A Scalable Security Model for Enabling Dynamic Virtual Private Execution Infrastructures on the Internet

Pascale Vicat-Blanc Primet, Jean-Patrick Gelas, Olivier Mornard, Guilherme Koslovski, Vincent Roca, Lionel Giraud, Johan Montagnat, Tram Truong Huu

► To cite this version:

Pascale Vicat-Blanc Primet, Jean-Patrick Gelas, Olivier Mornard, Guilherme Koslovski, Vincent Roca, et al.. A Scalable Security Model for Enabling Dynamic Virtual Private Execution Infrastructures on the Internet. Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09), May 2009, Shanghai, China. pp.348-355, 2009, <10.1109/CCGRID.2009.76>. <hal-00677790>

HAL Id: hal-00677790
https://hal.archives-ouvertes.fr/hal-00677790

Submitted on 11 Mar 2012

A scalable security model for enabling Dynamic Virtual Private Execution Infrastructures on the Internet

Pascale Vicat-Blanc Primet, Jean-Patrick Gelas, Olivier Mornard, Guilherme Koslovski
INRIA - University of Lyon
Pascale.Primet@inria.fr

Vincent Roca, Lionel Giraud
INRIA
Vincent.Roca@inria.fr

Johan Montagnat, Tram Truong Huu
CNRS / I3S, Sophia Antipolis
johan@i3s.unice.fr

ABSTRACT

With the expansion and the convergence of computing and communication, the dynamic provisioning of customized processing and networking infrastructures as well as resource virtualization are appealing concepts and technologies. Therefore, new models and tools are needed to allow users to create, trust and exploit such on-demand virtual infrastructures within wide area distributed environments. This paper proposes to combine network and system virtualization with cryptographic identification and SPKI/HIP principles to help the user communities to build and share their own resource reservoirs. These ideas are implemented in the HIPerNet framework enabling the creation and the management of customized confined execution environments in a large scale context. Based on the example of biomedical applications, the paper focuses on the security model of the HIPerNet system and develops the key aspects of our distributed security approach. Then the paper discusses and illustrates how HIPerNet solutions fulfill the security requirements of applications through different scenarios.

Keywords

Execution Infrastructure as a service, resource virtualization, authorization, SPKI.

1. INTRODUCTION

Today, the usage of the Internet is fundamentally changing. The convergence of communication and computation portrays a new vision of the Internet. It is no longer ”only” a huge shared communication facility between edge hosts. Instead, it is becoming a world wide cloud increasingly embedding the computational and storage resources able to meet the requirements of emerging applications. This resulting vision of a global facility, that brings together distributed resources to build large-scale computing environments, recalls the promising vision of Grid computing enabling both data-intensive and computing-intensive applications. However, it is difficult to adapt it to sensitive application areas, such as biomedical applications, due to the stringent security requirements encountered (e.g., in terms of confidentiality, privacy, and content integrity) and their complex implementation in an open environment. The goals of this work are (i) to address data and processing security requirements of sensitive applications and (ii) to provide agile resources access control policies that can adapt to real-life application needs. This paper proposes a framework to create and manage confined Virtual Private eXecution Infrastructure, called VPXI, in a large scale distributed environment such as Internet. The key idea of this framework is to combine network and system virtualization with the cryptographic identification of (virtual) resources. Resources in a VPXI are virtualized to ease allocation and improve applications confinement. They are securely interconnected by a virtual private overlay network. The cryptographic identification scheme is the core of the Simple Public Key Infrastructure (SPKI), which defines a flexible private authorization management layer for the VPXI resources. The cryptographic identifiers are automatically translated to IP addresses thanks to a mapping layer such as the Host Identity Protocol (HIP), which decouples the locator/identifier roles of addresses. This paper focus and details how these last two components (SPKI and HIP) work together to provide the required security.

The rest of the paper is structured as follows. Section 2 outlines the challenging security requirements of the biomedical community and the shortcomings of traditional approaches to address these needs. In section 3, we illustrate how end users can benefit from customized private execution infrastructures. We then develop our combined network and system virtualization approach embedded in the HIPerNet software. In section 4 we detail the security model and present our use of the SPKI and HIP protocols. Section 5 discusses the system implementation and reports experiments on a real-scale testbed using a medical image analysis application. Section 6 discusses related works. Finally conclusions and perspectives are developed in section 7.

2. SECURITY REQUIREMENTS

This section focuses on the security requirements, taking as a representative example the case of medical applications. Then it reviews the techniques found in classical Grid systems and discusses how they match (or not) the above requirements.

This work has been funded by the ANR CIS HIPCAL grant (contract ANR06-CIS-005), the French ministry of Education and Research, INRIA, and CNRS, via ACI GRID’s Grid’5000 project and Aladdin ADT.

