Chapter 5
Security Infrastructure for Dynamically Provisioned Cloud Infrastructure Services

Yuri Demchenko, Canh Ngo, Cees de Laat, Diego R. Lopez, Antonio Morales, and Joan A. García-Espín

Abstract This chapter discusses conceptual issues, basic requirements and practical suggestions for designing dynamically configured security infrastructure provisioned on demand as part of the cloud-based infrastructure. This chapter describes general use cases for provisioning cloud infrastructure services and the proposed architectural framework that provides a basis for defining the security infrastructure requirements. The proposed security services lifecycle management (SSLM) model addresses specific on-demand infrastructure service provisioning security problems that can be solved by introducing special security mechanisms to allow security services synchronisation and their binding to the virtualisation platforms run-time environment. This chapter describes the proposed dynamically provisioned access control infrastructure (DACI) architecture and defines the necessary security mechanisms to ensure consistent security services operation in the provisioned virtual infrastructure. In particular, this chapter discusses the design and use of a security token service for federated access control and security context management in the generically multi-domain and multi-provider cloud environment.

Keywords Access control • Cloud infrastructure • DACI • IaaS • Security • Trusted computing

Y. Demchenko (✉) • C. Ngo • C. de Laat
[AU1] University of Amsterdam, Amsterdam, The Netherlands

D.R. Lopez
Telefonica I+D, Madrid, Spain

A. Morales
RedIRIS, Madrid, Spain

J.A. García-Espín
I2CAT Foundation, Barcelona, Spain

S. Pearson and G. Yee (eds.), *Privacy and Security for Cloud Computing*, Computer Communications and Networks, DOI 10.1007/978-1-4471-4189-1_5,

5.1 Introduction

Cloud technologies [1, 2] are emerging as a new way of provisioning virtualised computing and network infrastructure services on demand for collaborative projects and groups. Security in provisioning virtual infrastructure services should address two general aspects: supporting secure operation of the provisioning infrastructure and provisioning a dynamic access control infrastructure as part of the provisioned on-demand virtual infrastructure.

The current cloud security model is based on the assumption that the user/customer should trust the cloud service provider (CSP). This is governed by the service level agreement (SLA) that in general defines mutual provider and user expectations and obligations. However, such an approach addresses only the first part of the problem and does not scale well with the potential need to combine cloud-based services from multiple providers when building complex infrastructures.

Cloud providers are investing significant efforts and costs into making their own infrastructures secure and achieving compliance with the existing industry security services management standards (e.g. Amazon Cloud recently achieved Payment Card Industry Data Security Standard (PCI DSS) compliance certification and Microsoft Azure Cloud claims compliance with ISO27001 security standards). However, overall security of cloud-based applications and services will depend on two other factors: security services implementation in user applications and binding between virtualised services and cloud virtualisation platforms. Advanced security services and fine-grained access control cannot be achieved without deeper integration with the cloud virtualisation platform and incumbent security services, which in its turn can be achieved with open and well-defined cloud IaaS platform architectures.

This chapter presents recent results of the ongoing research on developing architecture and framework for dynamically provisioned security services as part of the provisioned on-demand cloud-based infrastructure services. This chapter extends earlier published works by authors with the recent results and implementation experiences.

This chapter analyses the basic use cases and proposes an abstract model for on-demand infrastructure services provisioning. Section 5.3 provides a short description of the architectural framework for on-demand infrastructure services provisioning proposed in earlier authors' work [3, 4]. It is used as a basis to define the general security requirements to the security infrastructure. Section 5.4 discusses conceptual issues, basic requirements, proposed architectural solutions, supporting security mechanisms and practical suggestions for provisioning dynamically configured access control services as part of the provisioned on-demand cloud-based infrastructure services. This section summarises the earlier works by authors [5–7] and describes the proposed dynamically provisioned access control infrastructure (DACI). Section 5.5 describes the security token service that allows federated access control to distributed multi-domain cloud resources.

Consistent security services design, deployment and operation require continuous security context management during the whole security services lifecycle, which is aligned to the main provisioned services lifecycle. The proposed security services lifecycle management (SSLM) model addresses specific on-demand infrastructure service provisioning security problems that can be solved by introducing a special security mechanism to allow synchronisation of security services and their binding to virtualisation platform and run-time environment. This chapter discusses how these security mechanisms can be implemented by using Trusted Computing Group Architecture (TCG Architecture) and the functionality of the Trusted Platform Module (TPM) that is currently available in many computer platforms and supported by most VM management platforms. Section 5.4.5 describes the proposed security bootstrapping protocol that uses TPM functionality and can be integrated with DACI.

The practical implementation of DACI reveals a wide spectrum of problems related to distributed access control, policy and related security context management. This chapter discusses important security services and mechanisms that ensure consistency of the provisioned security infrastructure and its integration with user applications: authorisation tokens used for provisioning and authorisation session management and for security context exchange between infrastructure services and providers (Sect. 5.4.6) and the standard-based security token service as an important mechanism for inter-domain access control and identity management (Sect. 5.5).

5.2 Background

5.2.1 Cloud Computing as an Emerging Provisioning Model for Complex Infrastructure Services

Modern e-Science and high-technology industry require high-performance infrastructure to handle large volume of data and support complex scientific applications and technological processes. Dynamicity of projects and collaborative group environment require that such infrastructure is provisioned on demand and capable of dynamic (re-) configuration. A large amount of currently available e-Science/research infrastructures is currently available on the grid, which in the case of Europe are coordinated by the European Grid Initiative (EGI) [8]. Future research infrastructures will inevitably evolve in the direction of using cloud resources and will combine both grid and cloud resources.

Currently large grid projects and cloud computing providers use their own dedicated network infrastructure that can handle the required data throughput but typically are over-provisioned. Their network infrastructure and security model are commonly based on the traditional VPN model that spreads worldwide, creates distributed environment for running their own services geographically distributed (like Google and Amazon) and provides localised access for users and local providers. Their service delivery business model and consequently security model are typically based and governed by a service level agreement (SLA) that in general defines mutual provider and user expectations and obligations.

Recently, cloud technologies [1, 2, 9] are emerging as infrastructure services for provisioning computing and storage resources and gradually evolving into the general IT resources provisioning. Cloud computing can be considered as natural evolution of the grid computing technologies to more open infrastructure-based services. Cloud "elasticity", as recognised by researchers and technology practitioners, [AU3] brings a positive paradigm shift in relation to the problem and the problem-solving infrastructure from sizing a problem to infrastructure to sizing infrastructure to the problem.

The current cloud services implement three basic service models: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). There are many examples of the latter two models, PaaS and SaaS, that are typically built using existing SOA (service-oriented architecture) [10] and Web Services or REST (representational state transfer) [11] technologies. However, the IaaS model, if intended to provision user or operator manageable infrastructure services, requires a new type of service delivery and operation framework that should also include security infrastructure integration with the user or enterprise legacy security infrastructure.

This chapter presents the ongoing research aimed at developing an architectural framework that will address known problems in on-demand provisioning virtualised infrastructure services that may include both computing resources (computers and storage) and transport network. The solutions for pooling, virtualising and provisioning computing resources are provided by current grid and cloud infrastructures. New solutions should allow the combination of IT and network resources, supporting abstraction, composition and delivery for individual collaborating user groups and applications.

5.2.2 General Use Case for Cloud-Based On-Demand Infrastructure Services Provisioning

One general use case for on-demand cloud-based infrastructure services provisioning can be considered: large project-oriented scientific infrastructure provisioning including dedicated transport network infrastructure. However, two different perspectives in developing infrastructure services can be considered – users and application developers' perspective, on one side, and providers' perspective, on the other side. Users are interested in uniform and simple access to resources and services that are exposed as cloud resources and can be easily integrated into the scientific or business workflow. Infrastructure providers are interested in infrastructure resource pooling and virtualisation to simplify their on-demand provisioning and extend their service offering and business model to virtual infrastructure provisioning.

Figure 5.1 illustrates the typical e-Science infrastructure that includes grid and cloud-based computing and storage resources, instruments, control and monitoring

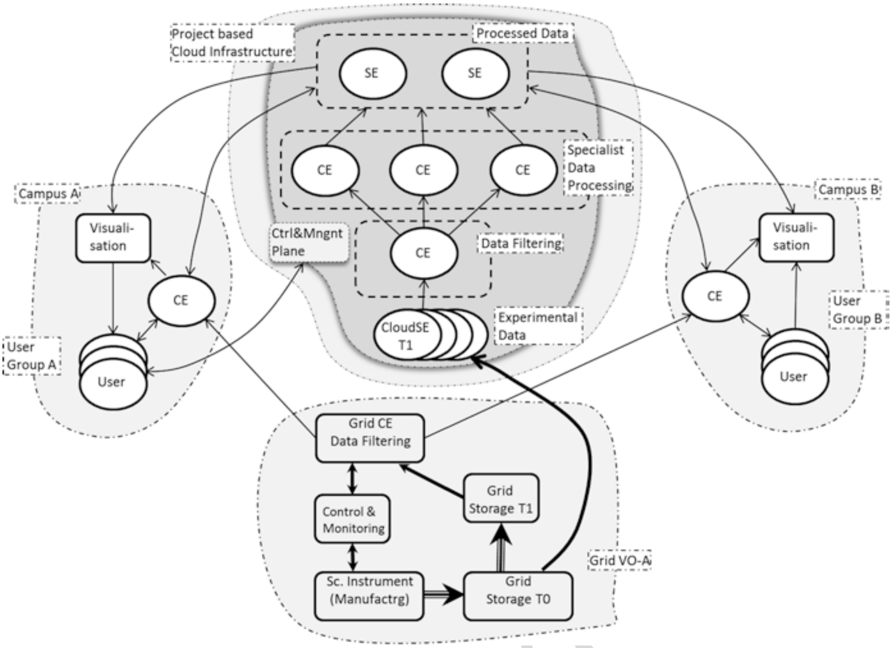


Fig. 5.1 Project-oriented collaborative infrastructure containing grid-based scientific instrument managed by grid VO-A, 2 campuses A and B, and cloud-based infrastructure provisioned on demand

system, visualisation system and users represented by user clients. The diagram also reflects that there may be different types of connecting network links: high-speed and low-speed which both can be permanent for the project or provisioned on demand.

The figure also illustrates a typical use case when a high-performance infrastructure is used by two or more cooperative users/researcher groups in different locations. In order to fulfil their task (e.g. cooperative image processing and analysis), they require a number of resources and services to process raw data on distributed grid or cloud data centres, analyse intermediate data on specialist applications and finally deliver the result data to the users/scientists. This use case includes all basic components of the typical e-Science research process: data collection, initial data mining and filtering, analysis with special scientific applications and finally presentation and visualisation to the users.

With the growing complexity and dynamicity of collaborative projects and applications, they will require access to network control and management functions to optimise their performance and resources usage. Currently, transport network, even if provided as VPN, is set up statically or can only be reconfigured by a network engineer.

5.3 Architectural Framework for Cloud IaaS Model

5.3.1 Abstract Model for On-Demand Infrastructure Services Provisioning

Figure 5.2 below illustrates the abstraction of the typical project- or group-oriented virtual infrastructure (VI) provisioning process that includes both computing resources and supporting network that is commonly referred as infrastructure services. The figure also shows the main actors involved in this process, such as physical infrastructure provider (PIP), virtual infrastructure provider (VIP) and virtual infrastructure operator (VIO).

The required supporting infrastructure services are depicted on the left side of the picture and include functional components and services used to support normal operation of all mentioned actors. The virtual infrastructure composition and management (VICM) layer includes the logical abstraction layer and the VI/VR adaptation layer facing correspondingly lower PIP and upper application layers. VICM-related functionality is described below as related to the proposed composable services architecture (CSA).

The proposed abstraction provides a basis and motivates the definition of architectural framework for cloud-based infrastructure services provisioning to support the main cloud IaaS features such as on-demand provisioning, elasticity, scalability, virtualisation, lifecycle management and combined compute and network resource

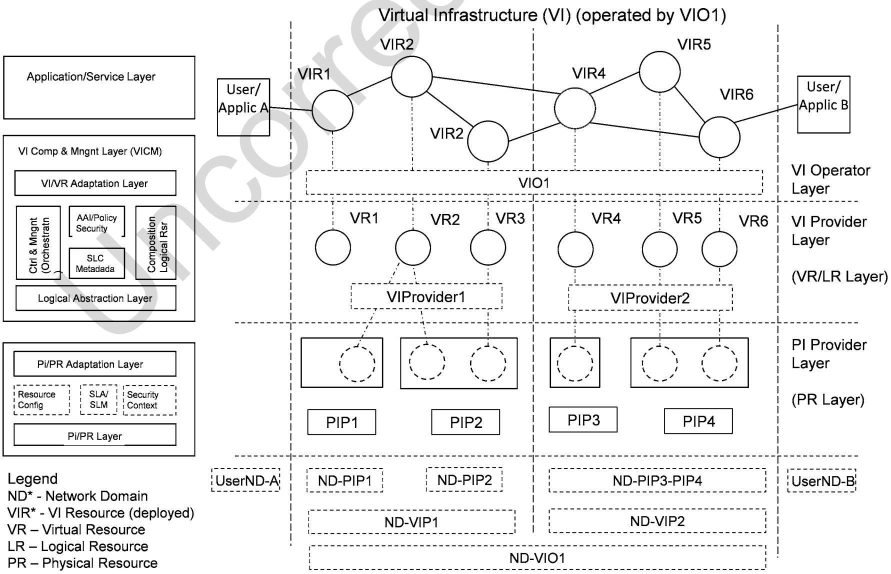


Fig. 5.2 Main actors, functional layers and processes in on-demand infrastructure services provisioning

provisioning. The proposed architectural framework comprises of the following components discussed in this chapter:

- Infrastructure services modelling framework (ISMF)
- Composable services architecture (CSA)
- Service delivery framework (SDF)
- Dynamically provisioned security infrastructure that includes dynamically provisioned access control infrastructure (DACI) and related security services and mechanisms for inter-domain security context management

The proposed architecture is SOA (service-oriented architecture) [10] based and uses the same basic operation principle as known and widely used SOA frameworks, which also provides a direct mapping to the possible VICM implementation platforms such as enterprise service bus (ESB) or OSGi framework [12, 13].

