Architecture de sécurité pour les grands systèmes ouverts, répartis et hétérogènes

Syed Salar Hussain Naqvi

Ecole Doctorale
d'Informatique,
Télécommunications
et Electronique
de Paris

Thèse

présentée pour obtenir le grade de Docteur
de l'École Nationale Supérieure des Télécommunications

Spécialité : **Informatique et Réseaux**

Syed Salar Hussain Naqvi

Architecture de Sécurité pour les Grands Systèmes, Ouverts, Répartis et Hétérogènes

Soutenue le 01 Décembre 2005 devant le jury composé de :

Pascal Urien	Président
Ken Chen	Rapporteurs
Ana Cavalli	
Marcel Soberman	Examinateurs
André Cotton	
Michel Riguidel	Directeurs de Thèse
Isabelle Demeure	

Acknowledgements

At the outset, I would like to express my sincere gratitude to my thesis advisors. Most beneficial to my doctoral research was the vision, direction and significant feedback from my advisor, Professor Michel Riguidel; and the guidance and committed concentration towards technical quality from my co-advisor Professor Isabelle Demeure.

I also thank the honorable members of the jury – Pascal Urien, Ken Chen, Ana Cavalli, Marcel Soberman, and André Cotton – for their attention and thoughtful comments.

During my thesis work, I had the opportunity to work in the EU-funded project SEINIT (Security Expert Initiative). Most importantly, the scope of this project – trusted and dependable security framework, ubiquitous, working across multiple devices, heterogeneous networks, and organization independent (inter-operable) – largely influenced my research. I would like to thank all the participants of this project for their help especially the project coordinator, André Cotton, and technical coordinator, Sathya Rao.

I would like to express my sincere gratitude to Professor Radha Poovendran of the University of Washington who provided me the opportunity to work with him in the Network Security Laboratory during the summer quarter of the year 2005. I learned a lot with him.

I owe gratitude to the members of the networks and computer science department (INFRES) of ENST, notably Gwendal Legrand, with whom I frequently engaged in scientific and technical discussions. I am equally indebted to the members of Network Security Laboratory of the University of Washington, especially Radhakrishna Sampigethaya and Tianwei Chen.

I am grateful to Ann Collins, Marian Bubak, Omer Rana, Rajkumar Buyya, Richard Olejnik, and Vincent Breton for their sincere help and guidance at various occasions during my thesis.

I also thank Peter Weyer-Brown, Lorna Monahan, and Vera Dickman for their invaluable help in shaping and refining my professional communication skills.

I would like to thank administrative and technical staff members of ENST who have been kind enough to help in their respective roles.

I want to thank all my doctoral fellows whose friendly company, during my thesis work, was a source of great pleasure.

Last but certainly not the least, I am proud to acknowledge the generous and enduring support and prayers of my mother, Abida Naqvi, throughout the years of efforts toward this dissertation. I dedicate this thesis to her. I am indebted to my uncle Azhar Zaidi and his family for their precious support and encouragement.

Table of Contents

Résumé

La sécurité des systèmes hétérogènes distribués ouverts à grande échelle d'aujourd'hui (tels que les grilles de calcul, les systèmes P2P, l'informatique omniprésente/ubiquitaire, etc.) est devenue une préoccupation opérationnelle généralisée. Les services de sécurité de pointe et les relations de confiance sont actuellement les caractéristiques les plus recherchées de ces systèmes. Nous avons proposé une architecture de sécurité apte à répondre aux besoins généraux de sécurité de ces systèmes [1]. Nous avons procédé à d'importants travaux de terrain pour déterminer les limitations et les failles des solutions de sécurité actuellement proposées pour ces systèmes et pour établir quels sont les véritables besoins que doit satisfaire l'architecture de sécurité, de manière à réduire les pertes de performances et à assurer une sécurité robuste [2]. Nous avons notamment identifié l'analyse des besoins [3], l'analyse du risque [4], la modélisation des menaces [5] et la faisabilité de mise en œuvre [6].

Le concept de *virtualisation* des services de sécurité est introduit pour les services en question. Il est nécessaire de disposer d'une totale liberté de choix des mécanismes de sécurité sous-jacents. Du point de vue de la sécurité, la virtualisation de la définition d'un service tient compte des besoins de sécurité qui permettent d'accéder à ce service. La virtualisation de la sémantique de sécurité impose d'utiliser des méthodes standardisées de segmentation des composantes de la sécurité (par exemple, authentification, contrôle d'accès, etc.) et de proposer des méthodes standardisées permettant de fédérer plusieurs mécanismes de sécurité. La virtualisation permet à chaque terminaison participante d'exprimer la politique qu'elle souhaite voir appliquer lorsqu'elle s'engage dans un échange sécurisé avec une autre terminaison [7]. Les politiques peuvent spécifier quels sont les mécanismes d'authentification pris en charge, le degré d'intégrité et de confidentialité requis, les politiques de confiance et de confidentialité, ainsi que d'autres contraintes de sécurité. Ce concept de virtualisation des services de sécurité peut être réalisé au moyen de moteurs virtuels distribués qui permettront d'unifier les appels au service de sécurité en fonction des besoins et non pas en fonction des technologies à prendre en charge.

Un mécanisme *configurable* d'appel des services de sécurité est proposé pour répondre aux besoins de sécurité des différentes catégories d'utilisateurs. Cette approche permet de faire évoluer l'infrastructure de sécurité avec des effets moindres sur les fonctionnalités de gestion des ressources, qui sont encore en pleine phase d'évolution. En outre, elle permet aux utilisateurs et aux fournisseurs de ressources de configurer l'architecture de sécurité en fonction de leurs besoins et de leur niveau de satisfaction. Cet ensemble de services de sécurité comprend des services de sécurité de base (authentification, autorisation, mappage des identités, audit, etc.), ainsi que des services de sécurité contemporains (contrôle d'accès mobile, signature numérique dynamique, etc.) [8].

1. Cadre de recherche

Depuis le début des années 1980, les entreprises se sont habituées à coopérer à travers des réseaux d'ordinateurs. Cette forme de coopération, très statique, prenait à ses débuts la forme d'échanges de données électroniques (EDI) [9]. Depuis qu'Internet est utilisé pour les transactions commerciales, des formes de coopération plus dynamiques sont rendues possibles. Cela étant, les besoins de sécurité des systèmes basés sur Internet sont très différents de ceux des réseaux traditionnels. Par exemple, Internet ne propose aucune infrastructure centralisée pour assurer la sécurité des réseaux. Les besoins de sécurité sont particulièrement critiques en cas d'utilisation de liaisons ultra-rapides visant à combiner des ressources de calcul réparties. Le meilleur exemple de ce type d'environnement collaboratif distribué est la grille de calcul [10]. Un compte-rendu d'étude publié par Virginia Tech à l'automne 2002 indique que plus de la moitié des membres de la communauté de grille pensent que les solutions existantes de sécurité de la grille ne constituent pas un service

adéquat pour les communautés de grille collaboratives. Les raisons invoquées vont de l'*absence de modèle de la menace sous-jacente* à la complexité et au coût des relations de confiance actuellement nécessaires entre sites [11]. Pour Sun Microsystems, l'adoption de grilles globales, où les sociétés partagent des ressources matérielles et logicielles pour atteindre un objectif de calcul, a été ralentie du fait des problèmes de sécurité et de l'absence de standards [12].

