grid computing [1], a virtual organization (VO) consists of a group of individuals or institutions who share some computing resources which can be used to achieve a common goal. All members in a VO follow predefined policies for resource sharing and coordination. Since its emergence, grid has achieved a remarkable union of distributed resources such as CPU cycles, storages, sensors, data and software via the creation and use of virtual organizations.

As grid technologies have become mature and pervasive, more and more VOs are formed and various types of resource and information sharing are possible. It is envisioned that the future development of grid applications will be more dynamic and complex, as these applications need to access miscellaneous services and resources that are scattered among different VOs.

However, existing grids impose certain restrictions on resource sharing and accesses. First, as a VO is established based on predefined policies and identities, the identity space of the VO is completely closed and the policy for resource access cannot be adjusted during runtime. Thus, it is not possible for a complex grid application to dynamically generate requests to access resources in multiple VOs. A possible solution is to break up the application into several parts, each carrying a specific task and being assigned to a different VO to execute. At runtime, a special mobile agent bearing the identity of the application will traverse the grid nodes of the potential VOs for on-line resource discovery and access rights negotiation.

Given the mobility and VO-crossing property of these special mobile agents, strong protection is required to guarantee the security of these special mobile agents. Current security infrastructures for grid, such as GSI (Grid Security Infrastructure) [2], PMI[12], and PRIMA[11], define trust models that can meet the requirements of today's grid applications. They are, however, not capable or flexible enough to adapt to dynamic grid environments and the increasingly more complicated grid applications. In particular, the delegation mechanism lacks support in two aspects:

Role-based authorization Each VO has its own local identity and privilege space. The special mobile agent may travel to multiple hosts in name of different VOs. To support VO-crossing, there should be a mechanism for dynamic privilege mapping, which is usually called role-based authorization [3]. That is, an identity should be dynamically assigned for a new role that replaces its current role. As we decompose the application into several mobile code segments, several roles may be necessary for a single grid application to function. Each of them may be granted (approved) only partial privileges to access resources at a grid node. This complicates the algorithm design. The approval "carrier" must be well designed, in order to avoid any confusion in role-based mapping and compliance checking. From the security point of view, all the approvals should be well protected to prevent malicious amplification or reduction of the approved privileges during execution.

Security knowledge collection During the special mobile agent's trip, one cannot be sure about the reliability of the visited hosts, nor that the mobile agent has not caused any harm to any of the hosts. Nevertheless, once any misbehavior is detected, corresponding measures should be quickly activated to locate the source of the damage and adjust the trust relationship automatically so that the occurrence of similar accidents can be

prevented. More knowledge and information on the distributed trust states and the mobile agent's past experience are required for establishing such measures and for making security decisions.

In this paper, we propose a new type of mobile agent, called *grid traveler*, which has the special ability to move across VO boundaries to coordinate the use of resources and access control in different protection domains. An accompanying security infrastructure named G-PASS is proposed for the credential management of grid travelers. G-PASS provides two useful functions: (1) It implements a new trust model for supporting simple credential verification and transfer, as well as the creation and atomicity of security transactions. At the core of this trust model is the concept of "security instance". A security instance includes a security transaction and its constraint specification. One can accomplish the delegation by binding his/her identity onto the security instance instead of onto some special host. (2) G-PASS supports VO crossing via an RBAC2 [3] qualified role-based privilege mapping with the granularity of a security instance. Different from traditional role-based access control (RBAC), a gateway service called G-custom is imported to map the original credentials (recorded in a credential carrier called G-passport) to a locally recognized approval table. The local resource publisher can then work with the normal access control directly without having to install the role-mapping mechanism.

The rest of the paper is organized as follows. Section 2 gives the background of this research. In Section 3, an overview of G-PASS and its features are given. The instance-oriented trust model and role-based privilege mapping are discussed in Section 4. Section 5 presents the performance test of G-PASS. Sections 6 and 7 discuss related work and conclude the paper.

2. Background

GSI (Grid Security Infrastructure) is the security system generally used in current grids. It maintains basic trust relationships for resource sharing and job submission. GSI, however, cannot satisfy the complex security requirements of grid travelers during VO crossings.

Firstly, GSI is too simple to deal with grid travelers. In general, each VO has its own security policy space. There is a set of identity bindings on access rights. A delegation issued from outside of the policy space will be regarded as an invalid request because the identity binding on it cannot be recognized by the local access control policy. Although GSI has implemented a simple role-based mapping mechanism, it is just a simple one-to-one mapping and can hardly deal with complex situations when crossing VOs. For example, a traveler that assists in the execution of a complicated grid application, may need to have multiple identities. Upon entering a VO, an identity is used to validate the application's role in the corresponding VO and to approve its resource usage. As an application may carry multiple identities and there are different resource access rules to follow in each VO, the one-to-one mapping cannot support such complex role mapping relationships.