The infrastructure provisioning process, also referred to as service delivery framework (SDF), is adopted from the TeleManagement Forum SDF [14, 15] with necessary extensions to allow dynamic services provisioning. It includes the following main stages: (1) infrastructure creation request sent to VIO or VIP that may include both required resources and network infrastructure to support distributed target user groups and/or consuming applications, (2) infrastructure planning and advance reservation, (3) infrastructure deployment including services synchronisation and initiation, (4) operation stage and (5) infrastructure decommissioning. The SDF combines in one provisioning workflow all processes that are run by different supporting systems and executed by different actors.

Physical resources (PR), including IT resources and network, are provided by physical infrastructure providers (PIP). In order to be included into VI composition and provisioning by the VIP, they need to be abstracted to logical resource (LR) that will undergo a number of abstract transformations including possibly interactive negotiation with the PIP. The composed VI needs to be deployed to the PIP which will create virtualised physical resources (VPR) that may be a part, a pool or a combination of the resources provided by PIP.

The deployment process includes distribution of common VI context, configuration of VPR at PIP, advance reservation and scheduling and virtualised infrastructure services synchronisation and initiation to make them available to application layer consumers.

The proposed abstract models allow outsourcing the provisioned VI operation to the VI operator (VIO) which is from the user/consumer point of view, provide valuable services of the required resources consolidation – both IT and networks – and take a burden of managing the provisioned services.

5.3.2 *Dynamically Provisioned Cloud Security Infrastructure*

The proposed architecture provides a basis and motivates development of the generalised framework for provisioning dynamic security infrastructure that includes

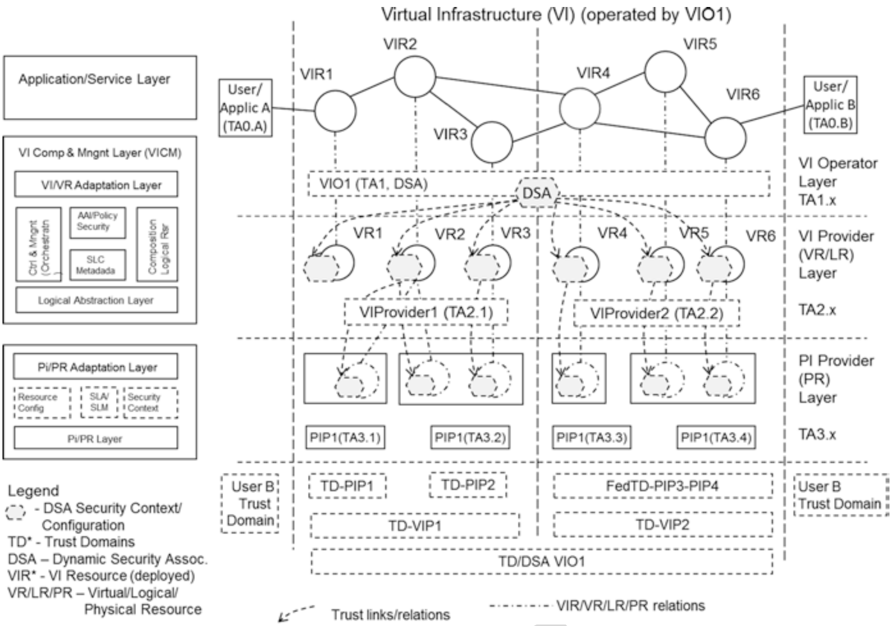


Fig. 5.3 Dynamic security association (*DSA*) to support security infrastructure provisioned on demand as a part of the overall infrastructure

the dynamically provisioned access control infrastructure (DACI), security services lifecycle management model (SSLM), common security services interface (CSSI) and related security services and mechanisms to ensure the consistency of the dynamically provisioned security services operation. The required security infrastructure should provide a common framework for operating security services at VIP and VIO layers and be integrated with the PIP and user legacy security services.

Figure 5.3 illustrates security and trust domain-related aspects in the infrastructure provisioning. It shows trust domains related to VIO, VIP and PIP that are defined by the corresponding trust anchors (TA) denoted as TA1, TA2 and TA3. The user (or requestor) trust domain is denoted as TA0 to indicate that the dynamically provisioned security infrastructure is bound to the requestor's security domain. The dynamic security association (DSA) is created as a part of the provisioning VI. It actually supports the VI security domain and is used to enable consistent operation of the VI security infrastructure.

5.3.3 *Infrastructure Services Modelling Framework*

The infrastructure services modelling framework (ISMF) provides a basis for virtualisation and management of infrastructure resources, including description, discovery, modelling, composition and monitoring. In this chapter, we mainly focus on the description of resources and the lifecycle of these resources. The described model in this section is being developed in the GEYSERS project [16].

5.3.3.1 Resource Modelling

The two main descriptive elements of the ISMF are the infrastructure topology and descriptions of resources in that topology. Besides these main ingredients, the ISMF also allows for describing QoS attributes of resources, energy-related attributes and attributes needed for access control.

The main requirement for the ISMF is that it should allow for describing physical resources (PR) as well as virtual resources (VR). Describing physical aspects of a resource means that a great level of detail in the description is required, while describing a virtual resource may require a more abstract view. Furthermore, the ISMF should allow for manipulation of resource descriptions such as partitioning and aggregation. Resources on which manipulation takes place and resources that are the outcome of manipulation are called logical resources (LR).

The ISMF is based on semantic Web technology. This means that the description format will be based on the Web Ontology Language (OWL) [17]. This approach ensures the ISMF is extensible and allows for easy abstraction of resources by adding or omitting resource description elements. Furthermore, this approach has enabled us to reuse the network description language [18] to describe infrastructure topologies.

5.3.3.2 Virtual Resource Lifecycle

Figure 5.4 illustrates relations between different resource presentations during the provisioning process stages that can also be defined as the virtual resource lifecycle.

The physical resource information is published by a PIP to the registry service serving VICM and VIP. This published information describes a PR. The published LR information presented in the commonly adopted form (using common data or semantic model) is then used by VICM/VIP composition service to create the requested infrastructure using a combination of (instantiated) virtual resources and interconnecting them with a network infrastructure. In its own turn, the network can be composed of a few network segments run by different network providers.

It is important to mention that physical and virtual resources discussed here are in fact complex software-enabled systems with their own operating systems and security services. The VI provisioning process should support the smooth integration into the common federated VI security infrastructure by allowing the definition of a common access control policy. Access decisions made at the VI level should be trusted and validated at the PIP level. This can be achieved by creating dynamic security associations during the provisioning process.

this figure will be printed in b/w

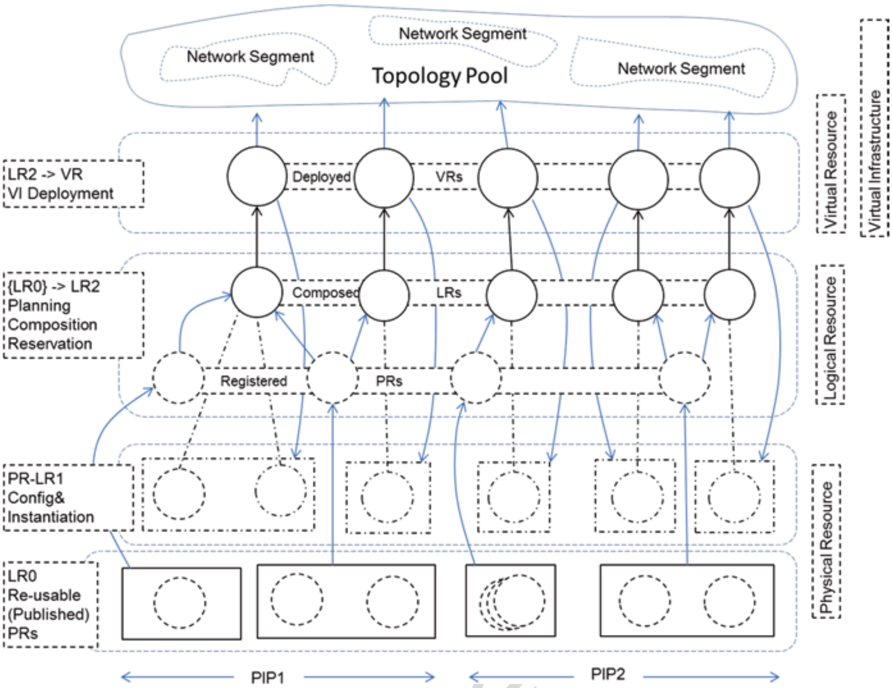


Fig. 5.4 Relation between different resource presentations in relation to different provisioning stages (Refer to Fig. 5.3 for the initial VI presentation)

5.3.4 *Service Delivery Framework (SDF)*

Service-oriented architecture (SOA) [10] allows for better integration between business process definition with higher abstraction description languages and dynamically composed services and provides a good basis for creating dynamically composable services that should also rely on the well-defined services lifecycle management (SLM) model. Most of existing SLM frameworks and definitions are oriented on rather traditional human-driven services development and management. Dynamically provisioned and reconfigured services will require rethinking of existing models and proposing new security mechanisms at each stage of the typical provisioning process.

The service delivery framework (SDF) [14] proposed by the TeleManagement Forum (TMF) provides a common basis for defining software-enabled services [15] lifecycle management framework that includes both the service delivery stages and required supporting infrastructure services.

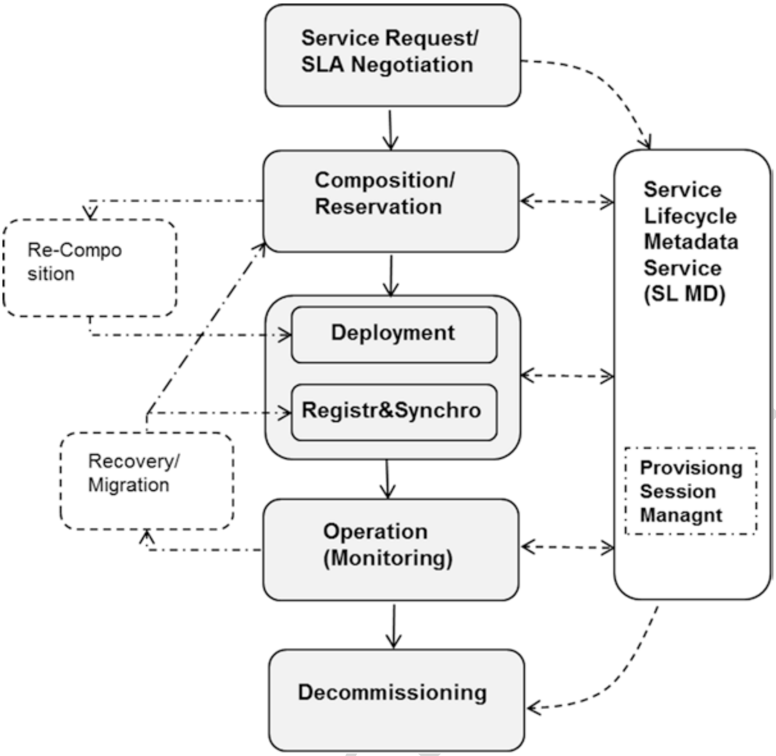


Fig. 5.5 On-demand composable services provisioning workflow

5.3.4.1 SDF Workflow

Figure 5.5 illustrates the main service provisioning or delivery stages:

Service request (including SLA negotiation). The SLA can describe QoS and security requirements of the negotiated infrastructure service along with information that facilitates authentication of service requests from users. This stage also includes generation of the global reservation ID (GRI) that will serve as a provisioning session identifier and will bind all other stages and related security context.

Composition/reservation, which also includes *reservation session binding* with GRI providing support for a complex reservation process in a potentially multi-domain multi-provider environment. This stage may require access control and SLA/policy enforcement.

Deployment, including services *registration and synchronisation*. Deployment stage begins after all component resources have been reserved and includes distributing the common composed service context (including security context) and binding the reserved resources or services to the GRI as a common provisioning session ID. The registration and synchronisation stage specifically targets possible scenarios with the provisioned services migration or re-planning. In a simple case, the registration stage binds the local resource or hosting platform run-time process ID to the GRI as a provisioning session ID.

*Operation (*including *monitoring)*. This is the main operational stage of the provisioned on-demand composable services. Monitoring is an important functionality of this stage to ensure service availability and secure operation, including SLA enforcement.