Au début de notre siècle, plusieurs agences de financement de la recherche ont souligné la nécessité d'immenses efforts de recherche visant à aboutir à l'excellence scientifique et technique en matière de sécurité, de fiabilité et de résistance des systèmes, des services et des infrastructures, tout en répondant aux besoins de confidentialité et de confiance [13, 14]. Nous avons répondu à ces appels en entamant un travail sur l'évolution de l'architecture de sécurité de pointe des systèmes hétérogènes distribués et ouverts à grande échelle. Ce travail de recherche est directement et indirectement soutenu par ces agences de financement de la recherche.

2. Architecture de sécurité

Nous avons identifié les composantes logiques, factorisé les caractéristiques communes et définies les interfaces globales de l'architecture de sécurité que nous proposons pour les systèmes hétérogènes distribués et ouverts à grande échelle [15]. L'architecture globale qui en a résulté est représentée en Figure 1. Une courte description des différentes composantes de notre projet est proposée dans la présente section [16].

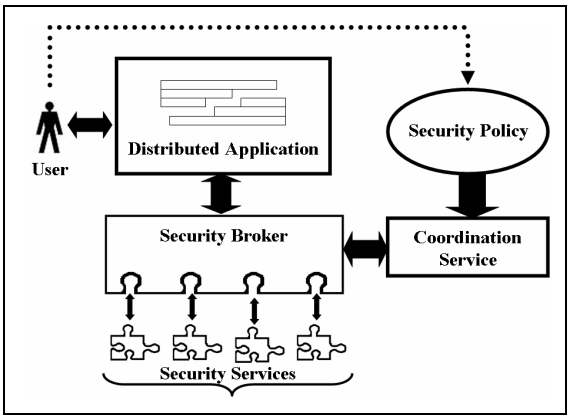

Fig 1 : Architecture globale

2.1. Architecture à Base de Courtier de Sécurité

Le Courtier de Sécurité sert de médiateur entre les applications (plus précisément, entre les applications distribuées) et les services de sécurité. Le courtier de sécurité possède un gestionnaire des services de sécurité, qui permet d'absorber l'hétérogénéité des services de sécurité sous-jacents et de fournir une interface homogène à la couche supérieure. Des moteurs virtuels distribués sont mis en œuvre à l'aide d'un agent de courtage des services de sécurité. L'idée d'introduire un courtier de services de sécurité est en fait inspirée de l'utilisation d'un agent de courtage pour l'exploitation des ressources de calcul/stockage adéquates (également appelé « courtier de ressources) dans les applications distribuées. L'architecture en couches du courtier de sécurité proposé est représentée en Figure 2. Les fonctionnalités associées à ces couches sont les suivantes :

Application/Client Interface	
Configuration Daemon	
Security Services Handler	Protocol Mapping
Security Architecture Interface	

Fig 2 : Architecture à base de Courtier de Sécurité

L'*Interface Application/Client* authentifie l'utilisateur/l'application et crée le lien entre l'utilisateur/l'application et l'infrastructure de courtier de sécurité sous-jacente pour leur permettre de communiquer l'un avec l'autre.

Le *Démon de Configuration* est un serveur de configuration. Il accepte une demande de configuration abstraite, indépendante de la machine, puis interagit avec le service de coordination à travers un canal sécurisé. Il signale le moment venu que le service de

coordination a approuvé la configuration du service de sécurité. Il peut tenir à jour un journal des configurations, voire gérer une configuration de sauvegarde complète.
Le *Gestionnaire des Services de Sécurité* absorbe la diversité des mécanismes de sécurité pour permettre d'unifier les appels au service de sécurité en fonction des besoins et non pas en fonction des technologies à prendre en charge.
Le *Mappage des Protocoles* contient la liste complète des protocoles pris en charge par l'architecture de sécurité à travers le Gestionnaire des Services de Sécurité.
L'*Interface de l'Architecture de Sécurité* est constituée de modules « *sockets* » permettant de « brancher » divers services de sécurité. L'appel à un service de sécurité donné est envoyé au gestionnaire des services de sécurité à travers le Démon de Configuration. Le gestionnaire des services de sécurité vérifie l'existence du service de sécurité en question à partir du mappage des protocoles de sécurité et, s'il existe effectivement, il appelle une instance chargée de relier le service de sécurité correspondant à l'interface de l'architecture de sécurité.
Les *Algorithmes Temps Réel* sont utilisés pour répondre aux problèmes de performances. Ils garantissent que la totalité du traitement du courtier de sécurité puisse se faire en temps réel et que les utilisateurs/services puissent appeler ces services de sécurité au niveau de la couche de sécurité. Ces caractéristiques temps réel sont mises en œuvre au niveau de chaque couche.

2.2. Architecture du Service de Coordination

Cette composante est chargée de garantir l'appel d'un ensemble coordonné de services de sécurité aux différentes terminaisons du système. Comme le montre la Figure 3, cette composante contient des traces de tous les services appelés au niveau des différents nœuds. Lorsqu'un utilisateur appelle un ensemble de services (ensemble par défaut ou défini par l'utilisateur) qui ne correspond pas à l'ensemble de services appelé par les autres nœuds, le problème est considéré comme un *conflit* entre services appelés. Ce conflit est géré en fonction de la politique de sécurité. Une fois qu'un conflit est résolu, l'appel des services de sécurité se fait au niveau du courtier de sécurité. Il convient de souligner ici que ce courtier de sécurité ne prend pas part à la gestion des conflits proprement dite. En fait, il réachemine l'appel de service effectué par un utilisateur/service, vers le service de coordination, qui devra le mapper et vérifier qu'il n'y a pas conflit avec les services de sécurité appelés par les autres nœuds. Les services de sécurité ne sont appelés par le courtier de sécurité que lorsque ce dernier en reçoit l'ordre du service de coordination.

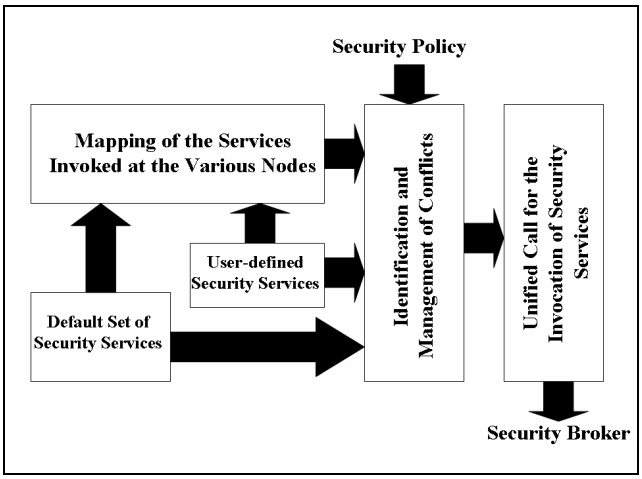

Fig 3 : Architecture du Service de Coordination

Dans l'organisation présentée en Figure 1, le courtier de sécurité est délibérément placé entre l'application et le service de coordination, de manière à les isoler l'un de l'autre. L'un des objectifs du courtier de sécurité est d'isoler l'architecture de sécurité de base des applications, de manière à relever le niveau de protection. La nécessité de protéger le service coordonné est évidente puisque, si un utilisateur/une application malveillant(e) parvient à l'influencer, la non-correspondance entre les différents services de sécurité appelés par les différents nœuds va entraîner l'auto-destruction de l'architecture de sécurité toute entière.