Secondly, GSI uses the X.509 delegation model and the delegation is bound to the target host. As the grid traveler travels over from hop to hop, the delegation will be transferred by continuously issuing new delegation documents. The verifier at a grid entry point has to check

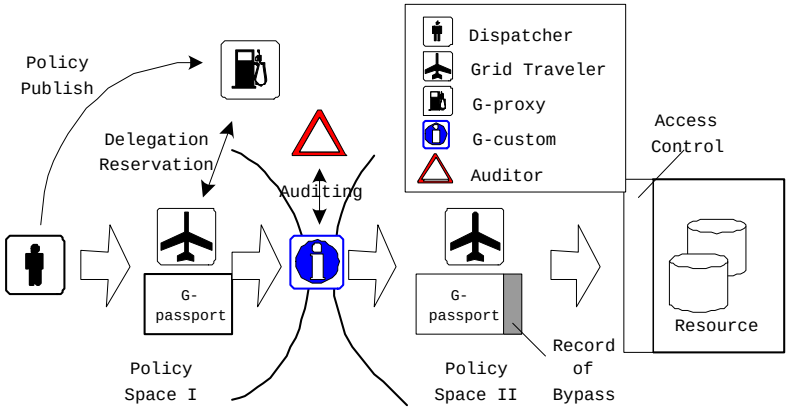


Figure 1. G-PASS Architecture

all the signatures created in the delegation chain, thus introducing large overheads, and the whole security system would become not so scalable.

Thirdly, the atomicity of security transactions cannot be ensured in GSI. For example, a modification to a bank account will include a read and a write operation. There is no safe state between the two operations. If the two operations are approved by two identities, it is difficult to decide which operation should take the responsibility upon the occurrence of an exception during the modification—the "separation of duty" problem. GSI allows only simple approvals from single identity. It can not deal with the separation of duty problem well.

3. System Architecture

Figure 1 shows the basic working mechanism of G-PASS. G-PASS consists of various security-related components to support grid travelers.

The *G-passport* resembles a passport in real life with continuous passport pages. The G-passport keeps several types of page content: (1) *G-dispatch* declares the delegation from the traveler's dispatcher. It records a privilege set asserted to be legal by the original dispatcher. (2) *G-warrant* specifies the intent of warranting a subset of the privileges declared in G-dispatch by a certain warrantor. At the same time, the warrantor promises to be responsible for the traveler's corresponding behavior. (3) *G-exception* records the security exceptions thrown by the hosts or resource access controllers. It is used for security monitoring. Systems can adjust their policy according to such records. (4) *G-event* records the user-defined security events.

With the G-passport, we design an instance-oriented trust model [4], which ensures the atomicity of security transactions. A chained signature technology [4] is adopted to ensure the

integrity of the G-passport. Thus, the attacker cannot find a way to substitute, modify, or delete a page in the G-passport, nor to insert a page. Each page is defined as a contract, in which at least two identities are required to provide a digital signature, and to claim their responsibilities. This complies with Clark-Wilson's principle of separation of duty [6]. An auditor, if it exists, may be required to reserve a copy of the contract and to accuse any concerned identity who attempts to deny the approval or the event.

G-custom is a border checker at the entry of a policy space. The role-based mapping is performed in G-custom. *G-proxy* grants new delegation in the name of the user. It enables single sign-on by the user and keeps his/her policy active even the user is currently off-line. This procedure is called *delegation reservation.*

4. Trust Management

4.1. Instance-oriented Trust Model

The traditional trust model sits on top of the Public Key Infrastructure (PKI). Suppose that user U holds the keypair $KP = (Pk, Sk)$, where Pk is the public key and Sk the private key. A digital signature $S_{KP}(v)$ proves the correctness of statement v in U's name. We define the delegation

$$Del(U^{'}, p, U) = ((U^{'} \parallel p), S_{KP}(U^{'} \parallel p), Pk),\ p = \{r_i, c_i)|i = 1, \ldots, n\},\ n \geq 1$$

to claim that $U^{'}$ can have privileges in the set p in the name of U, where r_i represents the detailed privilege and c_i is the constraint of r_i.