2.1 Medical data analysis security requirements

The medical imaging community faces several challenges making it one of the most constraining community for distributed infrastructure: a) the amount of data to process (tens of TB of data yearly); b) the distribution of data sources over the territory); c) the heterogeneity of data; and d) the confidentiality of medical data to preserve patients privacy. Moreover, the security requirements can not be tempered with, at the risk of discarding distributed computing usage in this area. All data belongs to patients whose confidentiality needs to be preserved in order to fulfill strict hospital privacy rules, in particular when data is transported from acquisition sources to processing sites and for storage of intermediate computing results. Secured communication channels alone are not sufficient to guarantee the protection of stored data and the associated metadata (minimally the file name which usually carries some meaningful information by itself, and possibly additional information). The data analysis process applied to data itself is sensitive as it is characterizing the kind of pathology a patient is affected by. Many scientific studies of medical data are performed using anonymized data sets, and on local resources only to enforce these constraints. Yet, re-identification of data is important for clinical applications, and may even be an obligation enforced by clinical ethical committees (the potential benefits for a patient whose data is used in a scientific study should always be applicable to this person).

Additionally, it is not acceptable for any clinical institution that the access to its data resources be managed externally by a centralized organization. The access control policy should ensure that each health organization solely controls its own data. The access control technique should allow for reactive access control rules to be set-up in the context of medical studies, whose life time is short (typically weeks) and the group composition highly dynamic (small specialist groups are involved in each study, possibly evolving along time to embark larger consortiums as needed by the experiments). Despite these strong constraints, Grids have been identified years ago as important tools to support various biomedical research efforts, including the processing of large medical databases, large-scale epidemiology, statistical study over populations, medical simulation or research on rare diseases.

To summarize, medical applications require i) **Protection** to ensure that data is not accessible to any outsider even though the execution infrastructure may physically expand to multiple institutions. In particular the data should neither be exposed during transfer, nor during on-disk storage; ii) **Privacy** to ensure that no outsider is able to track the data flow nor the computations applied for a particular patient; iii)**Adaptable and dynamic access control** to enable a fully customized access control policy, within a given institution or between several institutions;

2.2 Limit of classical Grid security approach

Grid infrastructures enable the federation of large-scale cross-institution user communities and resources Institutionally, user communities are identified as *Virtual Organizations* (VOs) composed of cross-institution computing resource users who collaborate in the context of a common scientific objective. Technically, VOs are implemented in Grids infrastructures through the security layer that comes at the very foundation of Grid middlewares. VOs implementation requires at least the authentication of cross-institution users. The "GLOBUS Security Infrastructure" (GSI) [8] for instance is an extension of the standard X509-certificate based on Public Key security Infrastructures (PKI). It has been extensively used in existing Grid infrastructures, including the very large scale EGEE European Grid [1] and OSG US Grid [2].

GSI provides fundamental security functionality such as user authentication and data transfer encryption. The authentication functionality is used to control the access to resources. The minimal requirement to access resources in a GSI-enabled Grid is to own a valid user certificate that has been delivered by a recognized *Certification Authority* (CA). Therefore, Grid access is controlled within a high level organization, a CA being typically operated at the national level. VO Management Servers (VOMS) have been introduced in order to provide a smaller scale, per-VO, access control management. Each VO server is managed independently by a local VO administrator.

These techniques, proposed in current Grid environments, feature several limitations. Relying on a global PKI plus a mix of global and local access control policies to make an authorization, they have major drawbacks in terms of scalability with respect to the number of participating domains. In particular, the burden required to reconfigure the shared environment (*e.g.* the addition or removal of a resource, a user, or an organization) is relatively high compared to the promises of dynamic resource sharing envisioned. And although it might be legitimate to require user authentication at a local site, authentication at the shared resource is not required *per se* to make an authorization decision. Moreover the security mechanisms proposed in current Grid environments fail to fulfill the possibly short-lived requirements of some applications. The deployment and management of secure communication channels between very dynamic coalitions of nodes should be tackled by the communicating end-points themselves because they have a better view of what the user and the application communication needs are. Instead of that, they are managed in a centralized way by the GSI and their trusted third parties.

In some contexts, as medical area, delegation to an external, centralized entity such as a CA or a VO, is not acceptable. In addition, current production grid infrastructures provide little isolation between users, especially users from a same VO, while much finer grain protection of data and computations is needed. The following sections will describe an alternative approach to overcome these limitations. They address (i) resources confinement, as provided by virtualization techniques, and (ii) an agile access control infrastructure, as provided by SPKI.

3. THE HIPERNET FRAMEWORK

3.1 The VPXI concept

We define the Virtual Private eXecution Infrastructure (VPXI) concept as the aggregation of virtual computing resources interconnected by a virtual private overlay network. Ideally,

[1] http://www.eu-egee.org

[2] http://www.opensciencegrid.org

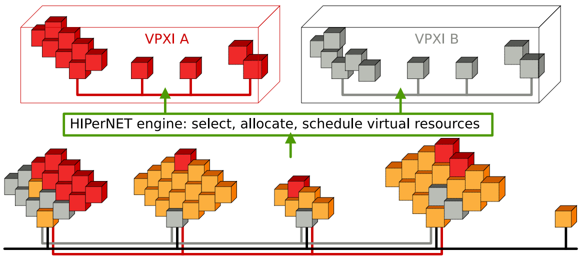
VPXI A
VPXI B
HIPerNET engine: select, allocate, schedule virtual resources

Figure 1: Global infrastructure (HIPerSpace)

any user of a VPXI has the illusion that he is using his own system, while in reality he is using multiple systems, part of the global system. The resulting virtual instances are kept isolated from each others. The members of a VPXI have a consistent view of a single private TCP/IP overlay, independently from the underlying physical topology. A VPXI can span multiple networks belongings to disparate administrative domains. A user can join from any location, and use the same TCP/IP applications he was using on the Internet or its intranet.