2.3. Politique de Sécurité

L'architecture de sécurité que nous proposons est assortie d'une politique de sécurité en couches, dont les principales caractéristiques sont les suivantes :

1. Mécanismes flexibles de contrôle d'accès régis par la politique de sécurité
2. Politiques de contrôle d'accès inter-domaine
3. Communication de groupe sécurisée
4. Mécanismes de délégation visant à prendre en charge l'évolutivité vers différentes ressources et différents utilisateurs

La politique de sécurité est constituée de deux volets bien distincts : la politique de sécurité « globale » (P_G) et la politique de sécurité « locale » (P_L). Les couches de la politique de sécurité locale sont la politique *d'application*, la politique *de contrôle d'accès*, la politique *d'intégrité des données*, la politique *d'authentification* et la politique *de chiffrement*. La politique de sécurité globale définit la politique de sécurité générale et constitue l'abstraction (la virtualisation) de toutes les politiques de sécurité locales.

2.4. Modèle de Confiance

Nous proposons un modèle de confiance dynamique distribué [17] qui constitue un mécanisme flexible permettant la délégation de la confiance et le suivi continu des changements qui interviennent au niveau de la confiance de chaque nœud. Ce modèle présente les avantages suivants : administration hiérarchisée décentralisée ; évolutivité des possibilités d'émission de certificats ; flexibilité de délégation. Les services ouverts n'étant pas limités à une gamme précise de domaines ou d'organisations, une gestion de la confiance distribuée, flexible et généraliste est nécessaire pour permettre l'établissement d'une relation de confiance entre des entités susceptibles de ne jamais se rencontrer. Un tel système constituerait un mécanisme de contrôle d'accès évolutif et décentralisé sur Internet [18].

Le modèle de confiance que nous proposons est basé sur une approche en deux temps [19] : tout d'abord, définition des relations de confiance directes ou mutuelles entre deux nœuds d'un domaine, ainsi que des relations de confiance indirectes entre les intermédiaires. Ensuite, du fait du caractère dynamique des collaborations, les relations de confiance peuvent également devoir être établies de manière dynamique à l'aide d'intermédiaires sur un support distribué. Notamment, ce modèle doit également définir une base répondant aux besoins de sécurité pour permettre la signature unique et la délégation.

2.5. Reconfigurabilité

Nous avons également étudié la reconfigurabilité/l'adaptabilité des services de sécurité qui leur permettrait de prendre en charge la sécurité des systèmes hétérogènes. Dans notre proposition, les services de sécurité doivent être capables de s'auto-reconfigurer si un nouveau nœud est introduit ou de réagir pour récupérer suite à un quelconque problème réseau. Cela peut être obtenu par l'utilisation d'une architecture à base de composantes dynamiquement reconfigurables. Cette architecture permet aux nœuds de négocier dynamiquement les services de sécurité, les protocoles et la prise en charge cryptographique dont ils ont besoin. Notre but ici est de permettre la configurabilité et la reconfigurabilité de la fonctionnalité des services de sécurité de base, sans avoir à formuler d'hypothèses particulières concernant l'architecture distribuée sous-jacente.

Cette caractéristique présente plusieurs avantages par rapport aux architectures de sécurité classiques :

1. Elle rend l'architecture de sécurité adaptable aux environnements hétérogènes pour lesquels la composition exacte des ressources du système est inconnue au départ. Elle prend donc en charge à chaque instant l'ajout et la suppression dynamiques de ressources du système général.
2. Elle rend l'architecture de sécurité résistante et, de ce fait, assure la capacité de survie de l'ensemble du système. La reconfigurabilité permet au système de récupérer sa

configuration de sécurité d'origine une fois terminé le scénario d'attaque et, par conséquent, relève la qualité de service de l'ensemble.

3. Elle permet au système de supporter les fréquentes évolutions technologiques, ce qui permet d'intégrer facilement de nouveaux dispositifs ou de nouvelles ressources dans les systèmes existants sans que cela modifie l'architecture de base ni n'affecte la qualité de service ou les performances du système. Par exemple, si un utilisateur soumet une demande d'analyse de données à une grille, cette dernière doit exécuter la tâche rapidement, en environnement sécurisé, de manière à éviter tout piratage et à garantir l'exactitude qui s'impose. (Cela étant, cette qualité de service n'était pas une priorité des premières générations de grille, dont l'objectif énoncé était la réalisation de toutes les tâches demandées).

3. Mise en Œuvre et Evaluation Fonctionnelle

Les différents kits d'outils proposés pour la modélisation ou la simulation de systèmes distribués à grande échelle ne prennent pas en charge la simulation des fonctionnalités de sécurité. Nous avons par conséquent développé des modules simulateurs, sous-produits de l'architecture que nous proposons, de manière à en tester et en valider les fonctionnalités [20, 21]. Ces modules simulateurs sont ensuite intégrés dans un kit d'outils existant, *GridSim* [22]. Un prototype de démonstration du contrôle d'accès dynamique par utilisation du contexte et de l'état des utilisateurs hormis leurs caractéristiques conventionnelles, est également mis en œuvre pour démontrer que l'architecture proposée est apte à fonctionner dans le monde réel [23, 24, 25].

De manière à suivre les pratiques standard en matière d'évaluation de la sécurité, nous avons mis au point un *profil de protection* de notre proposition [26] en utilisant la version 2.1 des Critères Communs (Common Criteria (CC) version 2.1) avec le Niveau d'Assurance d'Evaluation (Evaluation Assurance Level ou EAL) *4* (qui fournit l'assurance par une analyse des fonctions de sécurité, en utilisant une spécification d'interface fonctionnelle et complète, une documentation explicative, la conception de haut niveau et de bas niveau de l'Objectif d'Evaluation (Target of Evaluation ou TOE), ainsi qu'un sous-ensemble de la mise en œuvre, de manière à comprendre le comportement de sécurité) et une Puissance de Fonction (Strength of Function ou SOF) *élevée* (cela suppose qu'un agresseur s'attaquant au système dispose du potentiel nécessaire pour attaquer ce dernier).

4. Conclusions et Futures Orientations

Nous avons proposé une nouvelle approche permettant d'affronter différents défis de sécurité présentés par les systèmes hétérogènes distribués et ouverts à grande échelle. La caractéristique la plus marquante de notre démarche est le caractère flexible et adaptable des services de sécurité. Nous avons recouru à la virtualisation pour proposer une méthode standardisée de fédération de plusieurs mécanismes de sécurité hétérogènes.

Pour assurer une fiabilité minimale des fonctionnalités émergentes de gestion des ressources, et pour rendre notre modèle plus adaptable, nous avons étendu le concept de *sécurité en tant que services* à *sécurité en tant que services connectables ('pluggable')*. Les autres caractéristiques sont l'auto-sécurité de l'architecture de sécurité ; le recours au courtier de sécurité qui négocie les services de sécurité ; la description de l'ontologie de sécurité, qui permet l'interaction par protocoles standard des services de bootstrapping de la sécurité de base ; et les services de sécurité centrés sur l'utilisateur, dont l'objectif principal est la possibilité d'utilisation.

Notre recherche a constitué une première étape vers une approche systématique de la conception d'une architecture de sécurité destinée aux systèmes hétérogènes distribués et ouverts à grande échelle. Bien que divers systèmes complexes y soient envisagés, nous avons centré notre attention sur les systèmes à base de grille de calcul [27]. Ce travail pourra être poursuivi en explorant des solutions de sécurité plus spécifiques destinées à d'autres systèmes complexes tels que les systèmes ubiquitaires, les systèmes P2P, etc. En outre, le concept de virtualisation pourrait être étendu pour s'adapter à la législation des

différents pays, aux problèmes éthiques des populations, ou encore aux préoccupations des entreprises. En outre, la virtualisation pourrait être utilisée pour atteindre le meilleur compromis entre garanties de sécurités et capacités de traitement.