From the above, we can see that the traditional delegation is identity-oriented. During the migration of a grid traveler, the target identity is the identity of the target host. So the delegation is also a host-oriented one. It has the following disadvantages for supporting grid travelers. Firstly, it fails to achieve separation of duty. The Clark-Wilson's principle stipulates that the states before and after the transaction must be safe and verifiable, so that the responsibility is clear once any exception is raised. It is the issuer of the delegation who can define transactions; the privileges, however, are defined by the resource provider. So a new delegation mechanism should be developed to enforce the recording of privileges in the form of transactions. Secondly, it incurs a large overhead. Suppose that a grid traveler obtains a delegation of p_1 from identity U_1; and after $k-1$ times of migration, it arrives at the host with identity U_k. A delegation chain is generated during the trip. The host will need to check at least $k-1$ signatures to assert the validity of this delegation document.

To overcome the above drawbacks, we adopt an instance-oriented approach [4]. A security instance includes a security transaction and its validity specifications. Identities will be simply delegated to the transaction instead of the privilege operations. This provides the atomicity of security transactions by which the separation of duty can be achieved.

Let $T(r_1, \ldots, r_k)$ be a transaction including a sequence of k operations $(o_1, \ldots,$
$o_k)$, where the operation o_i is performed according to the defined privilege r_i, for $1 \leq i \leq k$. During delegation granting, the issuer can specify the constraint set C for the transaction. Let $Ins(T, C) = (\{r_1 \ldots r_k\}, C)$ be a security instance of $T(r_1, \ldots, r_k)$ under C. Let $req(r, S)$

represent a request for operating p under the system state S. When it satisfies $r \in \{r_1 \dots r_k\}$ and $S \in C$, the request is said to be covered by the instance $Ins(T, C)$.

When U wants to grant delegation to $Ins(T, C)$ with keypair $KP = (Pk, Sk)$, it can simply issue a capability, which is a signed document with permitted privileges to serve as the delegation of the instance; that is,

$$Del(Ins(T, C), U) = (Ins(T, C), S_{KP}(Ins(T, C)), Pk)$$

Note the target identity in the traditional host-oriented delegation has been removed from the new delegation document. Thus there need not be a delegation chain to implement the one-by-one security guarantee, and the overhead in verifying the delegation can be greatly reduced from $k - 1$ to 1.

The goal of the instance-oriented delegation is mainly for giving some convenience to the security designers, and providing a more flexible and stable protocol to carry the security polices. The most remarkable characteristic of the instance-oriented model is its mobility support. In conventional host-oriented delegation, as mentioned in Section 2, everything will be re-established upon the traveler's move, hence leading to inflexibility and abuse of delegation. With the support of host-absence binding, the traveler is allowed to move everywhere subject to the predefined conditions.

Besides, the existence of instances can also achieve an atomic assurance on authorizations and delegations of complex operations. Suppose there is a database updating operation in which two operations, namely read and write-back, are involved. This operation will be regarded normally as a transaction and thus cannot be divided. Assume one identity has delegated its name to the read operation, while another identity has delegated to the write-back. There will be responsibility confusion once some misbehavior is detected during this operation. Therefore we encapsulate the whole operation in a security instance. The instance will be delegated entirely or not at all. Thus the atomicity is guaranteed implicitly in the definition of security instances and policy establishment. The detail of designing and implementing the instance-oriented model is discussed in [4] [5]. With the atomic assurance, the complexity of role-based delegation mapping and compliance checking can be greatly reduced since it provides larger granularity in privilege definition as well as avoids the detection of potential delegation conflicts.

4.2. Contractual Model and Stamps

Stamps serve as the proof of past events and authorizations. A contractual model is used to realize the stamps. In the real world, people use contracts to record and prove some agreements that had been established, and to verify whether they are executed correctly. In the G-passport, contractual stamps are adopted for recording and proving some approvals or events, as well as protecting them from being denied. Usually, a real-life contract should include signatures of the people concerned. Similarly, all the identities that have direct relation with the to-be-proved approval or event are required to digitally sign on the stamp, in order to make a trustable assurance. The requirement of significant knowledge collection and protection mentioned in Section 1 can be achieved by importing and enforcing stamps on the G-passport. Figure 2 shows a sample G-passport.