Decommissioning stage ensures that all sessions are terminated, data are cleaned up and session security context is recycled. The decommissioning stage can also provide information to or initiate services usage accounting.

The two additional (sub-)stages can be initiated from the operation stage and/or based on the running composed service or component services state, such as their availability or failure:

Recomposition or replanning that should allow incremental infrastructure changes.

Recovery/migration can be initiated by both the user and the provider. This process can use MD SLC to initiate full or partial resources re-synchronisation; it may also require recomposition.

5.3.4.2 Infrastructure Services to Support SDF

Implementation of the proposed SDF requires a number of special infrastructure support services (ISS) to support consistent (on-demand) provisioned services lifecycle management (similar to the above-mentioned TMF SDF) that can be implemented as a part of the CSA middleware.

The following services are essential to support consistent service lifecycle management:

- Service repository or service registry that supports services registration and discovery
- Service lifecycle metadata repository (MD SLC as shown on Fig. 5.3) that keeps the services metadata during the whole services lifecycle that include services properties, services configuration information and services state
- Service and resource monitor, an additional functionality that can be implemented as a part of the CSA middleware and provides information about services and resources state and usage

5.3.5 The Composable Services Architecture

The infrastructure as a service provisioning involves dynamics creation of an infrastructure consisting of different types of resources together with necessary (infrastructure wide)

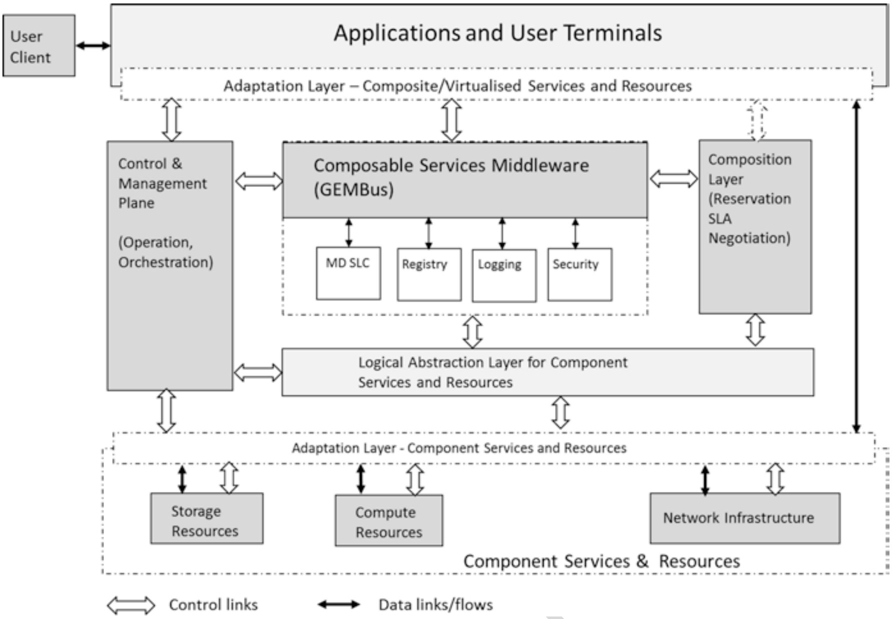


Fig. 5.6 Composable service architecture and main functional components

control and management planes, all provisioned on demand. The CSA proposed by authors [3] provides a framework for the design and operation of the composite/complex services provisioned on demand. It is based on the component services virtualisation, which in its own turn is based on the logical abstraction of the (physical) component services and their dynamic composition. Composite services may also use the orchestration service provisioned as a CSA infrastructure service to operate composite service-specific workflow.

Figure 5.6 shows the major functional components of the proposed CSA and their interaction. The central part of the architecture is the CSA middleware that should ensure smooth service operation during all stages of the composable services lifecycle.

Composable services middleware (CSA-MW) provides a common interaction environment for both (physical) component services and complex/composite services, built of component services. Besides exchanging messages, CSA-MW also contains/provides a set of basic/general infrastructure services required to support reliable and secure (composite) services delivery and operation:

- Service lifecycle metadata service (MD SLC) that stores the services metadata, including the lifecycle stage, the service state and the provisioning session context.
- Registry service that contains information about all component services and dynamically created composite services. The registry should support automatic services registration.
- Logging service that can be also combined with the monitoring service.

- Middleware security services that ensure secure operation of the CSA/ middleware.

Note that both logging and security services can be also provided as component services that can be composed with other services in a regular way.

The CSA defines also a logical abstraction layer for component services and [AU4] resources, which is a necessary part in creating services pool and virtualisation. Another functional layer is the services composition layer that allows presentation of the composed/composite services as regular services to the consumer.

The control and management plane provides necessary functionality for managing composed services during their normal operation. It may include orchestration [AU5] service to coordinate component services operation; in a simple case, it may be standard workflow management system.

CSA defines a special adaptation layer to support dynamically provisioned control and management plane interaction with the component services which, to be included into the CSA infrastructure, must implement adaptation layer interfaces that are capable of supporting major CSA provisioning stages, in particular, service identification, services configuration and metadata including security context, and provisioning session management.

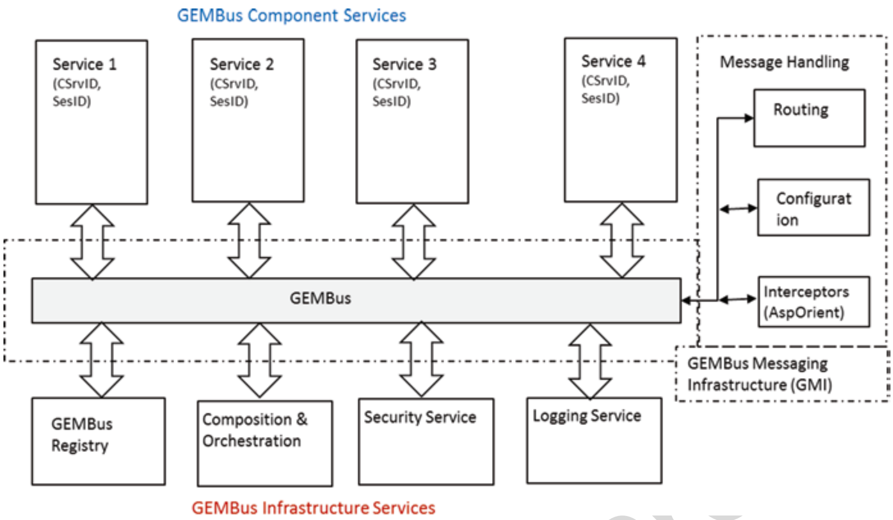
5.3.6 GEMBus as a CSA Middleware Platform

GÉANT Multi-domain service bus (GEMBus) is being developed as a middleware for composable services in the framework of GÉANT3 project [19, 20]. GEMBus incorporates the SOA services management paradigm in on-demand service provisioning. The GEMBus is built upon the industry accepted enterprise service bus (ESB) [12] and will extend it with the necessary functional components and design pattern to support multi-domain services and applications.

The goal of GEMBus is to establish seamless access to the network infrastructure and the services deployed upon it, using direct collaboration between network and applications, and therefore providing more complex community-oriented services through their composition.

Figure 5.7 illustrates the suggested GEMBus architecture. GEMBus infrastructure includes three main groups of functionalities:

- GEMBus messaging infrastructure (GMI) that includes, first of all, messaging backbone and other message handling supporting services such as message routing, configuration services, secure messaging and event handler/interceptors. The GMI is built on and extends the generic ESB functionality to support dynamically configured multi-domain services as defined by GEMBus.
- GEMBus infrastructure services that support reliable and secure composable services operation and the whole services provisioning process. These include such services as composition; orchestration; security, in particular, security token service; and the also important lifecycle metadata service, which are provided by the GEMBus environment/framework itself.



this figure will be printed in b/w

Fig. 5.7 GEMBus infrastructure, including component services, service template, infrastructure services and core message-processing services

- Component services, although typically provided by independent parties, need to implement special GEMBus adaptors or use special "plug-in sockets" that allow their integration into the GEMBus/CSA infrastructure.

The following issues have been identified to enable GEMBus operation in the multi-domain heterogeneous service provisioning environment:

- Service registries supporting service registration and discovery. Registries are considered as an important component to allow cross-domain heterogeneous services integration and metadata management during the whole services lifecycle.
- Security, access control and logging should provide consistent services and security context management during the whole provisioned services lifecycle.
- Service composition and orchestration models and mechanisms should allow integration with the higher-level scientific or business workflow.
- Messaging infrastructure should support both SOAP-based and RESTful (conforming to representational state transfer (REST) architecture) services [11].

The GEMBus and GMI, in particular, are built on the top of the standard Apache/Fuse messaging infrastructure that includes the following components [21, 22]:

- Fuse Message Broker (Apache ActiveMQ) messaging processor
- Fuse Mediation Router (Apache Camel) normalised message router

The GEMBus services and applications can be deployed on the standard Fuse or Apache ESB servers as component services that can be integrated with the standard OSGi [13] and Spring [23] compliant service development frameworks and platforms such as Fuse Services Framework/Apache CXF and Fuse ESB/Apache ServiceMix.

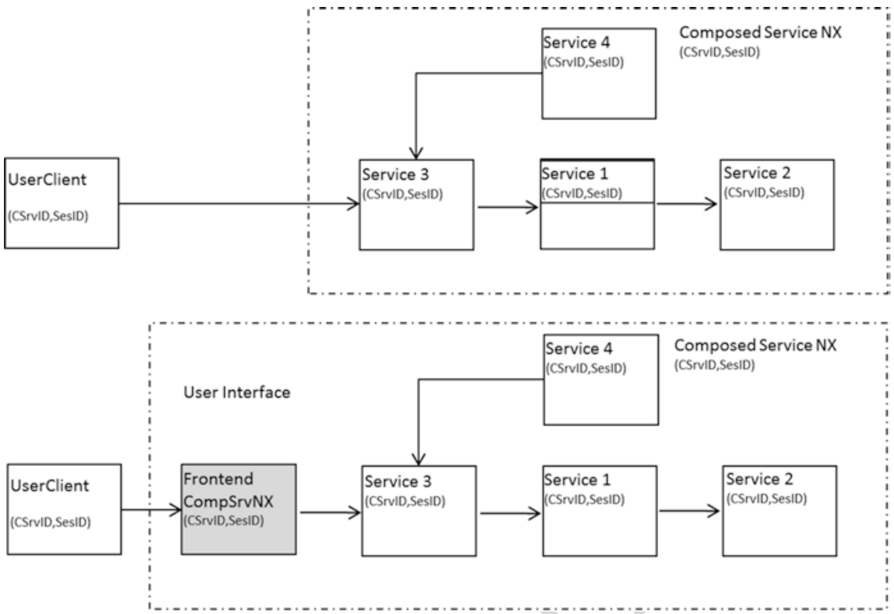


Fig. 5.8 Example of a composite service composed of services: service 1, service 2, service 3 and service 4 [AU6]

Figure 5.8 illustrates two examples of the composite services that are composed of four component services. In the second case, the composite service contains a special front-end service that is created of the corresponding service template that should be available for specific kind of applications. Examples of such service templates can be a user terminal (or rich user client) or a visualisation service. Requiring the GEMBus framework or toolkit to provide a number of typical service templates will provide more flexibility in delivery/provisioning composite services.

5.4 Cloud IaaS Security Infrastructure

5.4.1 General Requirements to Dynamically Provisioned Security Services

On-demand provisioning of cloud infrastructure services drives paradigm change in security design and operation. Considering evolutional relations between grids and clouds, it is interesting to compare their security models. This is also important from the point of view that future e-Science infrastructures will integrate both grid-based core e-Science infrastructure and cloud-based infrastructures provisioned on demand. Grid security architecture is primarily based on the virtual organisations (VO) that are created by the cooperating organisations that share resources (which however remain in their ownership) based on mutual agreement between VO members and common VO security policy. In grids, VO actually acts as a federation of the users and resources that enables federated access control based on the federated trust and security model [24, 25]. In general, the VO-based environment is considered as trusted.

In the clouds, data are sent to and processed in the environment that is not under the user or data owner control and potentially can be compromised either by cloud insiders or by other users sharing the same resource. Data/information must be secured during all processing stages – upload, process, store and stream/visualise. Policies and security requirements must be bound to the data, and there should be corresponding security mechanisms in place to enforce these policies.