A VPXI can be formally represented as a graph in which a vertex is in charge of active data processing functions and an edge in charge of moving the data between vertices. Its specification comprises the recursive description of: a) individual end resources or resource aggregates (clusters) involved, b) performance attributes for each resource element (capacity), c) security attributes for each resource element (access control, confidentiality level), d) commercial attributes for each resource element (maximum cost), e) temporal attributes for each resource element (time window for provisioning), f) elementary functions, which can be attributed to a single resource or a cluster, for example: request of *computing* nodes, *storage* nodes, *visualization* nodes, or *routing* nodes, g) specific service provided by the resource (data mining application, data compression software), h) the virtual network topology, including the performance characteristics (typically bandwidth and latency), the security, commercial and temporal attributes of the virtual channels.

To support the specifications of these VPXI, the VXDL language [12] has been studied and developed.

3.2 HIPerNet design principles

The HIPerNET engine is a key component in our proposal[3]. Its goal is to provide a framework to build and manage VPXIs which are private, dynamic, predictable and large-scale computing environments, that high-end challenging applications can use with traditional APIs: standard POSIX calls, sockets and Message Passing (MPI, OpenMP) communication libraries. A user preempt and interconnect virtually, for a given time-window, a pool of virtual resources from a hardware distributed infrastructure in order to execute his application. The originality of HIPerNet is to combine system and networking virtualization technologies with crypto-based-security, bandwidth sharing and advance reservation mechanisms. To make hosts allocation flexible, the OS-virtualization permits multiple virtual nodes to cohabit on the same physical host. The network is virtualized as well in order to permit multiple virtual overlay networks to cohabit on a shared communication infrastructure. An overlay network has an ideal vantage point to monitor and control the underlying physical network and the applications running on the VMs. All networks overlays and nodes are kept isolated both at the network and OS level to ensure security, performance-control and adaptability. To achieve this, HIPerNet leverages on network (and sub-transport) layer security (IPsec at layer 3, and HIP at layer 3.5), self-certifying naming (i.e. CBID/HIT) and SPKI delegation certificates to implement the security functionalities.

The HIPerNet substrate is transparent to all types of upper layers: upper layer protocols (e.g. TCP, UDP), APIs (e.g. sockets), middleware (e.g. Globus, Diet), applications, services and users. Hence, the HIPerNet model maintains backward compatibility with existing APIs, Middlewares and Applications which were designed for UNIX and TCP/IP APIs. Therefore, users do not need to learn new tools, developers do not need to port applications, legacy user authentication can still be used to enroll a user into a VPXI, and a middleware laid on the HIPerNet secure communication paradigm can securely provide the remaining services constituting the computing environment: Secure Storage and Secure Computation.

3.3 Resource management in HIPerNet

The HIPerNet framework aims at partitioning a distributed physical infrastructure (computers, disks, networks) into dedicated virtual private computing environment dynamically composed. When a new machine joins the physical resource set, HIPerNet prepares its operating system to enable several virtual machines (VMs) to be instantiated dynamically when required. This set of potential virtual machines is called an HIPerSpace and it is represented in the HIPerSpace Database. The HIPerSpace is the only entity that can see the physical entities.

A resource, volunteer to join the resource pool, is automatically initiated and registered in the HIPerSpace database. The discovery of all the devices of the physical node is also automatic. An image of the specific HIPerNet operating system is deployed on it. In our current HIPerNet implementation, the operating system image contains basically the Xen Hypervisor and its domain of administration called domain 0 (Dom 0). The HIPerSpace registrar (Operational HIPerVisor) collects and stores persistently data and manages accounts (e.g., the authentication database). It is therefore hosted by a physical machine outside of the HIPerSpace itself. For the sake of robustness and scalability, HIPerSpace registrar can be replicated or even distributed (section 4.2.2).

4. SECURITY MODEL AND TOOLS

In this section we detail our access control policies and virtual resource and channel privatization approach.

4.1 Towards a private, authorization centric approach

HIPerNet security model is based on the Simple Public Key Infrastructure (SPKI) [5][6] which follows a lighter approach

[3] The HIPerNET software is under development within the HIPCAL project [4]

to authentication and authorization than Virtual Organizations. An SPKI certificate carries an authorization granted to a certain entity and includes five attributes: 1) the public key of the sender (also called issuer), who grants the authorization; 2) the public key of the receiver (also called subject), who benefits from the authorization; 3) the description of this authorization (formulated according to a formalism called S-expression); 4) the optional delegation capabilities granted to the subject, i.e., whether or not, and under which conditions, he can delegate this authorization to other entities; 5) and a validity period for this authorization.