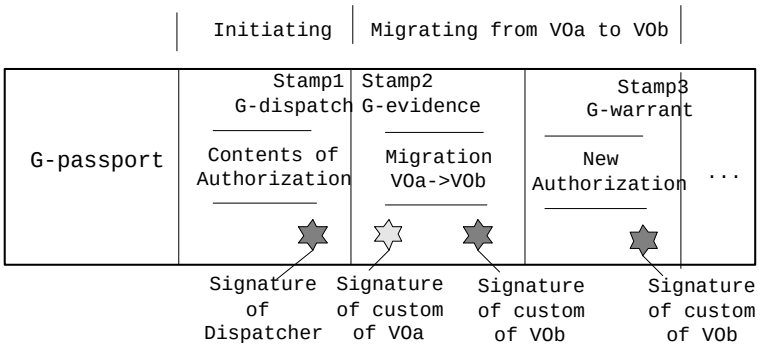


Figure 2. A sample G-passport with two recorded events: Initiation and Migration from VOa to VOb

The contractual model imports several functions and has the following benefits:

Validate-ability A third party can check and testify the past businesses by verifying the stamps. No one can deny the content of the contract, because all the concerned hosts have signed on it.

Accountability It is enforced that all the concern identities who have signed on the contract should reserve a copy of the signed contract. This is for assuring when one or some signatories (not all) try to deny the past, another contract holder can stand out to impeach the denial.

Integrity Stamps are chain-linked in the G-passport with proved double-direction pointers. The system can quickly detect that if one or more stamps have been disguised between the two given ones.

We introduce a protocol based on a bidirectional-linked contract list to be used to protect the integrity of the chain-linked stamps. The structure of this protocol is partially derived from that for traceable X.509 proxy certificates [9]. The criteria of this protocol are listed below:

1. The agent must travel with a document called *bidirectional-linked contract list*.
2. A *contract* is an event-recording document on which one *Initiator* and one *Acceptor* are required to digitally sign. Each of them will reserve a copy of the contract. The contract can be shown to any one as an evidence proving the historical existence of the recorded event. The Initiator and the Acceptor should be responsible for the correctness of the recorded event.
3. On agent migration, a new *stamp* should be generated and appended to the tail of the contract list. Each stamp will have an incremental ID number. A stamp is a contract while its Initiator is the source host of the migration and its Acceptor should be the target host.

4. A bidirectional-linked contract list with length n is valid if and only if for all $1 < i < n$, the source host of stamp i is equal to the target host of stamp $i-1$.
5. Once a new approval is generated or consumed (by resource access), a new *trace record* must be generated and pasted onto the stamp recording the next migration of the agent. Each trace record is also a contract.
   (a) On getting approval, the Initiator will be the warrantor, and the Acceptor will be the current host of the agent.
   (b) On consuming an approval, the Initiator will be the resource provider, and the Acceptor will be the current host of the agent.
6. Suppose an agent moves from host $i-1$ to host i. Once the agent wants to apply for a warrant approval from host i, it should contact host $i-1$ for a signed *ticket*, proving that host $i-1$ knows about this event. The agent then sends its warrant request to the warrantor, attaching the ticket and the stamp recording the agent's migration from host $i-1$ to host i. After checking all these contracts, the warrantor can issue and sign on the approval contract. Similarly for resource consumption. The agent also needs to show such a ticket from its previous host.
7. Once the agent is about to move from host i to host $i+1$, it will require host $i-1$ to issue a *stamp certificate* recording all the events during the time that the agent is at host i. Also the certificate needs to record the name of host $i+1$. Then a new stamp will be generated on host $i+1$ by pasting all the certified trace records onto it.
8. If the agent is about to move from host $i-1$ to host i, and then to host $i+1$, where host $i-1$, i and $i+1$ are the same hosts as $m-1$, m and $m+1$ in the agent's historical path, the agent must require host $i-1$ to claim that this is the second occurrence of the $m-1 \longrightarrow m \longrightarrow m+1$ pattern, and record it in the stamp certificate. This is to prevent the system from *replay* attack (the host plays the agent again and again to disturb the normal state of the system).

This protocol can protect the document's content from being illegally modified, removed, inserted and forged. Also, as mentioned above, it can even prevent the replay attack.

4.3. Role-based Privilege Mapping

To support VO crossing, the security credentials should be effectively made in the target VO's local policy space. G-PASS imports a role-based privilege mapping, which can proceed in two phrases: (1) role-based privilege mapping, and (2) normal access control. Figure 3 shows the main operations performed in phase (1). The credentials recorded in the G-passport are transformed into a privilege table that can be fully recognized in the target VO's local policy space. The privilege table is formed with an array of instances. Each instance is approved by several locally recognizable roles. As an assistance of the transformation, a role table is imported in which roles and their corresponding global identities are recorded. The role table can be published via a G-custom service. In phase (2), because the role-based mapping has been done when the grid traveler entered the VO, the local resource provider need not perform RBAC again on the G-passport. It will firstly select an instance according to the given requests.