Summary

Security of today's large scale, open, distributed heterogeneous systems (such as computational grids, peer-to-peer systems, pervasive/ubiquitous computing, etc.) has become a mainstream operational concern. Establishment of in-depth security services and trust relationships are the most desirable features for such systems. We have proposed a security architecture to address the comprehensive security needs of these systems [1]. Extensive groundwork was carried out to determine the limitations and shortcomings of the existing security solutions for these systems and to establish the real needs of the security architecture in order to reduce performance overheads and to provide robust security [2]. These include requirements analysis [3], risk analysis [4], threat modeling [5], and implementation feasibility [6].

The concept of *virtualization* of security services is introduced for the security services. It is needed to have the absolute freedom to choose the underlying security mechanisms. From the security point of view, the virtualization of a service definition encompasses the security requirements for accessing that service. The need arises in the virtualization of security semantics to use standardized ways of segmenting security components (e.g., authentication, access control, etc.) and to provide standardized ways of enabling the federation of multiple security mechanisms. Virtualization permits each participating end-point to express the policy it wishes to see applied when engaging in a secure conversation with another end-point [7]. Policies can specify supported authentication mechanisms, required integrity and confidentiality, trust, privacy policies, and other security constraints. This concept of virtualization of security services can be realized through distributed virtual engines that will enable security service calls to be unified according to requirements and not according to the technologies to be supported.

A *configurable* mechanism for the invocation of security services is proposed to address security needs of the different kinds of users. This approach permits the evolution of security infrastructure with less impact on the resource management functionalities, which are still on the verge of evolution. Moreover, it permits the users and resource providers to configure the security architecture according to their requirements and satisfaction level. The set of these security services include core security services (such as authentication, authorization, identity mapping, audit, etc.) as well as contemporary security services (such as mobile access control, dynamic digital signature, etc.) [8].

1. Research Context

Businesses have cooperated via computer networks since the early 1980s. These forms of cooperation were very static and took place in the form of electronic data interchange (EDI) [9]. Since the opening of the Internet for commercial use, more dynamic forms of cooperation are facilitated. However, the security needs of Internet-based systems are very different from those of traditional networking. For example, the Internet offers no centralized infrastructure to provide responsibility for network security. The security needs are particularly acute when high speed internets are used to combine widespread computational resources. The best example of such distributed collaborative environment is the computational grid [10]. A survey report of Virginia Tech in the fall of 2002 states that more than half of the grid community members believe that existing grid security solutions do not provide adequate services for collaborative grid communities. The reasons given ranged from the *lack of an underlying threat model* to the complexity and expense of inter-site trust relationships that are currently required [11]. Sun Microsystems says adoption of global grids, where companies share hardware and software resources to accomplish a computational goal, has been slowed because of security concerns and a lack of standards [12].

In the beginning of this century, various research funding agencies emphasized the need for a comprehensive research efforts of building scientific and technical excellences in

security, dependability and resilience of systems, services and infrastructures, whilst meeting demands for privacy and trust [13, 14]. We responded to these calls and started working on the evolution of in-depth security architecture for large scale, open, distributed heterogeneous systems. This research work is directly and indirectly supported by these research funding agencies.

2. Security Architecture

We have identified logical components, factored out common features, and have defined general framework interfaces for our proposed security architecture for large scale, open, distributed heterogeneous systems [15]. This devised framework architecture is shown in figure 1. A concise account of various components of our design is provided in this section [16].

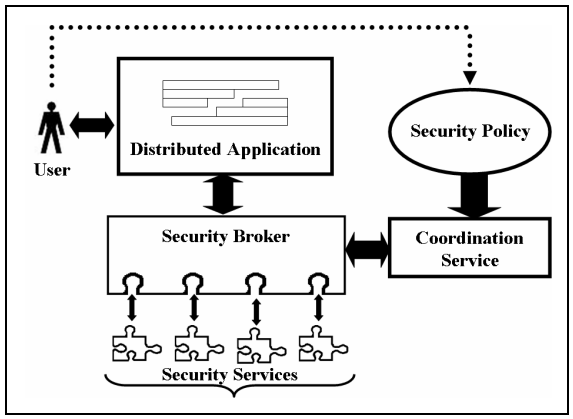

Fig 1: Framework Architecture

2.1. Security Broker Architecture

The Security Broker mediates between applications (more precisely the distributed applications) and the security services. The security broker has a security services handler, which is employed to absorb the heterogeneity of the underlying security services and to provide a homogeneous interface to the upper layer. Distributed virtual engines are implemented by using brokering agent for the security services. The idea of introducing a security services broker is actually inspired by the utilization of a brokering agent for the exploitation of suitable computing/storage resource (also known as the resource broker) in distributed applications. The layered architecture of the proposed security broker is shown in figure 2. The functionalities associated with these layers are:

Application/Client Interface	
Configuration Daemon	
Security Services Handler	Protocol Mapping
Security Architecture Interface	

Fig 2: Security Broker Architecture

Application/Client Interface authenticates the user/application and provides the glue between the user/application and the underlying security broker infrastructure to facilitate communications between them.

Configuration Daemon is a configuration server. It accepts a machine independent, abstract configuration request and then interacts with the coordination service through a secure channel. It notifies when the coordination service approves the security service configuration. It can keep a log of configurations done or even a complete backup configuration.

Security Services Handler absorbs the diversity of the security mechanisms to enable security service calls to be unified according to requirements and not according to the technologies to be supported.

Protocol Mapping contains a comprehensive list of the protocols supported by the security architecture through the Security Services Handler.

Security Architecture Interface consists of *socket* modules to plug various security services. Call for a particular security service is sent to the security services handler through the Configuration Daemon. The security services handler checks the existence of such a security service from the security protocol mapping and if it exists then an instance is invoked to hook the corresponding security service to the security architecture interface.

Real-Time Algorithms are used to address the performance concerns. They assure that the entire processing of the security broker takes place in real time and the users/services can invoke these security services at the application layer. These real-time features are implemented at each layer.

2.2. Coordination Service Architecture

This component is responsible for the surety that a coordinated set of security services are invoked at the various ends of the system. As shown is figure 3, it contains traces of all the services invoked at the various nodes. When a user invokes a set of services (default or user-defined) and it does not match with the set of services invoked at the other nodes then the mismatch is identified as *conflict* in the invoked services which is managed in the light of the security policy. Once the conflict is resolved, security services invocation is made to the security broker. It is worth mentioning here that this security broker is not involved in the conflict management itself, rather it forwards the service invocation, made by a user/service, to the coordination service for its mapping and to look for any conflict(s) with the security services invoked at the other nodes. The security services are invoked by the security broker only when it receives a command from the coordination service.