The certificate is then digitally signed by the sender, using its private key. Upon receiving the certificate, the receiver verifies the signature, using the sender's public key available in the certificate, in order to check the message integrity and authenticate the sender. This method is safe because the public key is *the one and only* identifier that fully identifies the entity (in this case the sender). Unlike the traditional PKI model, there is no indirection through a high level name, to which a public key is attached, and which is associated to a list of authorizations: the SPKI certificate itself carries a signed authorization associated to the resource identified by its public key and granted to a receiver, also identified by its public key.

This authorization transfer and delegation mechanism is used throughout HIPerNet as an efficient and scalable way to enforce security policies between the various entities: users, resources (virtual and physical) and HIPerNet Registrar(s). For instance, once a physical resource is registered, each virtual resource generates a {public key; private key} pair. Then it issues an SPKI certificate to the HIPerNet Registrar to grant this latter the right to access and use this virtual resource, and more importantly, the right to delegate this authorization to users. A virtual resource can limit this right to certain users (e.g., those who belong to a certain community). These rules, if any, are described in the SPKI certificate itself.

Then a user asks for the creation of a VPXI, composed of a certain number of virtual resources, specifying certain criteria that the resource will have to satisfy (e.g., the resource must be provided by a certain trusted community). The HIPerNet engine selects the virtual resources (after matching the criteria provided both by the user and the resources) and sends an SPKI authorization certificate for each resource. The user can then contact directly each virtual resource and provide it the SPKI authorization certificate he received. Once the resource has checked this SPKI certificate, the access is granted for the specified period. Before the validity period expires, the whole process must be reissued. A few detailed examples are provided in section 4.2.

Of course an additional mechanism is needed to prevent malicious users or resources to join the HIPerNet system, since SPKI only solves the problems of identity and authorization management. This is made possible by means of a traditional registration step, in which the identity of the registered users and physical resources is collected, and in which the end users have to accept a "good use" agreement form.

4.2 Examples of SPKI certificates exchanges

Let us illustrate these mechanisms with three scenarios, representative of the target applications.

4.2.1 *Basic scenario*

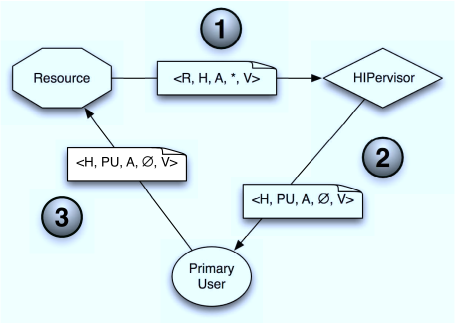

Figure 2: SPKI certificate flow in an VPXI.

Figure 2 illustrates the authorization certificates flow between the three main entities: the virtual resource, the HIPerNet Registrar, and the user. The `<R, H, A, *, V>` certificate is issued by the virtual resource (identified by its public key R), addressed to the Registrar (identified by its public key H), with certain authorizations (expressed as an S-expression whose semantics has been specified separately[5]), with a delegation capability to anybody (hence the *), for a certain validity period (V).

This SPKI certificate is kept by the HIPerNet Registrar, along with the similar certificates from other resources.

When a user asks for a VPXI creation, the Registrar first checks the rights associated to this user, selects the set of virtual resources that will constitute the VPXI, and sends to this user the associated SPKI certificates (Figure 2, step 2). The user therefore receives one or more certificates of the form: `<H, U, A, empty, V>`, that grant the access to the resource. However, this user does not have the right to delegate this authorization to anybody else, hence the `empty` delegation field. Note that providing the public key of the resource (its identifier) is sufficient to localize this resource thanks to HIP (see section 4.3).

Finally, the user contacts the resource, providing the certificate it received, in order to initiate the connection (Figure 2, step 2). The virtual resource checks the digital signature of the certificate, signed by the Registrar. The virtual resource also checks the certificate, making sure it corresponds to an authorization it has granted to the Registrar (we assume that this resource has kept a copy of the certificates it has generated to the HIPerNet Registrar). Once everything is okay, the secure connection is set up. In simple situations, it can mean storing the public key of the user locally, in order to initiate an SSH connection between the user and the virtual resource. In other situations, it can also mean creating the security association required to set up an IPsec link in mode transport.

[5] In simple use-cases where the authorization consists in an access to the resource, this field can contain the resource identifier and the nature of the authorization can be implicit. In more complex use-cases, the rules will be specified separately.

Authorizations are by nature valid for a limited period (infinite periods are not recommended). Therefore new SPKI certificates must be issued periodically by the resource before the expiration of the previous ones and then be propagated throughout the VPXI. The fact that an authorization is valid only for a limited period is a key feature. Associated to an enforcement mechanism (e.g, to shut an existing SSH connection when the resource realizes that the user's SPKI certificate has expired), this is a simple and natural way of managing authorizations.