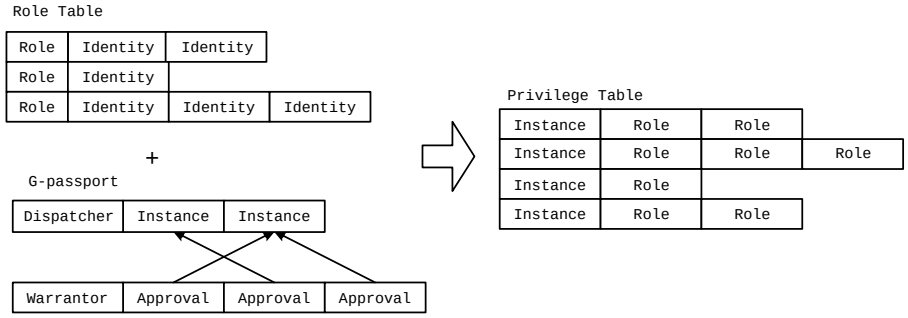


Figure 3. Role-based privilege mapping

Then it will check if there exists a role that is granted to use all the privileges recorded in the instance by the local access control policy.

The advantage of this two-phase procedure is that the local resource provider need not provide a role table themselves. This makes policy adjustment easier because no synchronization and consistency problem need to be considered.

5. Performance Evaluation

In this section, we evaluate the performance of G-PASS in two cases: (1) a large-scale mobile agent system on grid by simulation; (2) a parallel ray tracing program in a WAN-connected grid testbed.

5.1. Mobile Agents with G-PASS

We simulate 500 mobile agents traveling in a grid platform connected by the Internet. Suppose that the 500 mobile agents, $A[0..499]$, enter the randomly-connected network sequentially with the time interval of 50ms. There are two scenarios:

Single-task agent Each agent $A[i]$ has a randomly predefined path with $(i\%10)+1$ hops. The code of each agent is originally approved by its dispatcher (the first privilege granter). Having crossed a hop, the agent will try to access a local resource. Before each resource access, the agent is required to obtain an additional approval from a remote privilege granter. Assume there are totally five instances in the G-passport (for delegation, and different types of operations). In this scenario, the number of hops for the agents ranges from 2 to 11.

Multi-task agent Each agent $A[i]$ has a randomly predefined path with $(i\%30)/4+1$ tasks. After each hop, the agent will also try to access a local resource. Different from the single-task scenario, all the resource access operations within the same task should be approved by the same remote privilege granter. In this scenario, the number of hops for the agents varies from 2 to 30, and the number of instances in the G-passport is set to $hops/3$.

Suppose that the requirement of our application is that the agent's path and historical resource access events should be traceable, and the tracing information should be referred in the authorization decisions. Below are the protocols of the two trust models under this requirement.

Protocol of G-PASS When an agent is dispatched, it brings along the necessary instances for all possible operation types. For each resource access (in the single-task scenario) or at task start (multi-task scenario), the agent will send its corresponding instance and trace-list in a warrant request to the specific remote warrantor. The warrantor will approve on the request and return it to the agent after it has checked the instance. The agent can then submit the approved instance to the resource provider to get authorized for the required operation. A handover event will be added to the trace-list for each migration.

Protocol of delegation When the agent is dispatched, it will ask all the delegators to sign a delegation certificate for all the possible operations. The agent will then travel with all these certificates. Upon each resource access (in the single-task scenario) or at task start (multi-task scenario), the agent will simply show the delegation certificate list to the resource provider to get authorized for the required operation. At migration time, an additional certificate will be attached signifying that the current host further delegates all the privileges to the next host.

We used a Pentium III 500MHz dual-CPU server with 1GB memory to emulate the tasks' execution. The emulator is a single-thread process without networking ability. It simply records down the actual execution time of all non-network operations (called CPU overhead) and also the network traffic required in bytes. All these information will be dumped into a report file. Figure 4 shows the overheads under the single-task mode, and figure 5 shows the multi-task mode. From the figures we can see the different slopes of using G-PASS and delegation, which show G-PASS's superiority in scalability. Also we have noticed that the slopes of both curves in Figure 5 (multi-task agents) are increasing. This is because the instance number in the multi-task mode is set to be variable, which causes the complexity of generating the G-passport to increase non-linearly.

5.2. Parallel Ray Tracing on Grid

In this experiment, we execute a distributed ray tracing application in the Hong Kong Grid (HKGrid) testbed [17]. The testbed currently composes of five sites in four universities in Hong Kong. In each grid point, there are a group of Linux machines connected to a gatekeeper machine running G-PASS and the Globus middleware. These grid sites are interconnected by the Hong Kong Academic and Research NETwork (HARNET).