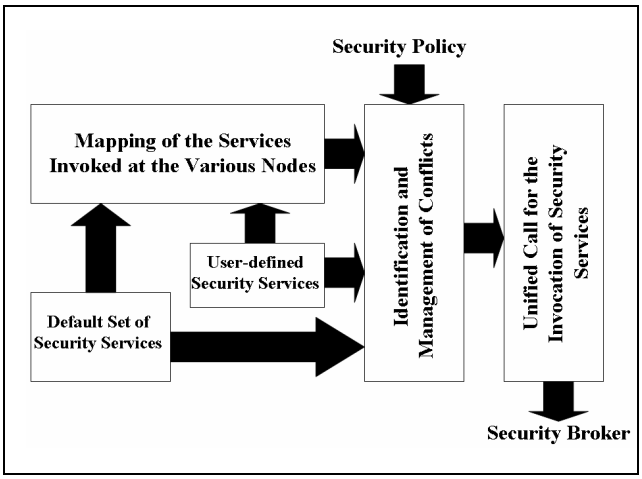

Fig 3: Coordination Service Architecture

In the arrangement shown in figure 1, the security broker is deliberately placed between the application and the coordination service so as to isolate the latter from the former. One of the objectives of the security broker is to isolate the core security architecture from the applications so as to increase the protection level. The need to protect the coordinated service is evident from the fact that if some malicious user/application succeeds in influencing it then the mismatch of the various security services invoked at the various nodes will cause the self-destruction of the entire security architecture.

2.3. Security Policy

We propose a layered security policy in our proposed security architecture. The salient features of this policy include:

5. Flexible policy-based access control mechanisms
6. Inter-domain access control policies
7. Secure group communication
8. Delegation mechanisms to support scalability to large numbers of resources and users

The security policy consists of two distinguished parts: Global Security Policy (P_G) and Local Security Policy (P_L). The Local Security Policy layers are *application* policy, *access control* policy, *data integrity* policy, *authentication* policy and *encryption* policy. The Global Security Policy defines general security policy and provides the abstraction (virtualization) of all Local Security Policies.

2.4. Trust Model

We propose a dynamic distributed trust model [17] that provides a flexible mechanism for delegation of trust and continuous monitoring of the changes to the level of trust of each node. It has the advantage of decentralized hierarchical administration, scalability of

certificate issuing capacity and the flexibility of delegation. Since the open services are not limited to a specific range of domains and organizations, a distributed, flexible and general-purpose trust management is necessary for establishing a trust relationship between entities that may never meet with each other to provide a scalable, decentralized access-control mechanism over the Internet [18].

Our proposed trust model has a two-pronged approach [19]: First, definition of direct or mutual trust relationships between two nodes within a domain, as well as indirect trust relationships traversing intermediaries. Second, due to the dynamic nature of collaborations, trust relationships might also need to be established dynamically using intermediaries in a distributed means. Specially, it should also set up the basis satisfying the security requirements to achieve single sign-on and delegation.

2.5. Reconfigurability

We have also explored the reconfigurability/adaptability of the security services to provide security support for the heterogeneous systems. We propose that the security services should be capable of reconfiguring themselves if some new node is introduced or to react to recover any system problem. It is achieved through the employment of a dynamically reconfigurable component-based architecture. This architecture allows nodes to dynamically negotiate the security services, protocols, and cryptographic support needed. Our motivation here is to enable configurability and reconfigurability of core security services functionality without having to make any particular assumptions about the underlying distributed architecture.

This feature has two advantages over the classical security architectures:

4. It makes the security architecture adaptable to such heterogeneous environments where the ultimate composition of the system resources is unknown in the beginning. Hence it supports the dynamic addition and suppression of resources from the overall system at any time instant.
5. It makes the security architecture resilient and hence assures survivability of the overall system. Reconfigurability makes the system to regain its original security configurations after the attack scenario is over and therefore it improves the quality of service of the entire system.
6. It enables the system to cope up with the frequent technology changes so that new devices and resources are easily integrated into the existing systems without changing the core architecture and without plunging the operation quality of service and performance. For example, if a user submits a request for data analysis to a grid; the grid should perform the task in a timely manner, in a secure environment to avoid tampering, and with all necessary accuracy (though, this quality of service was not a priority in the initial generations of the grid, where just getting it all to work first was the stated goal).

3. Implementation and Functional Assessment

The existing range of toolkits for modeling and simulations of large scale distributed systems does not provide any support for the simulations of security functionalities. So we developed simulator modules, as a by-product of our proposed architecture, to test and validate its functionalities [20, 21]. These simulator modules are then integrated into an existing toolkit *GridSim* [22]. A prototype for the demonstration of dynamic access control by using the context and state of the users beside their conventional credentials is also implemented to prove the real-world functioning of the proposed architecture [23, 24, 25].

In order to follow the standard security evaluation practice, we have prepared a *protection profile* of our proposition [26] by using Common Criteria (CC) version 2.1 with Evaluation Assurance Level (EAL) *4* (that provides assurance by an analysis of the security functions, using a functional and complete interface specification, guidance documentation, the high-level and low-level design of the Target of Evaluation (TOE), and a subset of the implementation, to understand the security behavior) and minimum Strength of Function

(SOF) *high* (that implies that an attacker to the system has the high potential to attack the system).

4. Conclusions and Future Directions

We have proposed a new approach to deal with a number of security challenges presented by large scale, open, distributed heterogeneous systems. The most salient feature of our approach is the flexible and adaptive nature of security services. We have used virtualization to provide standardized ways of enabling the federation of multiple heterogeneous security mechanisms. To have minimal reliance on the emerging resource management functionalities, and to make our model more adaptive, we have extended the concept of *security as services* to *security as pluggable services*. The other features are the self-security of the security architecture; use of security broker that negotiates for security services; description of security ontology to enable standard protocol interactions of core security bootstrapping services; and user-centered security services where usability is the prime motivation.

Our research has been a first step to come towards a systematic approach in the design process of security architecture for large scale, open, distributed heterogeneous systems. Although a wide variety of complex systems are considered but more consideration is given to the computational grid based systems [27]. This work can be continued to explore more specific security solutions for other complex systems such as ubiquitous systems, P2P systems, etc. Moreover, the concept of virtualization could be extended to adapt country-specific legal requirements, population-based ethical issues, and the business-oriented interests. Furthermore, virtualization could be used to achieve the best trade-off between security guarantees and processing capabilities.

Chapter 1

Introduction

1.1. Research Context

Businesses have cooperated via computer networks since the early 1980s. These forms of cooperation were very static and took place in the form of electronic data interchange (EDI) [9]. Since the opening of the Internet for commercial use, more dynamic forms of cooperation are facilitated. However, the security needs of Internet-based systems are very different from those of traditional networking. For example, the Internet offers no centralized infrastructure to provide responsibility for network security. The security needs are particularly acute when high speed internets are used to combine widespread computational resources. The best example of such distributed collaborative environment is the computational grids [10]. A computational grid is a distributed computing infrastructure for advanced science and engineering applications. The initial conception and implementation of these distributed systems were with the aim of providing global sharing of computing resources. Even though the Internet was originally a network built for 'national defense', the security of confidential information was considered secondary, because only trusted users had access to it. However, to exploit the full potential of the Internet and the associated computing resources, they must be made open. This is the point where the in-depth security becomes indispensable.

For example, a survey report, conducted by the Computer Science Department of Virginia Tech in the fall of 2002 among members of the grid community, states that more than half of the respondents believe that existing grid security solutions do not provide adequate services for collaborative grid communities. The reasons given ranged from the *lack of an underlying threat model* to the complexity and expense of inter-site trust relationships that are currently required [11]. Sun Microsystems says adoption of global grids, where companies share hardware and software resources to accomplish a computational goal, has been slowed because of security concerns and a lack of standards [12].

1.2. Security Challenges in a Large Scale Heterogeneous Distributed Computing Environment

The security challenges faced in a large scale heterogeneous distributed computing environments (such as Grid environment) can be grouped into three categories: integration with existing systems and technologies, interoperability with different *hosting environments* (e.g. J2EE servers, .NET servers, Linux systems), and trust relationships among interacting hosting environments [31].