Of course SPKI certificates can get lost (e.g., if UDP is used as the transport mechanism, or because a component like the HIPerNet Registrar temporarily fails). The validity period (V) and the SPKI certificate renewal frequency should be specified in such a way to make HIPerNet robust in front of SPKI certificate losses.

4.2.2 Managing several HIPerNet Registrars for improved scalability and robustness

Several extensions to the basic scenario are feasible. For instance, for the sake of scalability and robustness, several HIPerNet Registrars can be used. Physical/virtual resources register to one of them, indifferently. Then a secure synchronization mechanism between the Registrars enables them to be informed of the available virtual resources and their associated policies (e.g., by propagating the {SPKI certificates, Registrar} tuples).

When a VPXI is created, the Registrar `R1` selects the needed resources. When `R1` is not the Registrar that the resource contacted (let call this Registrar `R2`), he asks `R2` the possibility to use this resource. If the resource is available, `R2` generates a new certificate to `R1` that enables it to delegate the resource to the user and marks the virtual resource as used.

4.2.3 Managing several users per VPXI

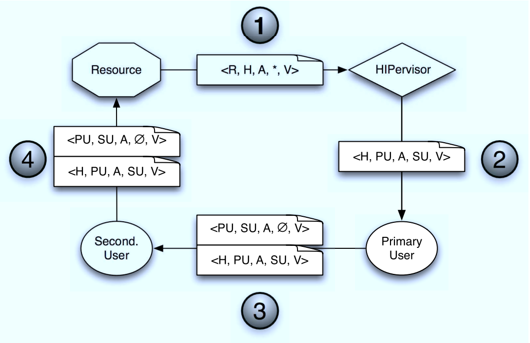

Figure 3: SPKI authorization certificate flow when a secondary user joins an existing VPXI.

Another common extension consists in having several users per VPXI. Let us call `U1` the primary user that asked for the creation of the VPXI, and `U2` the additional user. Once the VPXI is created, `U2` contacts `U1` for an access to the VPXI. This later has the possibility to deny this access (e.g., because of incompatible confidentiality requirements). If everything is okay, `U1` propagates the access request to the HIPerNet Registrar, who can also deny the access (e.g., because `U2` is not a registered HIPerNet user). If everything is okay, for each virtual resource, the Registrar generates a new SPKI authorization certificate to `U1` with delegation capabilities to `U2`. `U1` can then forward this SPKI certificate to `U2` along with a new certificate signed by itself to `U2`, but this time without any delegation capability. This is needed since `U2` must be able to prove to the virtual resource that an access has been (indirectly) granted to him by the Registrar (which is what the first SPKI certificate says) and that he also received directly from `U1` this authorization (which is what the second SPKI certificate says). `U2` then forwards the two certificates to the virtual resource.

4.3 Associating public keys to entities with HIP

In this section, we explain how the HIP protocol automatically associates public keys to entities.

Traditionally, the IP address plays two independent roles: it is both a locator and an identifier. Network Layer Protocols (NLPs, e.g. IPv4, IPv6) use the locator role to route packets, while the Upper Layer Protocols (ULPs, e.g., TCP, UDP) use the identifier role. Because of this deliberate confusion, ULPs depend on location and fail when mobility and multi-homing cause a modification of the IP address.

The Host Identity Protocol (HIP) [14] decouples these two roles while maintaining a binding between an identifier and a set of associated locators. This is a virtualization of the network infrastructure from an upper layer or application standpoint. The identifier namespace defined by HIP is composed of the public key of the host (in our context, an HIPerNet entity, typically a VM) and is called a Host Identity (HI). For convenience, HIP also defines the Host Identity Tag (HIT), a 128 bits long truncated hash of the HI. Thanks to HIP, an application can now use directly the HIT instead of an IPv6 address. The HIP layer is then in charge of mapping the HI/HIT into the appropriate IP address during communications, transparently.

We see that there is a perfect match between HIP and SPKI. The entity's public key is *the one and only* meaningful identifier at the HIPerNet level (which does not prevent to use higher level names during interactions with humans). These keys are carried during SPKI exchanges and each entity involved in a given VPXI can collect the public keys of the other entities. Finally, the translation to/from the network level locator (i.e., the IP address) is performed by the HIP layer, just on time, when sending or receiving IP datagrams.

4.4 Set-up of secure channels between entities

To provide secure communications, HIPerNet relies on IPsec, TLS/SSL or SSH. For instance, with IPsec, an encrypted link (in ESP mode) provides a secure point-to-point link while packets are actually traveling through a multi-hop, insecure path. In that case, the local HIPerNet system, depending on the authorizations gathered by SPKI, interacts with the IPsec layer in order to authorize or deny the establishment of an IPsec connection. The principles are similar with TLS/SSL.

In the particular case of SSH, the authorization enforcement mechanism is even simpler since an entity (e.g., a virtual resource of a VPXI) can simply store the public key of the remote entity who asks for a secure access (e.g., a user). The availability of this public key is sufficient for the remote entity to connect. When the authorization expires, the entity removes the public key, thereby denying any further connection by the remote entity, and in case an SSH connection is currently up and active, this connection is closed by the HIPerNet system running on the entity.