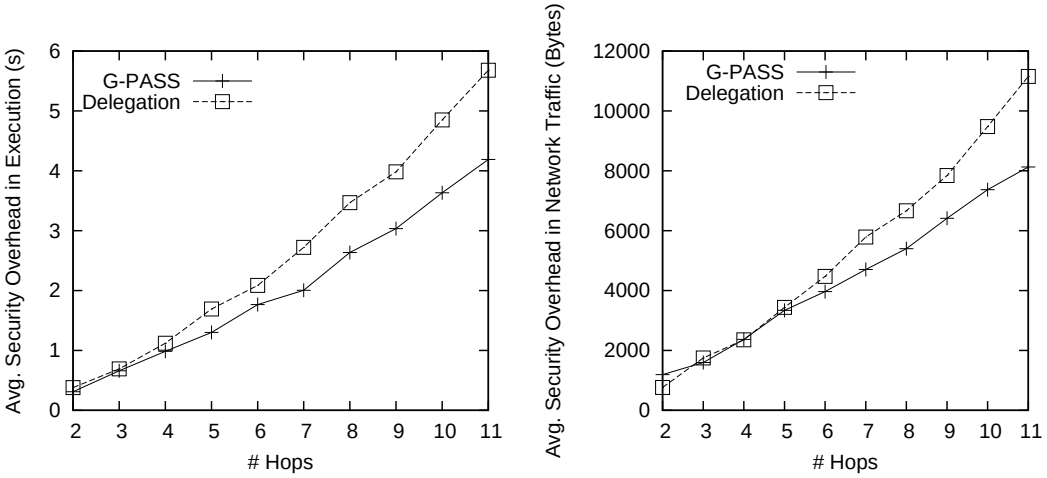


Figure 4. Comparison of average security overheads for single-task agents

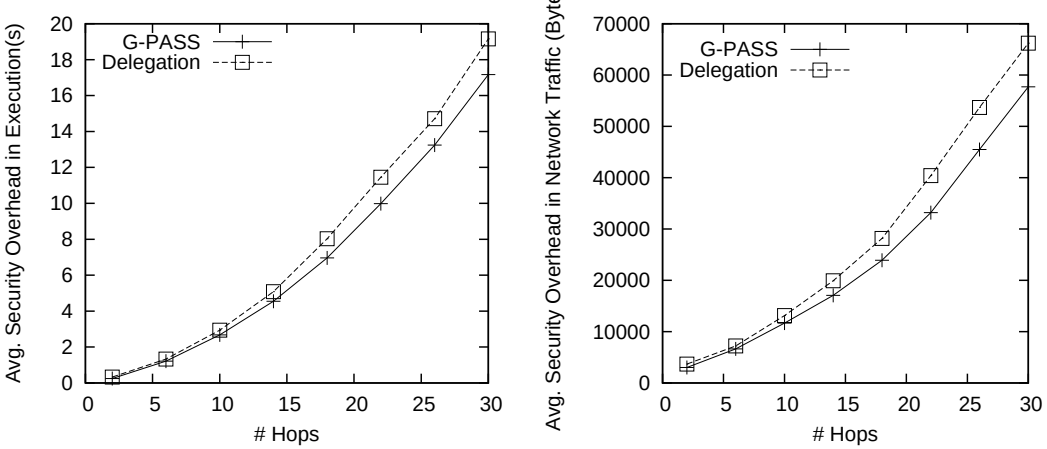


Figure 5. Comparison of average security overheads for multi-task agents

The ray tracing program was implemented using G-JavaMPI [10], which is an MPI middleware that supports parallel and distributed Java computing in a grid environment. A special feature of G-JavaMPI is its support of transparent Java process migration which can facilitate dynamic load sharing and resource-driven task migrations. The migratable JavaMPI process in G-JavaMPI is regarded as a grid traveler for which the G-PASS mechanisms can provide security protection when migration across VOs happens.