1.2.1. The Integration Challenge

For both technical and pragmatic reasons, it is unreasonable to expect that a single security technology can be defined that will both address all Grid security challenges and be adopted in every hosting environment. Existing security infrastructures cannot be replaced overnight. For example, each domain in a Grid environment is likely to have one or more

registries in which user accounts are maintained (e.g., LDAP directories); such registries are unlikely to be shared with other organizations or domains. Similarly, authentication mechanisms deployed in an existing environment that is reputed secure and reliable will continue to be used. Each domain typically has its own authorization infrastructure that is deployed, managed and supported. It will not typically be acceptable to replace any of these technologies in favor of a single model or mechanism.

Thus, to be successful, Grid security architecture needs to step up to the challenge of integrating with existing security architectures and models across platforms and hosting environments. This means that the architecture must be *implementation agnostic*, so that it can be instantiated in terms of any existing security mechanisms (e.g., Kerberos, PKI); *extensible*, so that it can incorporate new security services as they become available; and *integratable* with existing security services.

1.2.2. The Interoperability Challenge

Services that traverse multiple domains and hosting environments need to be able to interact with each other, thus introducing the need for interoperability at multiple levels:

- At the *protocol level*, we require mechanisms that allow domains to exchange messages. This can be achieved via SOAP/HTTP, for example.
- At the *policy level*, secure interoperability requires that each party be able to specify any policy it may wish in order to engage in a secure conversation – and that policies expressed by different parties can be made mutually comprehensible. Only then can the parties attempt to establish a secure communication channel and security context upon mutual authentication, trust relationship, and adherence to each other's policy.
- At the *identity level*, we require mechanisms for identifying a user from one domain in another domain. This requirement goes beyond the need to define trust relationships and achieve federation between security mechanisms (e.g. from Kerberos tickets to X.509 certificates). Irrespective of the authentication and authorization model, which can be group-based, role-based or other attribute-based, many models rely on the notion of an identity for reasons including authorization and accountability. It would be nice if a given identity could be (pre)defined across all participating domains, but that is not realistic in practice. For any cross-domain invocation to succeed in a secure environment, mapping of identities and credentials must be made possible. This can be enforced at either end of a session through proxy servers or through trusted intermediaries acting as trust proxies.

1.2.3. The Trust Relationship Challenge

Grid service requests can span multiple security domains. Trust relationships among these domains play an important role in the outcome of such end-to-end traversals. A service needs to make its access requirements available to interested entities, so that they can request secure access to it. Trust between end points can be *presumed*, based on topological assumptions (e.g., VPN), or *explicit*, specified as policies and enforced through exchange of some trust-forming credentials. In a Grid environment, presumed trust is rarely feasible due to the dynamic nature of VO relationships. Trust establishment may be a one-time activity per session or it may be evaluated dynamically on every request. The dynamic nature of the Grid in some cases can make it impossible to establish trust relationships among sites prior to application execution [28]. Given that the participating domains may have different security technologies in their infrastructure (e.g., Kerberos, PKI) it then becomes necessary to realize the required trust relationships through some form of federation among the security mechanisms.

The trust relationship problem is made more difficult in a Grid environment by the need to support the dynamic, user-controlled deployment and management of *transient services* [29]. End users create such transient services to perform request-specific tasks, which may involve the execution of user code. For example, in a distributed data mining scenario, transient services may be created at various locations both to extract information from

remote databases and to synthesize summary information. Challenges associated with user-created transient services include the following:

- *Identity and authorization*. It must be possible to control the authorization status (e.g., identity) under which transient services execute.
- *Policy enforcement*. Users may want to establish policies for services that they "own," to control, for example, who can access them and what actions they can perform. However, these policies must necessarily be bounded by policies enforced by the service provider that hosts the user service.
- *Assurance level discovery*. A user may want to take into account the assurance level of a hosting environment when deciding where to deploy services. Thus, this information must be discoverable. Issues of concern may include virus protection, firewall usage for Internet access, and internal VPN usage. One approach to providing this information is to use an accreditation mechanism in which a third-party accreditation agency attests to the level of security provided [30].
- *Policy composition*. Security policy on instantiated services can be generated dynamically from multiple sources: not just the resource owners, but from the entity whose request created the service and the VO in which the entity's membership entitles them to do so.
- *Delegation*. Transient services may need to be able to perform actions on a user's behalf without their direct intervention. For example, a computational job running overnight might need to access data stored in a different resource. Since there may be no direct trust relationship between the VO in which the service is running and the VO in which it wishes to make a request, the service needs to be able to delegate authority to act on the user's behalf.
- A number of secondary issues flow from this requirement. For example: how can a user minimize the credentials they delegate to a transient service to reduce their exposure? And what happens if the credentials delegated to the service expire before it has completed its task?

Controlled access to VO resources and services is clearly a critical aspect of a secure Grid environment.

Given the dynamic nature of Grids and the scale of the environment, serious challenges exist and need to be addressed in the area of security exposure detection, analysis, and recovery.

In summary, security challenges in a Grid environment can be addressed by categorizing the solution areas:

(a) integration solutions where existing services needs to be used, and interfaces should be abstracted to provide an extensible architecture;
(b) interoperability solutions so that services hosted in different virtual organizations that have different security mechanisms and policies will be able to invoke each other; and
(c) solutions to define, manage and enforce trust policies within a dynamic Grid environment.

A solution within a given category will often depend on a solution in another category. For example, any solution for federating credentials to achieve interoperability will be dependent on the trust models defined within the participating domains and the level of integration of the services within a domain. Defining a trust model is the basis for interoperability but trust model is independent of interoperability characteristics. Similarly level of integration implies a level of trust as well has a bearing on interoperability.

In a Grid environment, where identities are organized in VOs that transcend normal organizational boundaries, security threats are not easily divided by such boundaries. Identities may act as members of the same VO at one moment and as members of different VOs the next, depending on the tasks they perform at a given time. Thus, while the security threats to OGSA fall into the usual categories (snooping, man-in-the-middle, intrusion, denial of service, theft of service, viruses and Trojan horses, etc.) the malicious entity could be anyone. An additional risk is introduced, when multiple VOs share a virtualized resource

(such as a server or storage system) where each of participating VOs may not trust each other and therefore, may not be able to validate the usage and integrity of the shared resource. Security solutions that focus on establishing a perimeter to protect a trusted *inside* from an untrusted *outside* (e.g., firewalls, VPNs) are of only limited utility in a Grid environment.

The size of some Grid environments introduces the need to deal with large-scale distributed systems. The number, size, and scalability of security components such as user registries, policy repositories, and authorization servers pose new challenges. This is especially true in the area of inter-domain operations where the number of domains explodes. Many cross-domain functions that may be statically pre-defined in other environments will require dynamic configuration and processing in a Grid environment.

1.3. Security Requirements

The goal and purpose of Grid technologies is to support the sharing and coordinated use of diverse resources in dynamic, distributed VOs: in other words, to enable the creation, from distributed components, of virtual computing systems that are sufficiently integrated to deliver desired qualities of service. Security is one of the characteristics of an OGSA-compliant component. The basic requirements of an OGSA security model are that security mechanisms be *pluggable* and *discoverable* by a service requestor from a service description. This functionality then allows a service provider to choose from multiple distributed security architectures supported by multiple different vendors and to plug its preferred one(s) into the infrastructure supporting its Grid services.