5. EXPERIMENTS IN GRID'5000

5.1 HIPerNet software

To evaluate the combination of system and network virtualization with our security approach based on SPKI and HIP, we are developing and experimenting the HIPerNet software within the Grid'5000 testbed[4]. Grid'5000 enables user to request, reconfigure and access physical machines belonging to 9 sites distributed in France. In our experiment, we reserve several Grid'5000 nodes to compose a pool of physical resources that we initialize to form an HIPerSpace. To instantiate an HIPerSpace, specific tools provided by the hosting Grid are used. This is the only part aware of the physical infrastructure of the HIPerNet Software. All the other parts are independent of the physical resources because they use them indirectly through the services provided by HIPerNet. In Grid'5000, the HIPerSpace appears like a set of ordinary jobs scheduled by OAR [3] with the use of a specific operating system image deployed by kadeploy [6].

5.2 HIP layer evaluation

Currently, several implementations of HIP are available but do not implement the same version of the draft describing the HIP protocol and then do not always interoperate. We evaluated the three following implementations: OpenHIP [7], HIP for Linux (HIPL) [8] and HIP for FreeBSD [9].

OpenHIP (figure 4) consists of a user-space HIP daemon (hipd) that implements the control plane protocol. OpenHIP can be entirely in user-space or in user-space with kernel support. The user-space version does all ESP encryption of data packets in user-space, while the Linux kernel version uses patches to IPsec tools and a kernel patch to extend/reuse the Linux ESP implementation. As represented in the figure, in our architecture, the data submitted to the IP layer are rerouted to a virtual driver (TAP driver) and redirected to the HIP/SPKI layer which operate the HIT/IPaddress mapping and the SPKI validity check.

Host Identity Protocol for Linux (HIPL) enables to encrypt and protect all Internet connections similar to TLS, but does not require changes in applications and works also with UDP. HIPL also provides public key based access control, direct end-to-end connectivity and strong end-to-end authentication.

With OpenHIP, applications use the same IP addresses in their sockets. With HIPL, applications use HIT (section 4.3).

To implement the HIPerNet software and test it within Grid'5000, we chose the OpenHIP stack which is available for IPv4 as well as IPv6. We did a simple evaluation to measure IPsec overhead. Our experiment has been made over a 100Mbps Ethernet link connecting two PC back-to-back. We used the iperf tool to measure the achieved mean throughput of a TCP connection. The raw performance obtained with the standard IP stack were about 94.3 Mbps. Over IPsec we obtained a mean throughput of about 33 Mbps. The first run was even worse (due to the Security Association establishment) since we only reached 27 Mbps. This overhead is obviously due to encryption operations (AES in this experiment) and is of course very dependent of the processing power of the end nodes.

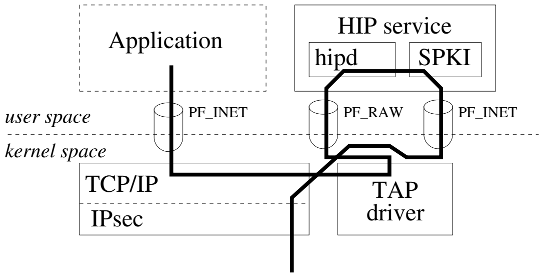

Figure 4: Our architecture based on OpenHIP with Linux kernel support.

5.3 Medical imaging application deployment on the testbed

The HIPerNet engine was tested through a complex, real-scale medical image analysis application known as *bronze standard*. This section reports on its porting for execution in a private VPXI infrastructure. The bronze standard [10] technique tackles the difficult problem of validating of medical image analysis procedures.

The bronze standard workflow is enacted with the data-intensive grid-interfaced MOTEUR workflow manager [9] designed to optimize the execution of data-parallel flows. It submits the workflow tasks to the VPXI infrastructure through the DIET middleware, a scalable grid scheduler based on a hierarchy of agents communicating through CORBA.

The algorithm performance estimated is valid for a typical image database sample. In the experiments reported below, we use a clinical database of 32 patient image pairs to be registered by the different algorithms involved in the workflow. For each run, the processing of the complete image database results in the generation of approximately 200 computing tasks (30 seconds to 5 minutes computation time each on a state of the art PC). The data volume transferred for each task is in the order of 30MB. The makespan of the application parallel execution is in the order of 20 minutes.

For testing VPXIs, a system image containing the operating system based on a standard Linux distribution Debian *Etch* with a kernel version 2.6.18-8 for *AMD64*, the domain-specific registration services and the middleware components (MOTEUR and DIET) was created. The application runs are performed on a testbed composed of 35 nodes: one node is dedicated to the DIET master agent, the execution traces collector and CORBA services; a second node is reserved for the MOTEUR workflow manager; a third node is the file

[6] http://kadeploy.imag.fr
[7] http://www.openhip.org
[8] http://infrahip.hiit.fi
[9] http://hip4inter.net

server containing the input image database and storing the computation results; and 32 virtual nodes are running the application services. The VPXI infrastructure is reserved in the HIPerNet hosted on the *capricorne* Grid'5000 cluster, in Lyon, France. Each node runs in a virtual OS instance on one of the 2.0 GHz Opteron CPUs dual-cores: each virtual node runs on a dedicated core with 512 MB of memory.