The following problems/challenges arise from the cloud IaaS environment analysis for security services/infrastructure design:

- Data protection both stored and "on-wire" that includes, besides the traditional confidentiality, integrity, access control services and also data lifecycle management and synchronisation
- Access control infrastructure virtualisation and dynamic provisioning, including dynamic/automated policy composition or generation
- Security services lifecycle management, in particular, service-related metadata and properties, and their binding to the main services
- Security sessions and related security context management during the whole security services lifecycle, including binding security context to the provisioning session and virtualisation platform
- Trust and key management in provisioned on-demand security infrastructure and support of the dynamic security associations (DSA) that should provide fully verifiable chain of trust from the user client/platform to the virtual resource and the virtualisation platform
- SLA management, including initial SLA negotiation and further SLA enforcement at the planning and operation stages

The security solutions and supporting infrastructure to support the data integrity and data processing security should provide the following functionalities:

- Secure data transfer that possibly should be enforced with the data activation mechanism
- Protection of data stored on the cloud platform
- Restore from the process failure that entails problems related to secure job/application session and data restoration

The security solutions and supporting infrastructure should support consistent security session management:

- Special session for data transfer that should also support data partitioning and run-time activation and synchronisation

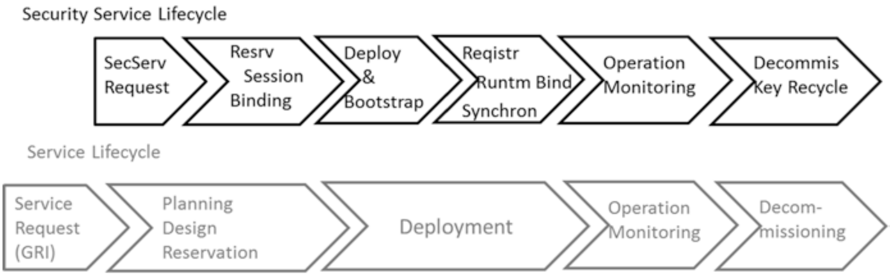


Fig. 5.9 The proposed security services lifecycle management model

- Session synchronisation mechanisms that should protect the integrity of the remote run-time environment
- Secure session failover that should rely on the session synchronisation mechanism when restoring the session

Wider cloud adoption by industry and their integration with advanced infrastructure services will require implementing manageable security services and mechanisms for the remote control of the cloud operational environment integrity by users.

5.4.2 Security Services Lifecycle Management Model (SSLM)

Most of the existing security lifecycle management frameworks, such as defined in the NIST Special Publication 800-14 "Generally Accepted Principles and Practices in Systems Security" [26], provide a good basis for security services development and management, but they still reflect the traditional approach to services and systems design driven by engineers. The defined security services lifecycle includes the following typical phases: initiation, development/acquisition, implementation, operation/maintenance and disposal.

Figure 5.9 illustrates the proposed security services lifecycle management (SSLM) model [5] that reflects security services operation in generically distributed multi-domain environment and their binding to the provisioned services and/or infrastructure. The SSLM includes the following stages:

- Service *request* and generation of the GRI that will serve as a provisioning session identifier (SessionID) and will bind all other stages and related security context [6, 7]. The request stage may also include SLA negotiation which will become a part of the binding agreement to start on-demand service provisioning.
- *Reservation* stage and *reservation session binding* with GRI (also a part of the general SDF/SLM) that provides support for complex reservation process including required access control and policy enforcement.
- *Deployment* stage (including *Bootstrapping*) begins after all component resources have been reserved and includes distribution of the security context and binding

Table 5.1 Relation between SSLM/SLM stages and supporting general and security mechanisms

SLM/SDF stages	Request	Planning Reservation	Deployment	Operation	Decommissioning
SSLM Process/ Activity	SLA Negotiation	Serv/Rsr Compose Reserve	Configure Bootstrap Synchron	Orchestration / Session Management	Logoff Accounting
Supporting Mechanisms (M– mandatory, O- optional)					
SLA	M				O
Workflow		O		M	
Metadata	M	M	M	M	
Dynamic Security Association		O	M	M	
AuthZ SecCtx Management		M	M	M	
Logging		O	O	M	M

the provisioned virtualised resources and hosting platform to the GRI as a provisioning session ID.

- *Registration and synchronisation* stage (including *run-time binding*) that allows the whole virtual infrastructure to start synchronously and specifically targets possible scenarios with the provisioned services migration or failover. In a simple case, the registration stage binds the local resource or hosting platform run-time process ID to the GRI as a provisioning session ID.
- During *operation* stage, the security services provide access control to the provisioned services and maintain the service access or usage session.
- *Decommissioning* stage ensures that all sessions are terminated, data are cleaned up and session security context is recycled.

The proposed SSLM model is compatible with the above-described SDF and extends the existing SLM frameworks with the additional stages "registration and synchronisation" that specifically target such security issues as the provisioned services/ resources restoration (in the framework of the active provisioning session) and provide a mechanism for remote data protection by binding them to the session context.

Table 5.1 explains what main processes/actions take place during the different SLM/SSLM stages and what general and security mechanisms are used:

- SLA – used at the stage of the service request placing and can also include SLA negotiation process.
- Workflow is typically used at the operation stage as service orchestration mechanism and can be originated from the design/reservation stage.
- Metadata are created and used during the whole service lifecycle and, together with security services, actually ensure the integrity of the SLM/SSLM.
- Dynamic security associations support the integrity of the provisioned resources and are bound to the security sessions.

- Authorisation session context supports integrity of the authorisation sessions during reservation, deployment and operation stages.
- Logging can be actually used at each stage and essentially important during the last 2 stages – operation and decommissioning.

The proposed SSLM model extends the existing SLM frameworks with the additional stages "reservation session binding" and "registration and synchronisation" which especially target such scenarios as the provisioned services/resources restoration, re-planning or migration (in the framework of the active provisioning session) and provide a mechanism for remote data protection by binding them to the session context. Important role in these processes belongs to the consistent security context management and dynamic security associations that should be supported by dynamic trust anchors binding and special bootstrapping procedure or protocol. However, it is perceived that implementing such functionality will require the service hosting platform that supports Trusted Computing Group Architecture (TCG Architecture) [27, 28].

5.4.3 Dynamically Provisioned Access Control Infrastructure (DACI)

Developing a consistent framework for dynamically provisioned security services requires deep analysis of all underlying processes and interactions. Many processes typically used in traditional security services need to be abstracted, decomposed and formalised. First of all, it is related to the security services setup, configuration and security context management that in many present solutions/frameworks is provided manually, during the service installation or configured out-of-band.

The general security framework for on-demand provisioned infrastructure services should address two general aspects: (1) supporting secure operation of the provisioning infrastructure which is typically provided by the providers' authentication and authorisation infrastructure (AAI) supported also by the federated identity management services (FIdM) and (2) provisioning a dynamic access control infrastructure as part of the provisioned on-demand virtual infrastructure. The first task is primarily focused on the security context exchanged between involved services, resources and access control services. The virtualised DACI must be bootstrapped to the provisioned on-demand VI and VIP/VIO trust domains as entities participating in the handling initial request for VI and legally and securely bound to the VI users. Such security bootstrapping can be done at the deployment stage. [AU7]

Virtual access control infrastructure setup and operation is based on the above-mentioned DSA that will link the VI dynamic trust anchor(s) with the main actors and/or entities participating in the VI provisioning – VIP and the requestor or target user organisation (if they are different). As discussed above, the creation of such DSA for the given VI can be done during the reservation and deployment stage. The reservation stage will allow the distribution of the initial provisioning session

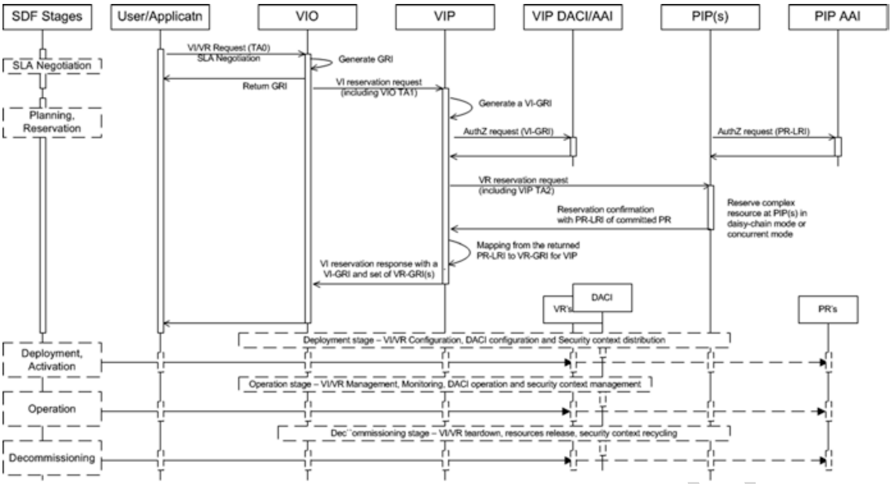


Fig. 5.10 Security context management during VI provisioning and operation

context and collection of the security context (e.g. public key certificates) from all participating infrastructure components. The deployment stage can securely distribute either shared cryptographic keys or another type of security credentials that will allow validating information exchange and apply access control to VI users, actors and services.

Figure 5.10 illustrates in detail the interaction between main actors and access control services during the reservation stage and includes also other stages of provisioned infrastructure lifecycle. The request to create VI (RequestVI) initiates a request to VIP that will be evaluated by VIP-AAI against access control policy, which will next be followed by VIP request to PIP for required or selected physical resources PRs, which in its own turn will be evaluated by PIP-AAI. It is an SDF and SSLM requirement that starting from the initial RequestVI all communication and access control evaluations should be bound to the provisioning session identifier GRI. The chain of requests from the user to VIO, VIP and PIP can also carry corresponding trust anchors TA0…TA2, for example, in a form of public key certificate (PKC) [29] or WS-Trust security tokens [30].

DACI is created at the deployment stage and controls access to and use of the VI resources; it uses dynamically created security association of the users and resources. The DACI bootstrapping can be done either by fully preconfiguring trust relations between VIP-AAI and DACI or by using special bootstrapping registration procedure similar to those used in TCG Architecture [22].To ensure unambiguous session [AU8] context and the identification of all involved entities and resources, the following types of identifiers are used:

- Global reservation ID (GRI) – generated at the beginning of the VI provisioning, stored at VIO and returned to user as identification of the provisioning session and the provisioned VI

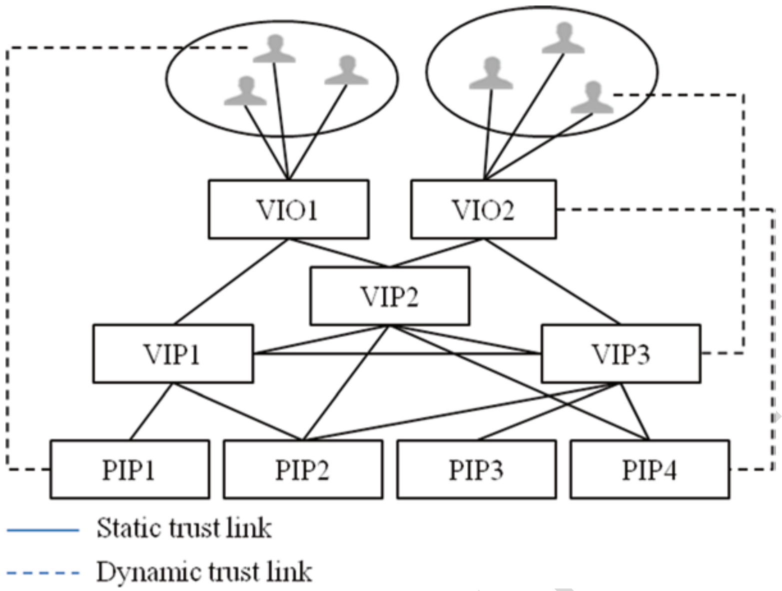


Fig. 5.11 Trust relationships in multi-provider cloud environment

- VI-GRI – generated by VIP as an internal reservation session ID, which can be also refolded GRI, depending on the VIP provisioning model
- Local reservation ID (LRI) that can be generated by PIP or VIP to provide identification PR-LRI and VR-LRI of the committed or created PR@PIP and VR@VIP

5.4.4 Dynamic Security Associations Management

5.4.4.1 Trust Relations

Figure 5.11 describes relations between entities in the cloud infrastructure services provisioned on demand. PIPs own virtualised physical devices to offer virtual resources (VRs). VIPs are intermediate providers to compose and aggregate VRs from multiple PIPs into the virtual infrastructures (VIs), which are subscribed by VIOs. The end-users then may consume VRs in the VI associated with VIOs' identifier. The involved actors form the cloud supply-chain service model from low-level providers (PIPs) to intermediate providers (VIPs), subscribers (VIOs) and end-users.

Providing trust between parties is basic for security services. This model has two types of trust relationships. The first one is static or direct trust between two direct parties based on SLA agreements. The second one is the dynamic trust, the trust relation established during provisioning stages between indirect parties (i.e. VIO and PIPs, VI-end-users and VIPs). These relationships are dynamic because they are established and released during the VI provisioning phases.

According to various models in distributed systems including public key cryptography models (e.g. PKI or PGP) and recommendation-based models, trust relationships are assumed not transitive [31]. For example, if A trusts B and B trusts C, it cannot conclude that A trusts C. In some specific conditions, the trust could be transitive [30] and A could trust C. In our approach, we select the transitive trust between parties as specified in [30] with a set of conditions, for example, with VI-end-users, VIO and VIP, VIO trusts VIP and recommends the trust to VI-end-users. VI-end-users then trust VIO as the recommender for trust relationships and could judge VIO's recommendations. With the above cloud supply-chain service model, they form recommendation paths or trust paths from PIP to VIP, VIO and VI-end-users. This dynamic trust model can follow one of the following categories. The first one is evidence-based model where entities establish a trust relationship based on evidence, such as cryptographic keys. The other one is recommendation-based model [AU9] [32]. For clouds, we propose to use the evidence-based model because direct/static trust relations are enforced by SLA along with specific cryptographic parameters that can be provided as a provisioning session security context. Dynamic trust relations are established based on direct trust relations and other assumptions as specified above to satisfy conditional transitive trust.