OGSA security must be seamless from edge of network to application and data servers, and allow the federation of security mechanisms not only at intermediaries, but also on the platforms that host the services being accessed. The basic OGSA security model must address the following security disciplines:

- ***Authentication*:** Provide plug points for multiple authentication mechanisms and the means for conveying the specific mechanism used in any given authentication operation. The authentication mechanism may be a custom authentication mechanism or an industry-standard technology. The authentication plug point must be agnostic to any specific authentication technology.
- ***Delegation*:** Provide facilities to allow for delegation of access rights from requestors to services, as well as to allow for delegation policies to be specified. When dealing with delegation of authority from an entity to another, care should be taken so that the authority transferred through delegation is scoped only to the task(s) intended to be performed and within a limited lifetime to minimize the misuse of delegated authority.
- ***Single Logon*:** Relieve an entity having successfully completed the act of authentication once from the need to participate in re-authentications upon subsequent accesses to OGSA-managed resources for some reasonable period of time. This must take into account that a request may span security domains and hence should factor in federation between authentication domains and mapping of identities. This requirement is important from two perspectives:
  a) It places a secondary requirement on an OGSA-compliant implementation to be able to delegate an entity's rights, subject to policy (e.g., lifespan of credentials, restrictions placed by the entity)
  b) If the credential material is delegated to intermediaries, it may be augmented to indicate the identity of the intermediaries, subject to policy.

- *Credential Lifespan and Renewal*: In many scenarios, a job initiated by a user may take longer than the life span of the user's initially delegated credential. In those cases, the user needs the ability to be notified prior to expiration of the credentials, or the ability to refresh those credentials such that the job can be completed.

- *Authorization*: Allow for controlling access to OGSA services based on authorization policies (i.e., who can access a service, under what conditions) attached to each service. Also allow for service requestors to specify invocation policies (i.e. who does the client trust to provide the requested service). Authorization should accommodate various access control models and implementation.
- *Privacy*: Allow both a service requester and a service provider to define and enforce privacy policies, for instance taking into account things like personally identifiable information (PII), purpose of invocation, etc. (Privacy policies may be treated as an aspect of authorization policy addressing privacy semantics such as information usage rather than plain information access.)
- *Confidentiality*: Protect the confidentiality of the underlying communication (transport) mechanism, and the confidentiality of the messages or documents that flow over the transport mechanism in an OGSA compliant infrastructure. The confidentiality requirement includes point–to–point transport as well as store-and-forward mechanisms.
- *Message integrity*: Ensure that unauthorized changes made to messages or documents may be detected by the recipient. The use of message or document level integrity checking is determined by policy, which is tied to the offered quality of the service (QoS).
- *Policy exchange:* Allow service requestors and providers to exchange dynamically security (among other) policy information to establish a negotiated security context between them. Such policy information can contain authentication requirements, supported functionality, constraints, privacy rules etc.
- *Secure logging*: Provide all services, including security services themselves, with facilities for time-stamping and securely logging any kind of operational information or event in the course of time - securely meaning here reliably and accurately, i.e. so that such collection is neither interruptible nor alterable by adverse agents. Secure logging is the foundation for addressing requirements for notarization, non-repudiation, and auditing.
- *Assurance*: Provide means to qualify the security assurance level that can be expected of a hosting environment. This can be used to express the protection characteristics of the environment such as virus protection, firewall usage for Internet access, internal VPN usage, etc. Such information can be taken into account when making a decision about which environment to deploy a service in.
- *Manageability*: Explicitly recognize the need for manageability of security functionality within the OGSA security model. For example, identity management, policy management, key management, and so forth. The need for security management also includes higher-level requirements such as anti-virus protection, intrusion detection and protection, which are requirements in their own rights but are typically provided as part of security management.
- *Firewall traversal*: A major barrier to dynamic, cross-domain Grid computing today is the existence of firewalls. As noted above, firewalls provide limited value within a dynamic Grid environment. However, it is also the case that firewalls are unlikely to disappear anytime soon. Thus, the OGSA security model must take them into account and provide mechanisms for cleanly traversing them—without compromising local control of firewall policy.
- *Securing the OGSA infrastructure*: The core Grid service specification (OGSI) presumes a set of basic infrastructure services, such as handleMap, registry, and factory services. The OGSA security model must address the security of these components. In addition, securing lower level components (e.g., DNSSEC) that OGSI relies on would enhance the security of the OGSI environment.

As Grid computing continues to evolve to support e-business applications in commercial settings, the requirements and functions discussed in this roadmap will form the foundation for standards-based interoperability not only between real organizations within a VO (intra VO) but also across organizations belonging in different VOs (inter VO). On this foundation applications and infrastructure can be built to establish trust relationships that are required

for commercial distributed computing, enterprise application integration and business-to-business (B2B) partner collaboration over the Internet.

1.4. Problematic

A large scale distributed computing system has many properties that can not be ignored when the issue of security is explored. The population of users and the resources they use are assumed to be large and dynamic. Every node in the system could possibly have a different administrator that decides when and how much computing power to make available to the system as a whole. The computation itself is dynamic in the fact that it can allocate resources and spawn other processes during execution, with each resource located at different nodes in the distributed architecture. Processes must be able to communicate using a variety of communication methods. Resources can require different authentication or verification schemes in order to allow use of the system, and can be located in different countries.

Since a metacomputing system is really the compilation of many administrative domains and their computing power a security system for the metacomputing level can not be expected, or for that matter allowed, to override local security policies but instead to work with them or on top of them.

At present, the provision of security services to such large scale heterogeneous distributed computing system remains very much an open research problem. In this research work, we have explored a comprehensive solution to tackle this problem.

1.5. Motivations and Prospects

The security and privacy issues are coming to the fore with the growing size and profile of the grid community. The forthcoming generations of the computational grid will make available a huge number of computing resources to a large and wide variety of users. The diversity of applications and mass of data being exchanged across the grid resources will attract the attention of hackers to a much higher extent. A comprehensive security system, capable of responding to any attack on its resources, is indispensable to guarantee the anticipated adoption of grid by both the grid users and the resource providers. In this article, we argue that the first brick of an effective plan of countermeasures against these threats is an analysis of the potential risks associated with grid computing.

A pragmatic analysis of the vulnerability of existing grid systems [4] and the potential threats posed to their resources once their spectrum of users is broadened. Various existing grid projects and their security mechanisms are reviewed. The experience of using common grid software and an examination of grid literature served as the basis for this analysis. Legal loopholes in the implementation of grid applications across the geopolitical frontiers, and the ethical issues that could obstruct the wide acceptance and trustworthiness of grids are also discussed. The weaknesses revealed are classified with respect to their sources and possible remedies are discussed. The results show that the main reason for the vulnerability is the fact that grid technology has been little used except by a certain kind of public (mainly academics and government researchers). This public benefit greatly from being able to share resources on the grid, and have no intention of harming the resource owners or fellow users. Thus there was no need to address security in depth. This is all about to change. The number of people who know about the grid is growing fast, as are the worthwhile targets for the potential attackers. The security nightmare can not be avoided unless the problem is addressed urgently.

In the evolution of computational grids, security threats were overlooked in the desire to implement a high performance distributed computational system. So far, the grid technology has been little used except by a certain kind of public (mainly academics and government researchers). The growing size and profile of the grid require comprehensive security solutions as they are critical to the success of the endeavor. A comprehensive security system, capable of responding to any attack on grid resources, is indispensable to guarantee

its anticipated adoption by both the users and the resource providers. The prospects of designing comprehensive security architecture for these systems are quite fascinating.