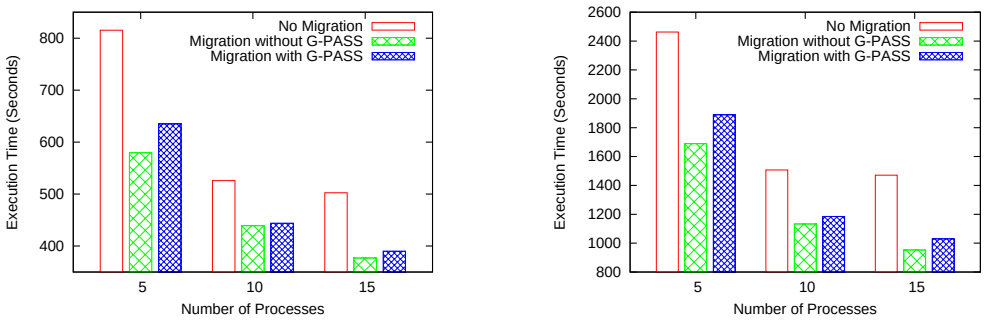


Figure 6. Comparison of execution times of ray tracing for Image A (left) and B (right)

We execute the ray tracing application with two different image sizes and we scale the number of processes. The performance difference has been observed under the conditions of no migration, migration without G-PASS and migration with GPASS. We partition the image into equal-sized image blocks, each being assigned to a different process. The processes are shuffled and distributed to the HKGrid sites for computation. During execution, the process migrations are decided by a simple but efficient load sharing strategy called WorkStealing. When a site finishes all its processes and becomes idle, the decision procedure for migration is invoked immediately. If there are several sites that are idle, the site with the largest CPU capacity is chosen as a destination host for the migration act. However, if the CPU capacity of the destination host is less than or equal to that of the source host and there is only one unfinished process in the source host, no process migration will take place.

Figure 6 presents the execution times of ray tracing for two images, A and B. Image B has a larger problem size than image A. There are two reasons that will trigger the process migration (and thus activate G-PASS): (1) the five grid nodes have different CPU capacities, so that the processes in the faster nodes will finish earlier; (2) the image blocks (of equal size) may create uneven workload because of their different contents.

As the figures show, with the support of process migration, the execution times are shortened very much because of better utilization of idle resources. In image A, comparing the execution time with migration and G-PASS enabled to that with migration disabled, process migration and G-PASS improve the executions by 15.5% $\sim$ 30%. The average improvement over all six testing cases for both image A and B is 22.4%.

The G-PASS operations introduce some additional overhead to the migration procedures, and therefore the execution times with G-PASS enabled are larger than those without G-PASS. The execution times are extended by 3.5% $\sim$ 11.8% in the six testing cases. The average increase of the execution times is 6.4%. Based on the WorkStealing strategy, the number of migrations observed ranges from 2 to 5. The increase ratio depends on the number of migrations.

When we increase the number of processes from 5 to 15, the average improvement of all 6 cases is 40.2%. By using more processes, the initial workload distribution is more even. The migration overhead becomes less as the problem size associated with each process is reduced. Moreover, the workload can be balanced in finer granularity. Therefore, performance improvement is observable.

The ray tracing application demonstrates the feasibility of using G-PASS on grid middleware, G-JavaMPI in this case. The experiments show that the execution performance can be improved by the efficient process migration mechanism and load sharing strategy. The security of process migration across the grid can be guaranteed by G-PASS without introducing significant overhead.

6. Related Work

In [8], a Community Authorization Server (CAS) is proposed to issue delegation capabilities. However, CAS is a centralized server that is pre-authorized by the resource provider. In G-PASS, the capabilities can be issued by the user, which do not have to be recognized by the resource provider. Indeed, it is the role-based mapping mechanism that makes this more efficient access control possible. Warrantors are allowed to issue their approvals to part of the capabilities in a distributed manner. Therefore G-PASS is more flexible than CAS and is more suitable for grid travelers.

Akenti [16] has provided a multi-stakeholder, pull-sequenced mechanism for grid-based resource authorizations. It merges policy tokens from all the stakeholders together, and provides central authorization. The policy tokens are self-proved, and hence can be securely transferred to everywhere. Also, indentures specifying the condition of the authorization are attached to the token. Akenti shows a good example of decoupling the policy enforcement point and policy determination point. The virtual custom feature of G-PASS may also require multiple stakeholders inside the administrative domain to update the role table cooperatively. Anchor [14] is a mobile agent platform supported by Akenti for the purpose of agent authorization. It requires all the hosts in an agent's history itinerary to record the agent's code and behavior. Once an agent wants to authenticate to a host, all these trace information are required to be inspected. The Anchor, however, has not provided the technique to protect these trace information from malicious insertion or removal.

In [9], an extensible delegation profile is proposed with support for host tracing and privilege shrinking. Host tracing can also be implemented in our G-passport by defining a handover event and enforcing the hosts to record it. Privilege shrinking can be implemented as another event in which the host declares that the delegation on some events is invalid from that moment on. However, extensible delegation cannot provide atomicity support on authorization, while such a support is provided by the G-passport.