5.4.4.2 Establishing Dynamic Trust Relationships

A trust relationship between two entities is described by a security association [AU10] (SA). It contains agreed security attributes between parties. The SA could include cryptographic parameters (certificate, keys, algorithms, etc.) to make sure one end-point assure about other one on its trustworthiness.

The direct/static trust relations described in the previous section are known as the static security association (SSA), while the dynamic trust relations can be defined as the dynamic security association (DSA). In the reference model, SSAs include SSA (VI-user, VIO), SSA (VIO, VIP) and SSA (VIP, PIP). Set of DSAs include DSA (VI-end-user, VIP), DSA (VI-end-user, PIP) and DSA (VIO, PIP).

Generic steps to establish dynamic trust relationship are as follows:

Conditions: SSA (A, B), SSA (B, C)
Goal: Establish the DSA (A, C)
Procedures:

1. A asks B to establish trust to C.
2. B retrieves its SA list to find SSA (B, A) and SSA (B, C). It then generates a new SA. This SA is sent back to A and C by protecting with SSA (B, A) and SSA (B, C), respectively.
3. A receives the generated SA. By verifying the SSA (B, A) protector, it adds the new generated SA to its SA list as the DSA (A, C).

4. C receives the generated SA and verifies it with SSA (C, B). Since it is valid, C adds the new SA, known as DSA (C, A), to its SA list.

For specific mechanisms such as PKI, PGP or SAML [33], the procedure needs to be modified to generate SA dynamically and sent to both indirect parties A and C. Further development of these mechanisms will require additional research.

5.4.5 Security Infrastructure Bootstrapping Protocol

This section describes the proposed security bootstrapping protocol that was proposed in authors' papers [25] and [7] and currently being implemented in the framework of the GEYSERS project [16].

DACI trust model relies on a number of trust anchors residing at PIP, VIP and VIO and rooted in the VI provisioning request or SLA between user/customer and VI/cloud provider (in our model, VIP or VIO). However, to protect it from compromise (e.g. by cloning) and make it integrity protected, it needs to be bootstrapped to the virtualisation platform run-time environment. The proposed bootstrapping protocol is using a Trusted Computing Platform Architecture (TCG Architecture) and Trusted Platform Module (TPM) which can provide a trustworthy platform from which secure systems may be built. They can provide a static root of trust to allow booting a system to a known and trusted state by taking measurements and verifying each piece of software before it is executed [34].

In order to create a trusted computing environment, it is necessary to build an unbroken chain of trust from the most fundamental hardware (such as the BIOS and firmware) through to the operating system and virtualisation platform that hosts virtualised services and the DACI itself. The TPM can be configured to take measurements of each software component before it is executed. Only if the signature is valid will the system proceed. Software needs to be specifically designed to take advantage of these capabilities; as an example, such solutions and firmware are provided by Intel [35] and VMware [36].

The initial TPM-based platform initiation uses a special method for remote TPM attestation called direct anonymous attestation (DAA) [37] that actually requires a third-party role (the issuer) [26] that can be a part of cloud provider security infrastructure.

In order to authenticate the TPM-enabled system, the service provider would provide a signed package that contains relevant TPM public keys, system keys and valid trusted states for those machines. Next, a special Vanguard application is sent to a remote machine via the SCP protocol as an initial stage in the required service deployment. It determines the safety of the remote machine before more sensitive information or software is transferred to it. As part of the bootstrapping process a Vanguard application verifies the identity and state of the remote machine based on the fingerprint provided in the security package.

After verification, a trusted platform session token can be generated based on GRI or LRI that is then sealed by the TPM. It is included as a part of the general VI or DACI security context and can only be decrypted by the same TPM and only when in the same state [38]. This prevents the session from being decrypted on another machine and in effect binds the session to the machine in a trusted state. In order to defeat a cloning attack, an encryption key or other metadata can also be sealed to a TPM. When used to encrypt disk images, this prevents the images from being decrypted on another untrusted machine.

5.4.6 Security Context Management in DACI

Although DACI operates at the operation stage of the SSLM/SLM, its security context is bound to the overall provisioning process starting from initial stage of the service request and SLA negotiation that will provide a trust anchor TA0 to user/application security domain with which the DACI will interact during the operation stage. The RequestVI initiates the provisioning session inside of which we can also distinguish two other types of sessions: reservation session and access session, which however can use that same access control policy and security context management model and consequently can use the same format and type of the session credentials. In the discussed DACI, we use the authorisation token (AuthzToken) mechanism initially proposed in the GAAA-NRP framework and used for authorisation session context management in multi-domain network resource provisioning [39, 40]. Tokens as session credentials are abstract constructs that refer to the related session context stored in the provisioned resources or services. The token should carry session identifier, in our case GRI or VI-GRI.

When requesting VI services or resources at the operation stage, the requestor needs to include the reservation session credentials together with the requested resource or service description which in its own turn should include or be bound to the provisioned VI identifier in a form of GRI or VI-GRI. The DACI context handling service should provide resolution and mapping between the provided identifiers and those maintained by the VIP and PIP, in our case VR-LRI or PR-LRI. If session credentials are not sufficient, for example, in case delegation or conditional policy decision is required, all session context information must be extracted from AuthzToken and the normalised policy decision request will be sent to the DACI policy decision point (PDP) which will evaluate the request against the applied access control policy.

In the discussed DACI architecture, the tokens are used both for access control and signalling at different SSLM/SDF stages as a flexible mechanism for communicating and signalling security context between administrative and security domains (that may represent PIP or individual physical resources). Inherited from GAAA-NRP, the DACI uses two types of tokens:

- Access tokens that are used as AuthZ/access session credentials and refer to the stored reservation context.
- Pilot tokens that provide flexible functionality for managing the AuthZ session during the reservation stage and the whole provisioning process. Few types of the pilot token are defined that can communicate different domain-related context information during the services or resources reservation stage.

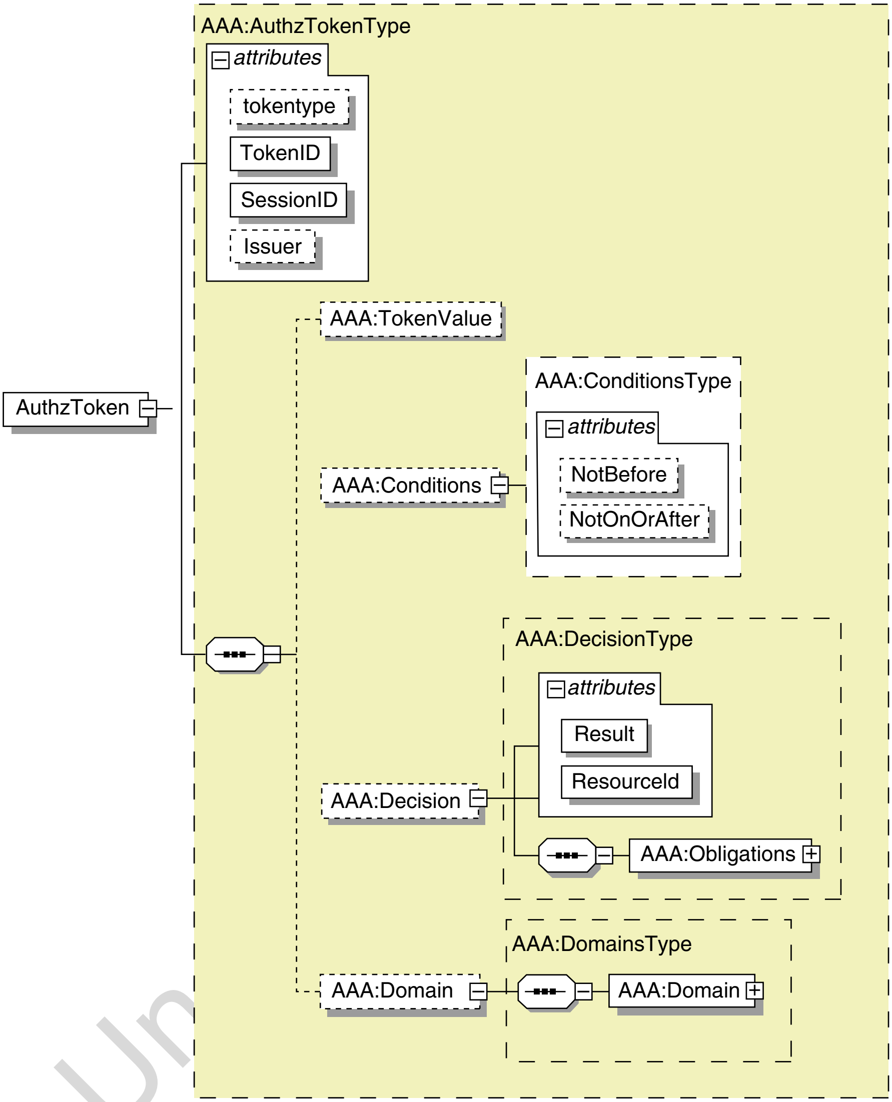


Fig. 5.12 Common access and pilot token data model (**a**) and example of the XML token (**b**)

Figure 5.12 illustrates the common data model of both access token and pilot token. Although the tokens share a common data model, they are different in the operational model and in the way they are generated and processed. When processed by the AuthZ service components, they can be distinguished by the token type attribute which is optional for access token and mandatory for pilot token.

(a) High-level access and pilot token data model

```
<AAA:AuthzToken
xmlns:AAA="http://www.aaauthreach.org/ns/AAA"
```

```
        Issuer="http://testbed.ist-
phosphorus.eu/phosphorus/aaa/TVS/token-pilot"
        SessionId="0912182e7f9c7d156028e77e3d6b460de8e4
937c"
    TokenId="a99b91e70307bdd329c9a0aec18bb4a3"
type="pilot-type3">
<AAA:TokenValue>3923c7ecb979e7078ab8745191a7b25348cdc
b48</AAA:TokenValue>
   <AAA:Conditions NotBefore="2008-07-25T09:38:39.890Z"
        NotOnOrAfter="2008-07-26T09:38:39.890Z"/>
   <AAA:DomainsContext>
  <AAA:Domain     domainId="http://testbed.ist-
phosphorus.eu/viola">
          <AAA:AuthzToken   Issuer="http://testbed.ist-
phosphorus.eu/viola/aaa/TVS/token-pilot"
SessionId="b0b6202d7bd7fb7b591b5de29950d21fdb8bf375"
        TokenId="e7c88fad8cff42d7faaa961b96411ae6">
   <AAA:TokenValue>f09194bbddeef95bc4acb187f71b0bb20b2d
0b44</AAA:TokenValue>
<AAA:Conditions                     NotBefore="2008-07-
18T21:55:15.296Z"
           NotOnOrAfter="2008-07-18T21:55:15.296Z"/>
  </AAA:AuthzToken>
  <AAA:KeyInfo>http://testbed.ist-
phosphorus.eu/viola/_public_key_</AAA:KeyInfo>
     </AAA:Domain>
   </AAA:DomainsContext>
  </AAA:AuthzToken>
```

(b) Example XML token type 3 containing domain-related context that may include the pilot token and key information from the previous domain

Access tokens contain three mandatory elements: the SessionId attribute that holds the GRI, the TokenId attribute that holds a unique token ID attribute and is used for token identification and authentication and the TokenValue element. The optional elements include: the condition element that may contain two time validity attributes notBefore and notOnOrAfter, the decision element that holds two attributes ResourceId and result, and optional element obligations that may hold policy obligations returned by the PDP. Pilot tokens may contain another optional domains element that serves as a container for collecting and distributing domain-related security context.

For the purpose of authenticating token origin, the pilot token value is calculated of the concatenated strings "DomainId, GRI, TokenId". This approach provides a simple protection mechanism against pilot token duplication or replay during the same reservation/authorisation session. The following expressions are used to calculate the TokenValue for the access token and pilot token:

```
TokenValue = HMAC(concat(DomainId, GRI, TokenId), TokenKey)
```

In the current implementation [40], the TokenKey is generated from the GRI and a common shared secret value among all trusted domains. It means that only these domains can generate valid tokens and correspondingly verify the authenticity of the received tokens. The shared secret can be distributed as a part of the DSA creation. It is also suggested that all participating resources and/or cache domains [AU11] receive tokens and check their uniqueness.

5.5 Security Token Service for Federated Access Control to Provisioned Cloud Infrastructure

Consistent access control to the provisioned cloud infrastructure services requires security mechanisms that should allow federated access control and identity management to potentially multi-domain and multi-provider cloud resources from the user organisational or residential domains. Such functionality is generically provided by the GEMBus security token service (STS) that complies with the related WS-Security standards such as WS-Trust and WS-Federation [30, 41]. The STS is a mechanism that conveys security context information between services that may reside in different security and administrative domains. STS can issue and validate security tokens and support service identity federation and federated identity delegation.

Figure 5.13 depicts an example of the messages exchanged when a user attempts to access a service using tokens to secure the connection. First, the service consumer initialises and sends an authentication request to STS. The STS then validates the consumer credentials and issues a security token to it. With the token, the consumer sends a request message including the token to the producer. The consumer sends the token to STS to check its validity. After running its validation process, the STS sends a response with the status of the token to the producer, which processes it and replies to the consumer.