Three experiments were performed: (i) sequential runs (baseline), (ii) parallel runs on the physical grid and (iii) parallel runs on the VPXI. In each experiment, we repeated the application 10 times to measure the average and standard deviation of the application makespan, the data transfer and task execution time. In the base-line experiment, the application makespan obtained is 3h 49min 17s (±13s) on a physical resource and 3h 51min 9s (±67s) on a virtualized one. The average execution time overhead of the tasks is low: +2.1% (+6.76% in the worst case and usually less than +2%). In the parallel runs, the application makespan is 14min 31s (±2min) on the physical grid and 17min 37s (±2min) on the VPXI. This corresponds to a +21.3% makespan increase, that is due to +2.1% tasks average execution time overhead and +91.3% average data transfer overhead (+168% in the worst case and +7.9% in the best case). The impact on the makespan is partly compensated for by parallel execution and data transfers.

The following metrics are useful to assess the experimental results:
Portability: the application porting from the physical to the virtual infrastructure was almost transparent. Apart from minor user environment configuration issues (use of of 64 bits libraries on the VPXI 64 bits OS), the application was successfully executed on the virtualized infrastructure without need for any adaptation thanks to the transparency of the VPXI network and OS virtualization.
Security: the application running on the virtual infrastructure is fully confined. Data is neither exposed to outsiders during storage on disk nor network transfer. In the future, the SPKI file access control mechanism will enable for more elaborate control on the resources utilization.
Performance: the virtualized environment is highly secured but it comes at the cost of execution and the transfer overheads. The execution time is not impaired by a factor higher than +6.76%. Disk to disk data transfers are more critical and may experience an overhead of +168%. This overhead does not increase much with the distance. These delays are however compensated by parallel execution. Overall, the application makespan measurement shows that the VPXI overhead does not exceed a 21.3% loss in the worst case.

6. RELATED WORKS

In this section, we briefly describe the security approach of some projects that explore a virtual infrastructure composition on distributed resources.

In [2], the authors propose VINI, a virtual network infrastructure that allows several virtual networks to share a single physical infrastructure in a similar way to HIPerNet. Researchers can run experiences in isolated virtual network slices with real routing software they can personalize. HIPerNet pushes this facility a step further by allowing the user to deploy a personalized operating system on his virtual nodes running each one its own kernel inside Xen virtual machines. This provides also a full isolation between virtual nodes and a controlled resource sharing. VINI has dedicated IP address blocks, while HIPerNet interposes a HIP-layer between the network and the application level that provides each resource with a unique identifier (HIT).

VioCluster [16] logically gather machines between virtual domains, allowing a cluster to dynamically grow and shrink based on resource demand. Network virtualization in VioCluster is made by an hybrid version of VIOLIN [11] which gives to a machine the ability to connect to the private network through a distributed virtual switch (in user space) which forward the network traffic to corresponding processes on other nodes of the virtual domains. VNET, a virtual private network that implements a virtual local area network spread over a wide area using layer 2 tunneling, was extended in Virtuoso middleware [17]. These extension improves VNET to act as an adaptive overlay network for virtual machines that can be optionally encrypted using SSL. The session establishment among VNET servers and clients is performed using a text-based protocol. The clients presents a password or an SSL certificate.

PVC [15] explores the combination of various Grid, P2P and VPN approaches in the creation of instant grids. This distributed system is composed by a daemon process (peer) and a brokering service, which helps in the connections establishment. The security policies are implemented in two levels: intra and inter-domain. Each host connected to PVC has a private/public keys pair, and knows the public key of its master, before connecting to PVC infrastructure. Only the master peer registers participants on the brokering service. To start connection both peers obtain the other public key from the brokering service and decode the message with the master public key. The peers authentication is performed using the classical security challenge-response.

XtreemOS [13] is proposing new services that should be added to current operating systems to build Grid infrastructure in a simple way. For security XtreemOS implements a VO manager which grants a set of short-lived VO credentials (XOS-Cred) to users applications. Each credential is composed by a private and a public part: the first one is a private keys and the second one includes VO identity, VO attributes and public keys. The system assumes that resource nodes trust the VO manager through a preinstallation of manager's root CA certificate in all nodes. None of these approaches rely as HIPernet on SPKI authorization certificates to manage authorizations, improving the delegation and multi-users authentication. Decoupling identifier/locator roles as does HIP is a hot topic as it provides many advantages. Alternatives to HIP like AIP (Accountable Internet Protocol) [1] have recently been proposed. AIP is a pure layer 3 protocol replacing current IPv4 and IPv6 addressing models. AIP targets Future Internet design and does not envision an incremental deployment in the current Internet. While HIP adopts the same philosophy for IP address role it is less ambitious. It is a layer 3,5 protocol and can be deployed without the change of all Internet routers. Another advantage of HIP is that it can currently be used with standards programming interfaces (e.g Socket). There is no AIP implementation available yet. Another recent alternative, called LISP (Locator/Identifier Separation Protocol) [7] proposes to insert a new IP layer below the current IP layer. This additional layer is used to map and encapsulate packets. This operation, applied by the border router is called *map-n-encap*. HIP required a mapping operation too, however this mapping is performed by the end nodes which is more realistic. AIP and LISP can not directly address the problems raised by the convergence of computing and communication and resource virtualization as HIP does when combined with SPKI.