PRIMA [11] achieves direct user-user delegation and fine-grained privilege granting by importing a privilege container. It allows a user to hold container-based privileges from multiple users, and the user can select some of them and bind to the resource request. The access control policy is also dynamically generated by merging all the privileges together. Actually, the privilege container is similar to the attribute certificate in X.509 v4. When it is transferred,

there is also a delegation chain that needs to be generated. The authorization of a job is simply a many-to-one mapping to an existing account or a dynamically generated account with permission control, but without any accountability issues being considered. PRIMA therefore is still not sufficiently equipped to support grid traveler.

A pull sequence, role-based access control is proposed in PERMIS [13], which adopts the Privilege Management Infrastructure (PMI) [12] defined in X.509 v4. It defines the role assignment in attribute certificates, by its own DTD language. Therefore, it will not need a role-table and the role-based policy can be dynamically generated by retrieving all the corresponding role assignments from a public LDAP service. This is good enough if the role-based policy is not updated frequently. In a scenario like ours where the security infrastructure is built on a trust model, an identity needs to be dynamically assigned with a proper role, according to its current credit. The attribute certificates can hardly adapt to this kind of frequent adjustment. Therefore, we still use a local role table.

Cardea [15] has provided an access control by dynamically evaluating authorization requests according to a set of relevant characteristics of the resource and requester rather than considering specific local identities. It facilitates resource usage reporting and dynamic session management together with authorization. However, it intends to collect the related information itself. This requires the stakeholder to be privileged and powerful in history information searching. A better mechanism, which we introduce in this paper, is to enforce the information collection in the protocol of the regular behavior of the mobile processes, and transfer that along with the processes.

7. Conclusion

In this paper, a security infrastructure called G-PASS is proposed. The goal is to support VO crossing and information gathering for grid travelers. G-PASS works as an infrastructure that provides protocols and documents (G-passport) as well as primary establishments (G-custom). It is compatible with the GSI in terms of key preparation and GRAM plug-in. Therefore, G-PASS can be used together with existing grid middleware, especially the Globus Toolkit. We envision that with more large-scale applications taking advantage of the grid environment, mobile travelers will be more common and demand more capabilities; and thus mobility support and role-based privilege mapping will be under the limelight in the grid security field.

REFERENCES

1. I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications* 2001; **15** (3):200–222.
2. 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.
3. R. S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman. Role-Based Access Control Models. *IEEE Computer* 1996; **29**(2): 38–47.
4. T. Ma, S. Li. An Instance-Oriented Security Mechanism in Grid-based Mobile Agent System. *IEEE International Conference on Cluster Computing* 2003; 492-495.

5. Tianchi Ma, Shanping Li. Instance-Oriented Delegation: A Solution for Providing Security to Grid-based Mobile Agent Middleware. *Journal of Zhejiang Univerisity Science*; Accepted in Apr. 2004, to be published.
6. D.D. Clark. and D.R. Wilson. Non Discretionary Controls Commercial Applications. *IEEE Symposium on Security and Privacy* 1997; 184-194.
7. S. F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol* 1990; 215:403-410.
8. L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. A Community Authorization Service for Group Collaboration. *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks* 2002.
9. S. Tuecke, et.al. Internet X.509 Public Key Infrastructure Proxy Certificate Profile. *IETF* 2003.
10. L. Chen, C.L. Wang, and F.C.M. Lau. A Grid Middleware for Distributed Java Computing with MPI Binding and Process Migration Supports. *Journal of Computer Science and Technology* 2003; **18**(4): 505–514.
11. M. Lorch, D. Kafura. Supporting Secure Ad-hoc User Collaboration in Grid Environments. *3rd International Workshop on Grid Computing, Baltimore* 2002.
12. D.W.Chadwick, A. Otenko. RBAC Policies in XML for X.509 Based Privilege Management. *Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002)* 2002; 39–53.
13. D. W. Chadwick, O. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. *Proceeding of the 7th ACM SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES (SACMAT 2002)* 2002.
14. S. Mudumbai, A. Essiari and W. Johnston. Anchor Toolkit (A Secure Mobile Agent System). *Technical paper* 1999.
15. R. Lepro. Cardea: Dynamic Access Control in Distributed Systems. *NASA Technical Report NAS-03-020* 2003.
16. M. Thompson, A. Essiari, S. Mudumbai. Certificate-based Authorization Policy in a PKI Environment. *ACM Transactions on Information and System Security (TISSEC), Volume 6, Issue 4* 2003. pp: 566-588
17. Hong Kong Grid. http://www.hkgrid.org