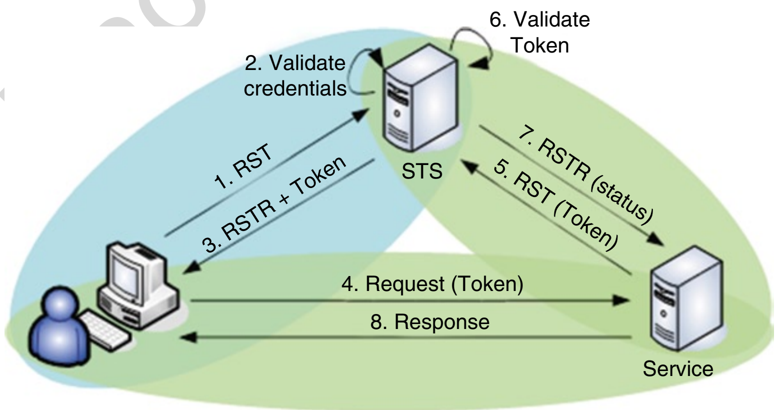


Fig. 5.13 STS operation in federated access control to multi-domain resources

The two different architectural elements are defined for token issuance and validation: the ticket translation service (TTS) responsible for generating valid tokens according to the received credentials, renewing and converting security tokens, and authorisation service (AS) that performs token validation and can retrieve additional attributes or policies from other sources to perform the validation.

The GEMBus STS can be used in both cases as part of the provider access control infrastructure or provisioned and deployed as part of the delivered cloud infrastructure that is managed by user where GEMBus is used as a platform for on-demand services provisioning and management.

5.5.1 STS Functionality and Standard Compliance

Security mechanisms must comply with requirements that may conflict with security, privacy and simplicity of use. It is important that the security protocols deal with user attributes and related information in an appropriate manner, taking the conservative disclosure of attributes and abiding to user privacy policies whenever possible. It is also important that these directives are enforced by all entities, both in the infrastructure itself and in the participant services, dealing with user data in a consistent manner. From the point of view of services, it is very important to protect information by ensuring the identity of consumers who use the services. The most adequate manner to satisfy these requirements relies on the use of a token that allows the transfer of security data along the exchanged messages.

The mechanisms needed to provide secure communications within the GEMBus architecture base their operation on the STS. This service, described in WS-Trust, makes it possible to issue and validate security tokens. The GEMBus STS supports the WS-Trust interoperability profile defined by the EMI, and support for other profiles can be easily added.

Web Services Security (WS-Security) is a communication protocol that provides the means for applying security to Web Services. It is part of the WS-* family of Web service specifications published by OASIS. It is a flexible and feature-rich extension to SOAP to apply security to Web Services. The protocol specifies how integrity and confidentiality can be enforced on messages. It allows the communication of various security token formats, such as SAML [33], Kerberos [42] and X.509 [29], though the protocol is able to accommodate practically any kind of token format. Its main focus is the use of XML Signature [43] and XML Encryption [44] to provide end-to-end security. The protocol is officially called WSS and associated with other specifications like WS-Trust, WS-SecureConversation [45] and WS-Policy [46].

WS-Trust provides extensions to WS-Security, specifically dealing with issuing, renewing and validating security tokens, as well as how to establish, assess (the presence of) and broker trust relationships between participants in a secure message exchange. WS-Trust defines:

- The concept of a STS: A Web service that issues security tokens as defined in the WS-Security specification

- The formats of the messages used to request security tokens and the responses to those messages
- Mechanisms for key exchange

5.5.2 STS Operational Models

In what relates to establishing the identity of a requesting party, it is important to take into account that not only the identity of the entity performing the actual request must be established. Being able to identify the original requestor (the one the requesting party is acting on behalf of) is crucial as well. In this respect, we can reduce the possible situations to two basic models: star model and chain model, suggesting possibility of more complex combination of both (see Fig. 5.14).

In the star model (Fig. 5.14a), the final user identifies at a client endpoint, which acts as consumer of the requested services on behalf of them by connecting to the appropriate service producer endpoints. Therefore, a single statement (or its translations into the required formats thereof) can be used to identify the consumer and the original requesting user. The figure illustrates this architecture, in the case of using SOAP for transport requests and an SAML token to express security statements.

In the chain model (Fig. 5.14b), the final user identifies at a consumer endpoint, which sends an initial request on behalf of them requesting a service to a first service producer endpoint, which then forwards the request to a second producer endpoint, and this to a third one, and thus successively. Therefore, the initial statement (built by the original consumer endpoint) needs to be forwarded as requests are passed from one service endpoint to the next in the chain. The statement must contain information about the original user and the initial consumer endpoint and should contain information about the service endpoints the request has been forwarded through.

(a) Star operational model [AU12]

(b) Chain model

The AS in the figures above refers to a service taking care of validating the security statements received within a certain request. It relies on the use of security tokens along with requests to transfer relevant identity statements plus the availability of a service (provided by the infrastructure itself) able to verify the validity of the security tokens. If a common token format is used or, conversely, a service able to generate appropriate tokens by translating among equivalent ones is available, there are two distinct phases in securing service access in the general case:

1. Token request and generation, that it is up to the local mechanism that the user decides to employ, as long as a minimal set of requirements on level of assurance (in several aspects: identity assessment, required credentials, strength of the link to the individual, etc.) is fulfilled
2. The validation of the token received by the requested service, probably using some of the statements inside the token to retrieve additional attributes from trusted sources and/or to request an access decision from a policy decision point

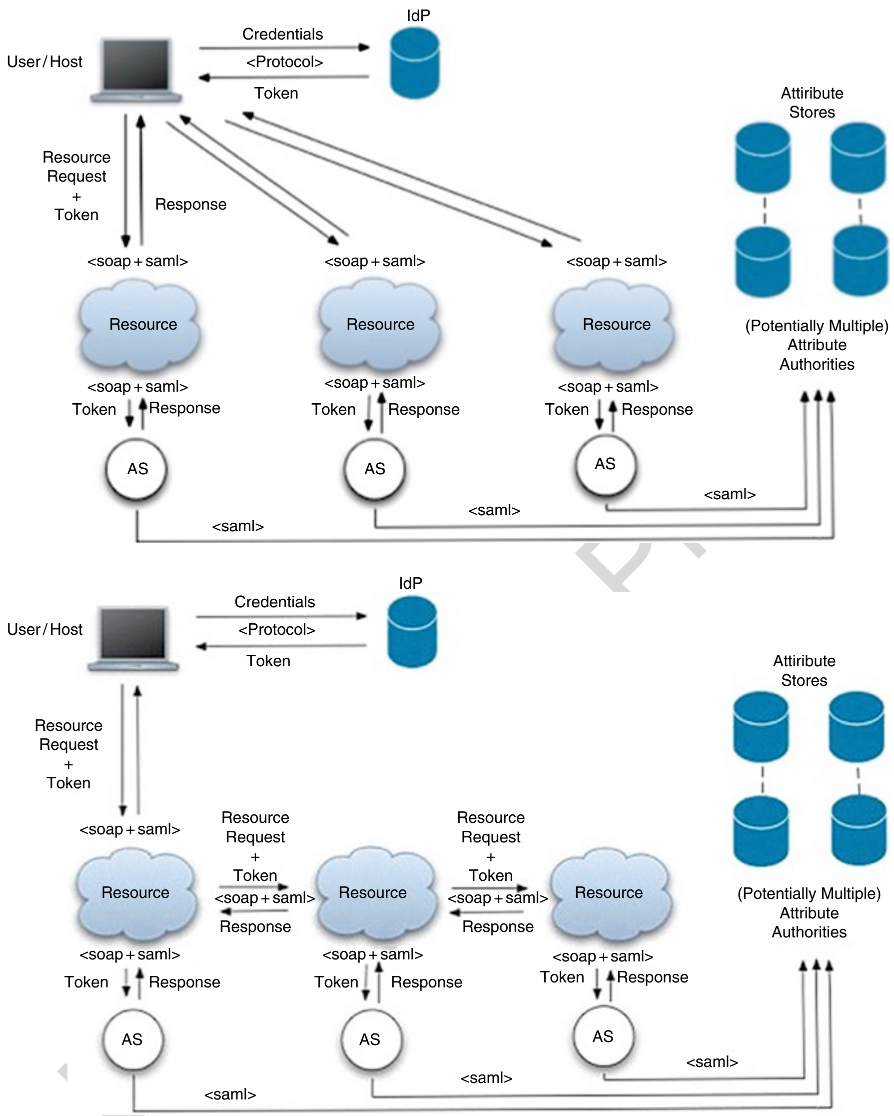


Fig. 5.14 STS operational models: (**a**) star; (**b**) chain

In conclusion, the GEMBus security architecture requires:

- A common token format to guarantee interoperability at the security level
- A service able to act as the source of such tokens and provide a way to translate other token formats into the common format
- A service able to validate security tokens and to provide authorisation decisions

In accordance to these requirements and as said above, two different architectural elements are defined for token issuance and validation in the GEMBus STS. The ticket translation service (TTS) is responsible for generating, renewing and transforming valid tokens in the system, while the authorisation service (AS) performs token validation.

The TTS mostly relies on external identity providers that must verify the identity of the requester based on valid identification material. To support a large amount of services, the application of different authentication methods must be ensured. This must include the support of currently standardised authentication methods as well as methods incorporated in the future. In particular, GEMBus has imbedded support for the eduGAIN identity federation services [47], eduPKI [48], TERENA Certificate Service (TCS) [49] and other International Grid Trust Federation (IGTF) [50] accredited identity infrastructures.

The AS is responsible for checking the validity of the presented tokens. In this case, the requester is usually a service that has received a token along with a request message and needs to check the validity of the token before providing a response. Checks carried out on the token can be related to issue date, expiration date or signature(s). This process can also be associated with more complex processes of authorisation that imply attribute request and check security policies. If the token is valid, the AS provides an affirmative answer to the service.

5.5.3 STS Token Formats

The WS-Security specification allows a variety of signature formats, encryption algorithms and multiple trust domains. It is open to various security token models, such as X.509 certificates, userid/password pairs, SAML assertions and custom-defined tokens.

The GEMBus TTS supports the transformations among different token formats, according to service descriptions as stored in the GEMBus registry by means of the appropriate profile identifiers. Nevertheless, the canonical GEMBus security token (applicable by default in all GEMBus-supported exchanges) is the relayed-trust SAML assertion originally defined within the GN2 project [45] to provide identity information in scenarios where a service is acting on behalf of a user identified through an identity federation.

The SAML construct used in this case is able to convey information about the user accessing the producer. It fulfils two essential constraints:

- It is bound to the consumer by the original identity provider (IdP) that identified the requesting user, so it is possible to check that the information it contains about the user has been legally obtained.
- It is bound to the producer by the consumer, so a potentially malicious producer cannot use this information to further impersonate either the consumer or the user.

To comply with these two requirements, the token consists of an SAML assertion expressing data related to the user authentication with:

- A valid audience restricted to the producer(s) it is addressed to, through an SAML condition element containing an identifier uniquely associated with them
- A statement expressing that this specific method of relayed trust must be used to evaluate the assertion, through a specific value in the SAML construct identifying the subject confirmation method
- The identity assertion(s) received from the IdP as evidence for this confirmation process, as part of the SAML element SubjectConfirmationData

A sample SAML assertion following the above procedures for a consumer with the identifier:

`urn:geant:edugain:component:perfsonarclient:NetflowClient10082`

Acting on behalf of a user identified at the IdP:

`urn:geant:edugain:be:uninett:idp1`

And connecting to a consumer identified by:

`urn:geant:edugain:component:perfsonarresource:netflow.uninett.no/data`

Should have an SAML 2.0 content as the one displayed below (some line breaks and indentation added to improve readability):

```
<?xml version="1.0" encoding="UTF-8"?>
<Assertion
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 xsi:schemaLocation="urn:oasis:names:tc:SAML:2.0:assertion"
 Version="2.0" ID="100001"
 IssueInstant="2006-12-03T10:00:00Z">
 <Issuer>
 urn:geant:gembus:security:sts:gemsts
 </Issuer>

<!-- An audience restriction, that will restrict this security token to be valid for one single resource only.
-->
 <Conditions>
  <AudienceRestriction>
   <Audience>
    urn:geant:edugain:component:perfsonarresource:
    netflow.uninett.no/data
   </Audience>
```

```
   </AudienceRestriction>
    </Conditions>

   <Subject>
    <NameID>aksjc7e736452829we8</NameID>
    <SubjectConfirmation
   Meth-od="urn:geant:edugain:reference:relayed-trust">
   <SubjectConfirmationData>
     <Assertion
      xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
      xmlns:xsi="http://www.w3.org/2006/XMLSchema-
     instance"
      Version="2.0" ID="_200001"
      IssueInstant="2006-12-03T10:00:00Z">
      <Issuer>
      urn:geant:edugain:be:uninett:idp1
      </Issuer>

<!-- This inner assertion is limited to only be valid for
the client performing the WebSSO authentication. This
inner assertion cannot be reused or used at all by others
than the NetflowClient10082 instance. But NetflowClient10082
can use it as an evidence when used inside an assertion
issued by NetflowClient10082 using the relayed-trust
confirmationMethod. -->
      <Conditions>
       <AudienceRestriction>
        <Audience>
         urn:geant:edugain:component:perfsonarclient:
        NetflowClient10082
        </Audience>
       </AudienceRestriction>
      </Conditions>

<!-- This is the inner Subject and authNstatement prov-
ing the authentication itself.
  These elements and attributes must be identical in the
inner and outer assertion:
     - Assertion/Subject/NameID
     - Assertion/AuthnStatement@AuthenticationMethod
The inner assertion confirmation Method must be
     urn:oasis:names:tc:SAML:1.0:cm:bearer. -->
      <Subject>
```

```
     <NameID>aksjc7e736452829we8</NameID>
     <SubjectConfirmation Meth-
    od=”urn:oasis:names:tc:SAML:2.0:cm:bearer”/>
    </Subject>
    <AuthnStatement AuthnInstant=”2006-12-
   03T10:00:00Z”>
      <AuthnContext>
       <AuthnContextClassRef>
    urn:oasis:names:tc:SAML:2.0:ac:classes:Password
      </AuthnContextClassRef>
     </AuthnContext>
    </AuthnStatement>
<!-- Enveloped Signature for SubjectConfirmation -->
<Signature>
  <!-- Signed by the IdP -->
   <SignedInfo>
   <CanonicalizationMethod Algorithm=”…”/>
   <SignatureMethod Algorithm=”…”/>
   <Reference>
    <DigestMethod Algorithm=”…”/>
    <DigestValue/>
   </Reference>
  </SignedInfo>
  <SignatureValue/>
</Signature>
</Assertion>
</SubjectConfirmationData>
</SubjectConfirmation>
</Subject>

<Signature>
<!-- Signed by TTS -->
   <SignedInfo>
    <CanonicalizationMethod Algorithm=”…”/>
    <SignatureMethod Algorithm=”…”/>
    <Reference>
     <DigestMethod Algorithm=”..”/>
     <DigestValue/>
    </Reference>
   </SignedInfo>
  <SignatureValue/>
  </Signature>
</Assertion>
```

5.5.4 TTS and AS

The ticket translation service (TTS) is responsible for issuing, renewing and converting security tokens, responding to consumer requests for issuing, renewing or converting security tokens for services that require it.

Each of these operations can only be done by the TTS, unlike token validation that can be offloaded in certain cases from the security service, the own service or at the framework integration elements such as interceptors, message routers or binding components, especially when session tokens (as described below) are used to simplify interactions.

The main TTS operations are:

- Issuing: To obtain a security token from an identity credentials (identity token)
- Renewing: To renew an issued security token
- Converting: To convert a security token type to another security token type

The TTS operation is as follows:

1. The consumer obtains an identity token (SAML assertion, grid proxy certificate token, etc.) from an identity infrastructure. Typically, the consumer requires users to send such a token in order to provide access.
2. The consumer sends a request for issuance, renewal or conversion to the TTS using either the identity token (issuance) or a security token (renewal or conversion).
3. The STS validates the consumer's token (using security policies) and sends a security token to the consumer.

The authorisation service (AS) is responsible for supporting the token validation functions, responding to requests for validating tokens of consumers and services that require it.

The token validation process can be performed by the AS itself or act as a proxy redirecting the validation process to the external service that generated it. For external validation, the authorisation service may query an external service or IdP and forwards the response to the consumer. When the authorisation service itself performs validation, the process must verify the information contained in the token checking the issuer, issue and expiration date, signatures, etc. In addition to the token, the authorisation service can perform a more complex authorisation process, retrieving attributes related to the token subject and consulting a policy decision point (PDP) for authorisation decisions.

As described in the previous section, the architecture proposed by GEMBus is based on message exchanges performed by different services that can be connected in many ways. Since the ESB is the main integration mechanism provided by GEMBus and it can also act as a container, it is possible to develop and deploy a service directly on the bus. But it is more interesting to exercise its integration capabilities, such as interceptors, message routers and binding components. Whether deployed inside the bus or running as an external service, the STS can be used in a service composition to transparently provide its capabilities, using the above-mentioned mechanisms.

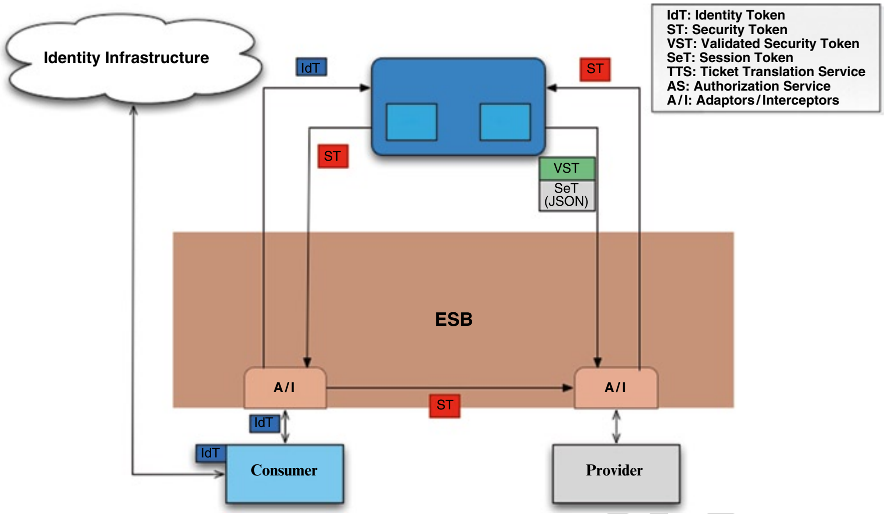


Fig. 5.15 STS extended operation with support of the session tokens

Figure 5.15 illustrates a scenario in which a security token service extended with support for session tokens is integrated in the GEMBus architecture. In this example, the consumer obtains an identity token (e.g. an SAML assertion) from an identity infrastructure. Then it sends an authentication request to the STS using the identity token. The STS validates the consumer identity token and issues a security token (ST) to the consumer. With the new token, the consumer sends a request message to the provider that is intercepted by an element that extracts the ST and sends a token validation request to the STS. The AS module validates the consumer token and issues a response with a validated security token with an optional session token (SeT). Finally, the interceptor passes the message to the provider. It processes the consumer request and sends the response message to the consumer.

5.5.5 Session Management

Session management is the process of keeping track of consumer activity across different levels of interaction with the producer.

Assuming that each message to a service is attached with a token that the service must validate at the authorisation service, this will very likely mean a high workload for the security services and additional delays in service provision. The objective of managing GEMBus sessions is to speed up the security system performance without compromising security goals.

There are several mechanisms to strengthen the validation of the tokens based on the idea of sessions: It is possible to include a new type of token called session token that is returned to the requester after successful validation in the AS. The main feature of this type of token is rapid validation at the expense of lower security features compared to a normal token, though this can be alleviated (if not solved) by reducing its lifetime. When the requester makes a new request for validation to the AS, it can include the two tokens or just the session token. When the AS receives the query, it first checks the session token and, if it is valid, it can respond directly to expedite the process. The GEMBus STS employs a lightweight yet powerful session token format based on JWT, much faster to parse and validate. There are plans to extend this format to make them fully valid security tokens.

Another type of optimisation can be applied to the token validation mechanism done by the AS by making the AS temporarily store a reference to each validated token. Within a given validity period, whenever the AS receives a request for the same token, it does not make a full revalidation. The idea is close to the use of a cache, providing a performance enhancement similar to the use of session tokens, and with the additional advantage of not involving changes in the requesters that make use of the AS.

A JWT session token example looks like this:


```
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiJ1c
m46Z2VhbnQ6ZWR1Z2Fpbjpjb21wb25lbnQ6cGVyZnNvbmFycmVzb3
VyY2U6bmV0Zmxvdy51bmluZXR0Lm5vXC9kYXRhIiwiaX_
NzIjoidXJuOmdlYW50OmVkdWdhaW46Y29tcG9uZW50OnBlcmZzb
25hcmNsaWVudDpOZXRmbG93Q2xpZW50MTAwODIiLCJpYXQiOjEzM-
jA0MDQ0MDk2MzAsImF0dHIiOnt9LCJleHAiOjEzMjA0MDgw
MDk3MTR9.UGl_PoSyd45QqY7m4IoQj9rDdIt3IvXfHRYSa27I1
JbKacI6bDTLewn_0JUuUjeKJoEwQ0MX9KmnT2M1ZD1lRhFGPFhhXm
5MyHNPSC7v9ruzXqk89M8MWbJwpo9elIh8aG4gPGcpGIIuHJ2VLHHDI
IstnX4Z83XfTjg4RHzLkWCRzwzbb4hkIvx6vAPNcGhcC5CfERa
opI6qiDJzpNE_StaU_BI0POUa_3BZU0mVoV4gc_fV_gJipCHXER0z
8rrRBqDuS1Alw2hxBmM2adMTQz9Zk0FlW_74WLMVVHysjltk7Vn4oEc
phXNl54wg1A8sKk6uaIZaH6oI1-f_oDtfA
```


This token is divided in three parts (header, claims and signature), all of them base64 encoded. The header and claims contain the following information: [AU13]


```
<?xml version="1.0" encoding="UTF-8"?>
//JWT Header
  {
   "typ": "JWT",
   "alg": "RS256"
  }
  //JWT Claims
  {
   "aud":
"urn:geant:edugain:component:perfsonarresource:netflow.uninett.no\/data",
   "iss": "urn:geant:gembus:security:sts:gemsts",
   "iat": 1320404409630,
   "attr": {},
   "exp": 1320408009714
  } </Issuer>
```

where
`typ` – type of token, normally JWT
`alg` – algorithm used to sign and verify, in this case, RSA with SHA256
`aud` – represents the audience restriction
`iss` – token issuer
`iat` – issue instant
`exp` – expiration time
`attr` – attributes contained in the token.

The token can contain more claims such as nbf (not before condition) and custom claims. The signature represents the base64-encoded header and claims parts concatenated by a dot.

5.6 Future Research Directions

This chapter presents the ongoing research on developing architecture and framework for dynamically provisioned security services as part of the provisioned on-demand infrastructure services. The presented results provide a good basis for further research in the few important directions that should lead to the problem solution including architecture, information models, required security services, mechanisms and protocols and implementation platform.

Consistent security services implementation and operation require well-defined general infrastructure definition and design, which is considered by authors as a necessary part of the further research on cloud security architecture. Currently existing cloud architecture frameworks are primarily oriented toward business-oriented applications and service delivery from the cloud provider to the user. Internal cloud implementation by cloud providers remains behind the "cloud curtain" what imposes also limitations on the quality of services control and security of the provisioned cloud environment. Virtualisation technologies used in clouds bring services design and related security problems to a new level and actually allow decoupling of the functional services infrastructure from the physical infrastructure and platform. To achieve the same level of the security assurance in virtual infrastructure as in physical infrastructure, many currently adopted security models need to be revisited and re-factored to support new requirements originating from the distributed virtualised environment in clouds.

The following main topics are identified as further research topics related to both general cloud architecture and cloud security architecture:

- Defining new relational models in the provisioning of cloud-based infrastructure services that should reflect different ownership, administration and use relation between main actors in the current cloud services provisioning process such as provider, operator, broker, carrier, customer (enterprise) and user
- Extending the composable services architecture to reflect different virtualisation techniques for compute, storage and network components of the provisioned virtualised infrastructure, defining CSA control and management functionality

- Extending the GEMBus middleware platform to support full functionality of the cloud PaaS model for SOA-based services, in particular, creation of the dynamically configured infrastructure security services that can be used by user application in the provisioned on-demand services
- Extending the infrastructure services modelling framework to include security-related attributes into the services composition and management information base
- Extending dynamic access control infrastructure, currently defined for infrastructure level access control, to integrate it with the user access control using federated user campus or enterprise identity and account
- Further definition and development of the DACI trust management model and virtual infrastructure bootstrapping protocol

5.7 Conclusion

The primary focus of this chapter is the security infrastructure for cloud-based infrastructure services provisioned on demand that in fact should be a part of the overall cloud infrastructure provisioned on demand. The proposed solutions should allow moving current enterprise security infrastructure that currently requires large amount of manual configuration and setup to fully functional virtualised infrastructure service.

To provide the background for defining security infrastructure, the authors provide an overview and short description of the proposed architectural framework for on-demand provisioned cloud-based infrastructure services that includes such components as the infrastructure services modelling framework (ISMF), the composable services architecture (CSA) and the service delivery framework (SDF).

This chapter discusses conceptual issues, basic requirements and practical suggestions for provisioning dynamically configured security infrastructure services. This chapter describes the proposed dynamically provisioned access control infrastructure (DACI) architecture and defines the necessary security mechanisms to ensure consistent security services operation in the provisioned virtual infrastructure. Practical implementation of DACI reveals a wide spectrum of problems related to the distributed access control, policy, trust management and related security context management. In particular, this chapter discusses the use of the security token service for federated inter-domain access control and identity management, authorisation tokens for security context exchange during provisioning session in multi-domain and multi-provider environment.

Consistent security services design, deployment and operation require continuous security context management during the whole security services lifecycle, which must be aligned to the main provisioned services lifecycle. The proposed security services lifecycle management (SSLM) model addresses security problems specific for on-demand infrastructure service provisioning that can be solved by introducing special security mechanisms to allow security services synchronisation and their binding to the virtualisation platform and run-time environment. This chapter discusses how these security mechanisms can be implemented by using the TCG Architecture and functionality of Trusted Platform Module that are currently available in almost all computer platforms and supported by most of VM management platforms. This chapter also describes the proposed security infrastructure bootstrapping protocol that uses TPM functionality and can be integrated with DACI.