7. CONCLUSIONS

This paper proposes the "Virtual Private eXecution Infrastructure" (VPXI) paradigm to transform the Internet into a huge shared computing and communication facility for many purposes, including sensitive applications. This paper examines in particular data and processing security requirements of sensitive applications and describes a solution to implement agile resource access control policies that can adapt to real-life application needs. This work proposes a substrate, named HIPerNet, to create and manage confined Virtual Private eXecution Infrastructures in a large scale distributed environment. The HIPerNet security model is based on the combination of network and system virtualization with the cryptographic identification of (virtual) resources. We have highlighted how SPKI defines a flexible private authorization management layer for these resources (section 4.1). We also presented another key component of our proposal which is the mapping layer, the Host Identity Protocol (HIP). HIP which decouples the locator/identifier roles of addresses (section 4.3) and handles the automatic translation of the cryptographic identifiers to IP addresses. We have then discuss implementation issues and experimentations in Grid'5000. Finally we have illustrated how HIPerNet addresses the challenging security requirements of the biomedical community. In future work we plan to investigate and integrate, trust-aware resource management solutions within the HIPerNet tool and a trust level constrain in VPXI specification to customize the security cost.

8. REFERENCES

[1] David G. Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponenand Daekyeong Moon, and Scott Shenker. Accountable Internet Protocol (AIP). In *ACM SIGCOMM*, August 2008.

[2] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation. *ACM SIGCOMM Computer Communication Review (CCR)*, 36(4):3–14, 2006.

[3] Nicolas Capit, Georges Da Costa, Yiannis Georgiou, Guillaume Huard, Cyrille Martin, Gregory Mounie, Pierre Neyron, and Olivier Richard. A batch scheduler with high level components. In *Cluster Computing and Grid 2005 (CCGrid05)*, 2005.

[4] Franck Cappello, Frederic Desprez, Michel Dayde, Emmanuel Jeannot, Yvon Jegou, Stephane Lanteri, Nouredine Melab, Raymond Namyst, Pascale Vicat-Blanc Primet, Olivier Richard, Eddy Caron, Julien Leduc, and Guillaume Mornet. Grid5000: a nation wide experimental grid testbed. *Int. Journal on High Performance Computing Applications*, 2006.

[5] C. Ellison. *SPKI Requirements*, September 1999. IETF Request for Comments, RFC 2692.

[6] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI Certificate Theory*, Sept. 1999. IETF Request for Comments, RFC 2693.

[7] D Farinacci. Locator/ID Separation Protocol (LISP). *Internet Engineering Task Force (IETF), draft-farinacci-lisp (work in progress)*, October 2008.

[8] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A Security Architecture for Computational Grids. In *5th ACM Conference on Computer and Communications Security (CCS)*, 1998.

[9] T. Glatard, J. Montagnat, D. Lingrand, and X. Pennec. Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR. *Int. Journal of High Performance Computing and Applications (IJHPCA)*, 22(3):347–360, August 2008.

[10] T. Glatard, X. Pennec, and J. Montagnat. Performance evaluation of grid-enabled registration algorithms using bronze-standards. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'06)*, October 2006.

[11] X. Jiang and D. Xu. VIOLIN: Virtual Internetworking on Overlay Infrastructure. In *ISPA*, pages 937–946, 2004.

[12] G. P. Koslovski, P. Vicat-Blanc Primet, and A. S. Charão. VXDL: Virtual Resources and Interconnection Networks Description Language. In *2nd Int. Conference on Networks for Grid Applications (GridNets'08)*, Oct. 2008.

[13] Christine Morin. Xtreemos: A grid operating system making your computer ready for participating in virtual organizations. In *10th IEEE Int. Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*, 2007.

[14] R Moskowitz and P Nikander. Host Identity Protocol (HIP) Architecture. *IETF Request for Comments, RFC 4423*, May 2006.

[15] Ala Rezmerita, Tangui Morlier, Vincent Néri, and Franck Cappello. Private virtual cluster: Infrastructure and protocol for instant grids. In *Euro-Par 2006, Parallel Processing, 12th International Euro-Par Conference*, volume 4128 of *Lecture Notes in Computer Science*, pages 393–404, Dresden, Germany, August 2006. Springer.

[16] P. Ruth, P. McGachey, and Dongyan Xu. Viocluster: Virtualization for dynamic computational domains. *Cluster Computing, 2005. IEEE International*, pages 1–10, Sept. 2005.

[17] A. Sundararaj and P. Dinda. Towards Virtual Networks for Virtual Machine Grid Computing. In *Proceedings of the third USENIX Virtual Machine Research and Technology Symposium (VM 04)*, May 2004.