The proposed DACI and its component functionalities are currently being developed and implemented in the framework of the two EU projects GEYSERS and GEANT3.

Acknowledgement This work is supported by the FP7 EU-funded project GEANT3 (FP7-ICT-238875) and the FP7 EU-funded integrated project the Generalised Architecture for Dynamic Infrastructure Services (GEYSERS, FP7-ICT-248657).

References

1. NIST SP 800-145: The NIST definition of cloud computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Accessed 29 Jan 2012
2. NIST SP 500-292: Cloud computing reference architecture, v1.0. http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf. Accessed 29 Jan 2012
3. Demchenko, Y., Mavrin, A., de Laat, C.: Defining generic architecture for cloud infrastructure as a service provisioning model. In: Proceedings CLOSER2011 Conference, Nordwijk, Netherlands, 7–9 May 2011. SciTePress (2011). ISBN 978-989-8425-52-2
4. Demchenko, Y., van der Ham, J., Ghijsen, M., Cristea. M., Yakovenko, V., de Laat, C.: On-demand provisioning of cloud and grid based infrastructure services for collaborative projects and groups. In: Proceedings of the 2011 International Conference on Collaboration Technologies and Systems (CTS 2011), Philadelphia, PA, USA, 23–27 May 2011
5. Demchenko, Y., de Laat, C., Lopez, D.R., Garcia-Espin, J.A.: Security services lifecycle management in on-demand infrastructure services provisioning. In: Proceedings of the IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, IN, USA, pp. 644–650 (2010)
6. Demchenko, Y., Ngo, C., de Laat, C., Wlodarczyk, T., Rong, C., Ziegler, W.: Security infrastructure for on-demand provisioned cloud infrastructure services. In: Proceedings of the 3rd IEEE Conference on Cloud Computing Technologies and Science (CloudCom2011), Athens, Greece, 29 Nov–1 Dec 2011 (2011). ISBN 978-0-7695-4622-3
7. Ngo, C., Membrey, P., Demchenko, Y., de Laat, C.: Security framework for virtualised infrastructure services provisioned on-demand. In: Proceedings of the 3rd IEEE Conference on Cloud Computing Technologies and Science (CloudCom2011), Athens, Greece, 29 Nov–1 Dec 2011 (2011). ISBN 978-0-7695-4622-3
8. European Grid Infrastructure (EGI). https://www.egi.eu/. Accessed 9 Nov 2011
9. NIST-SP 500-291: NIST cloud computing standards roadmap. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909024. Accessed 29 Jan 2012
10. OASIS reference architecture foundation for service oriented architecture 1.0, Committee Draft 2, 14 Oct 2009. http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf (2009). Accessed 9 Nov 2011
11. Pautasso, C., Zimmermann, O., Leymann, F.: RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision, 17th International World Wide Web Conference (WWW2008), Beijing, China (2008)

12. Chappell, D.: Enterprise Service Bus. O'Reilly, Beijing/Cambridge (2004)
13. OSGi service platform release 4, version 4.2. http://www.osgi.org/Download/Release4V42. Accessed 9 Nov 2011
14. TMF service delivery framework. http://www.tmforum.org/servicedeliveryframework/4664/home.html. Accessed 9 Nov 2011
15. TMF software enabled services management solution. At http://www.tmforum.org/BestPracticesStandards/SoftwareEnabledServices/4664/Home.html. Accessed 9 Nov 2011
16. Generalised architecture for dynamic infrastructure services (GEYSERS Project). http://www.geysers.eu/. Accessed 9 Nov 2011
17. OWL 2 web ontology language. http://www.w3.org/TR/owl2-overview/. Accessed 9 Nov 2011
18. van der Ham, J., Dijkstra, F., Grosso, P., van der Pol, R., Toonk, A., de Laat, C.: A distributed topology information system for optical networks based on the semantic web. Elsevier J. Opt. Switch. Netw. **5**(2–3), 85–93 (2008)
19. GEANT project. http://www.geant.net/pages/home.aspx. Accessed 9 Nov 2011
20. GEMBus architecture, GEANT3 project report deliverable DJ3.3.2, Jan 2011
21. Fuse ESB: OSGi based ESB. http://fusesource.com/products/enterpriseservicemix/#documentation. Accessed 9 Nov 2011
22. Apache ServiceMix an open source ESB. http://servicemix.apache.org/home.html. Accessed 9 Nov 2011
23. Spring security. Reference documentation. http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity-single.html. Accessed 9 Nov 2011
24. Demchenko, Y., de Laat, C., Koeroo, O., Groep, D.: Re-thinking grid security architecture. In: Proceedings of the IEEE Fourth eScience 2008 Conference, Indianapolis, USA, 7–12 Dec 2008, pp. 79–86. IEEE Computer Society Publishing, Los Alamitos (2008). ISBN 978-0-7695-3535-7/ISBN 978-1-4244-3380-3
25. Foster, I., Kishimoto, H., Savva, A., Berry, D., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., Von Reich, J.: GFD.80 The Open Grid Services Architecture, Version 1.5. Open Grid Forum, 5 Sept 2006
26. NIST SP 800-14: Generally accepted principles and practices for securing information technology systems. National Institute of Standards and Technology. September 1996. http://csrc.nist.gov/publications/nistpubs/800-27/sp800-27.pdf (1996). Accessed 29 Jan 2012
27. TCG Infrastructure Working Group reference architecture for interoperability. Specification ver. 1.0. 16 June 2005. http://www.trustedcomputinggroup.org/specs/IWG/IWG_Architecture_v1_0_r1.pdf (2005). Accessed 9 Nov 2011
28. Demchenko, Y., Gommans, L., de Laat, C.: Extending user-controlled security domain with TPM/TCG in grid-based virtual collaborative environment. In: Proceedings of the International Symposium on Collaborative Technologies and Systems, Orlando, FL, USA, 2007, pp. 57–65
29. RFC5280 Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) Profile. May 2008. http://www.ietf.org/rfc//rfc5280 (2008). Accessed 9 Nov 2011
30. Web services trust language (WS-Trust). ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf. Accessed 9 Nov 2011
31. Li, H., Singhal, M.: Trust management in distributed systems. Computer **40**(2), 45–53 (2007)
32. Abdul-Rahman, A., Hailes, S.: A distributed trust model. In: Proceedings of the 1997 Workshop on New Security Paradigms – NSPW'97, Langdale, Cumbria, UK, pp. 48–60 (1997)
33. Assertions and protocols for the OASIS security assertion markup language (SAML) V2.0, OASIS Standard, 15 March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf (2005). Accessed 9 Nov 2011
34. Fisher, D., McCune, J.M., Andrews, A.D.: Trust and Trusted Computing Platforms. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2011)
35. Intel hardware technologies to secure clouds. http://www.intel.com/content/www/us/en/enterprise-security/processors-with-built-in-cloud-security.html. Accessed 9 Nov 2011
36. Intel cloud builders guide for enhancing server platform security with VMWare. http://www.intel.com/Assets/PDF/general/icb_ra_cloud_computing_VMware_TCP.pdf. Accessed 9 Nov 2011

37. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security – CCS'04, Washington DC, p. 132 (2004)
38. Parno, B.: The Trusted Platform Module (TPM) and Sealed Storage (2007) [AQ14]
39. Demchenko, Y., Wan, A., Cristea, M., de Laat, C.: Authorisation infrastructure for on-demand network resource provisioning. In: Proceedings of the 9th IEEE/ACM International Conference on Grid Computing (Grid 2008), Tsukuba, Japan, 29 Sept–1 Oct 2008, pp. 95–103 (2008). ISBN 978-1-4244-2579-2
40. GAAA Toolkit pluggable components and XACML policy profile for ONRP. Phosphorus Project Deliverable D4.3.1, 30 September 2008. http://www.ist-phosphorus.eu/files/deliverables/Phosphorus-deliverable-D4.3.1.pdf. Accessed 9 Nov 2011
41. Web services federation language (WS-Federation), version 1.0, 8 July 2003. http://msdn.microsoft.com/ws/2003/07/ws-federation/ (2003). Accessed 9 Nov 2011
42. RFC4120 The Kerberos network authentication service (V5). http://www.ietf.org/rfc/rfc4120.txt. Accessed 9 Nov 2011
43. XML-signature syntax and processing. W3C recommendation, 10 June 2008. http://www.w3.org/TR/xmldsig-core/. Accessed 9 Nov 2011
44. XML encryption XML encryption syntax and processing. W3C recommendation, 10 December 2002. http://www.w3.org/TR/xmlenc-core/ (2002). Accessed 9 Nov 2011
45. Web services secure conversation language (WS-SecureConversation). http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-secureconversation.asp. Accessed 9 Nov 2011
46. Web services policy framework (WSPolicy), version 1.2, March 2006. http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf. Accessed 9 Nov 2011
47. eduGAIN – GEANT federated authentication and authorisation infrastructure. http://www.geant.net/service/edugain/pages/home.aspx. Accessed 9 Nov 2011
48. EduPKI GEANT PKI service. https://www.edupki.org/. Accessed 9 Nov 2011
49. TERENA Certificate Service (TCS). http://www.terena.org/activities/tcs/. Accessed 9 Nov 2011
50. The International Grid Trust Federation. http://www.igtf.net/. Accessed 9 Nov 2011

Recommended Reading

For interested readers, it is recommended to become familiar with the general background information related to both cloud technologies and basic security models and standards. In particular, the following additional literature can be recommended.

First of all, it is recommended to read NIST standards on cloud computing and virtualisation technologies in which up-to-date list is available at the NIST Cloud Program webpage (http://www.nist.gov/itl/cloud/):

NIST SP 800-145, "A NIST definition of cloud computing". http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

NIST SP 500-292, Cloud Computing Reference Architecture, v1.0. http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf

DRAFT NIST SP 800-146, Cloud Computing Synopsis and Recommendations. http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf

Draft SP 800-144 Guidelines on Security and Privacy in Public Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf

DRAFT NIST SP 800-293, US Government Cloud Computing Technology Roadmap, Volume I, Release 1.0. http://www.nist.gov/itl/cloud/upload/SP_500_293_volumeI-2.pdf

NIST SP500-291 NIST Cloud Computing Standards Roadmap. http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/StandardsRoadmap/NIST_SP_500-291_Jul5A.pdf

SP 800-125 Guide to Security for Full Virtualisation Technologies.
http://csrc.nist.gov/publications/nistpubs/800-125/SP800-125-final.pdf

For the background security, read the following literature:

These RFCs on the generic AAA Authorisation framework provide a general context for developing authorisation infrastructure for on-demand provisioned services and access control infrastructure:

RFC2903 Generic AAA Architecture Experimental RFC 2903, Internet Engineering Task Force, August 2000. ftp://ftp.isi.edu/in-notes/rfc2903.txt

RFC 2904 AAA Authorization Framework. Internet Engineering Task Force, August 2000.ftp://ftp.isi.edu/in-notes/rfc2904.txt

Cloud computing technologies with their distributed virtualised computing environment motivate revisiting foundational security concepts and models and rethinking existing security models and solutions. The following foundation publications on computer security (proposed for mainframe-based computing model) can be recommended:

Anderson, J.: Computer Security Technology Planning Study. ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA 01731 (Oct. 1972) [NTIS AD-758 206]. http://csrc.nist.gov/publications/history/ande72.pdf

Bell. DE., La Padula, L.: Secure Computer System: Unified Exposition and Multics Interpretation. ESD-TR-75-306, ESD/AFSC, Hanscom AFB, Bedford, MA 01731 (1975) [DTIC AD-A023588]. http://csrc.nist.gov/publications/history/bell76.pdf

Biba K.J.: Integrity Considerations for Secure Computer Systems. MTR-3153, The Mitre Corporation, Apr 1977

Anderson, R., Stajano, F., Lee, J:. Security Policies. http://www.cl.cam.ac.uk/~rja14/Papers/security-policies.pdf

Author Queries

Chapter No.: 5 0001545000

Query	Details Required	Author's Response
AU1	Please confirm corresponding author and provide email id for the author.	
AU2	Please check if edit to the sentence starting "The proposed security…" is ok.	
AU3	Please check if edit to the sentence starting: "Cloud "elasticity", as recognised…" is ok.	
AU4	Please check if edit to the sentence starting: "The CSA defines…" is ok.	
AU5	Please check if edit to the sentence starting "It may include…" is okay.	
AU6	Please check if edit to Fig. 5.8 caption is ok.	
AU7	Please check if edit to the sentence starting: "The general security framework…" is ok.	
AU8	Please check if edit to the sentence starting "To ensure unambiguous…" is ok.	
AU9	Please check if edit to the sentence starting "For clouds, we…" is ok.	
AU10	Please check the sentence starting "The SA could…" for sense.	
AU11	Please check if edit to the sentence starting: "It is also suggested…" is ok.	
AU12	Please check if the text "(a) Star operational model and (b) Chain model" can be deleted from here.	
AU13	Please check if edit to the sentence starting "The header and…" is okay.	
AU14	Please provide complete details for the reference [38].	