1.6. Approach and Methodology

The research method is based on theoretical insight and best practices. It is supplemented with practical explorative experience. The research makes use of the following:

- a study of information security literature;
- a study of system design literature;
- experience gained while participating in the various EU projects in which information security design was the main subject;

The research method consists of six parts, which all contribute to the answering of the research problem. These parts are:

1. Identification of the threats posed to the large scale open heterogeneous distributed computing systems;
2. Identification of shortcomings of current security processes when addressing information security issues;
3. Proposition of a security model consisting of supporting means for designers to address information security issues in the design process of these systems;
4. Verification of the proposed model;
5. Development of a tool for the simulations of the security services and the formulation of evaluation criteria.
6. Description of the applications that can benefit from the proposed security model.

1.7. Organization of Thesis

In this section, we give an overview of the rest of the chapters of this thesis. The remaining chapters are organized as:

In *Chapter 2* we provide a detailed pragmatic analysis of the threats posed to the various existing technologies.

In *Chapter 3* we describe a number of key concepts needed to be elaborated before designing a security model.

In *Chapter 4* we provide state of the art security mechanisms in existing systems with special emphasis on their security features and shortcomings.

In *Chapter 5* we present our proposed security model to handle the security needs of the underlying systems.

In *Chapter 6* we present the salient features of our proposed architecture.

In *Chapter 7* we discuss the assessment of the various security functionalities. It includes Common Criteria, a case study and the description of our proposed security simulation tool.

In *Chapter 8* we present a set of applications that can benefit from our current research work.

Finally, in *Chapter 9* we give conclusions and suggestions for future research.

Chapter 2

Threats Analysis

Computer crimes particularly hacking and denial of service have become so ubiquitous that they are almost accepted as few of those unavoidable facts of modern life. The statistics of such incidents underline the vulnerability of even apparently secure computer systems to various attacks [32-34]. We believe that the actual statistics of incidence are higher than the reported ones, because the last thing a company wants everybody else to know is that they have been attacked. It is certainly embarrassing for them to admit that their IT system was not secure enough. These challenging nature of the threats exacerbate the state of the IT security:

- Intruders are prepared and organized.
- Attacks are getting easier, low risk and hard to trace.
- Intruder tools are increasingly sophisticated, easy to use (especially by novice intruders) and are designed to support large-scale attacks.
- Source code is no more required to find vulnerabilities.
- The complexity of the Internet, protocols, and applications are all increasing along with our reliance on them.
- Critical infrastructures increasingly rely upon the connected IT services and products for operations.

Besides a number of exploitable vulnerabilities present in technology, lack of awareness regarding information security is a great source of exploitation. Most intrusions result from exploitation of known vulnerabilities, configuration errors, or virus attacks where countermeasures were already available. For example, San Diego Supercomputer Center (SDSC) conducted an experiment of Red hat 5.2 installation without any security patches on a computer in December 1999. It is amazing that within 8 hours of installation, the system was probed! The complete chronology of the events of this experiment is available at [35]. So if a system was properly administered with timely installation of proper patches then would the great source of exploitation be throttled? The answer could be positive if the administrations of computer and network infrastructures are not as complex as they are.

The complexity administration of computer and network infrastructures of administration of computer and network infrastructures makes it difficult to properly manage the security of computer and network resources. For example 5500 vulnerabilities reported in 2002 [36]; it means an administrator has to read the description of these vulnerabilities – if he takes 20 minutes for each description, then he requires 229 days just to read the description; if he is affected by only 10% of these vulnerabilities and requires 1 hour for the installation of each patch, then he requires 69 days to install patches on one machine. So just to read security news and patch a single system requires 298 days! Even just 5 minutes to read new vulnerability bulletin and only a 1% *hit rate* costs almost 65 days (or about 25% of a perfectly efficient administrator). These problems can be mitigated if a rigorous vulnerability assessment is made before the actual installation of these systems. With the ever-increasing computing power, conception of robust security simulators is not just a sweet dream. A portion of security investments is necessary for the evolution of security simulation tools. Network Simulator (NS) is the best-known simulator for various simulations; however, it has a number of limitations including the scalability [37]. The scalability is an important factor in today's heterogeneous distributed systems. If a system is floated in the market without its

proper testing and simulations then there is no guarantee that its functionality will meet the anticipated performance targets. The consequences are obvious – there will remain a number of vulnerabilities in the system and there will always be a need to release patches and the job of the system administers will remain complex! Apart from this, the losses incurred due to the exploitation of these vulnerabilities might not have any remedy; for example, there is no way to undo the theft of priceless human data from a medical database.

Lack of accountability feature in the security models is a matter of great concern. Malicious insiders can exploit the information, which lies within their reach. In a survey of 1225 information security managers, the greatest threat to their networks was virus infections followed by abuse of access privileges by employees [38]. Efficient accountability measures are particularly needed to tackle the recent technology advancement, which facilitate the risks of intellectual property theft. An example is the growing number of low cost high capacity portable storage media devices. Such devices bypass traditional safeguards and open the enterprise to a range of threats because they remain invisible to normal perimeter-based security. Flash drives and MP3 players, which many employees bring into the organization, can be plugged into any USB port and are automatically recognized by recent Windows operating systems. They can transport very large quantities of data and that is why they are so dangerous.

The risks come from the fact that there is no way to control what is already on the devices when they are connected to the corporate network, nor is it easy to control what is transferred onto them. In the first case such things as Trojans, hacking tools, viruses, worms, or any number of other malware infections can easily access the system and bypass corporate security systems and procedures. A single infected file on a flash memory stick could cause havoc. What is even more concerning is that the huge storage capacity of modern MP3 players and other removable storage devices means that they can be used to steal considerable quantities of sensitive or propriety information. What's needed is a secure solution that manages how, where and when these media devices can be connected to a network.

2.1. Client-Server Architecture

In client/server systems where the data may be distributed across multiple servers and sites, each with its own administrators, centralized security services are impractical as they do not scale well and more opportunities are available for intruders to access the system. The client PCs often run operating systems with little or no thought to security and the network connecting clients to servers is vulnerable. The distribution of services in client/server increases the susceptibility of these systems to damage from viruses, fraud, physical damage and misuse than in any centralized computer system. With businesses moving towards multi-vendor systems, often chosen on the basis of cost alone, the security issues multiply. Security has to encompass the host system, PCs, LANs, workstations, global WANs and the users.

Generally, client-server security is deemed to be achieved by encrypting the data flow between the server and its clients. Encryption is necessary for client-server data exchange; however, it is not sufficient for complete security assurance as there are a number of factors, such as availability and covert channels [39], which are not dealt by cryptographic solutions. There is no one solution that addresses all security issues raised when implementing a client-server system. The view to the users must be that of a single homogeneous system when the reality is that it is a system made up of multiple levels and parts, each with its own security issues. We have classified the vulnerabilities associated to the client/server architecture into the following two categories:

2.1.1. Mutual Confidence/Trust already exists

In this case, users already know each other and/or they can identify each other. Also, they can determine the confidence level themselves. Although, there exists a mutual confidence/ trust among the users; however, the network connecting clients and servers is a less than

secure vehicle that intruders can use to break into computer systems and their various resources. Using publicly available utilities and hardware an attacker can eavesdrop on a network, or "sniff" the network to read packets of information. These packets can contain useful information, e.g. passwords, company details, etc, or reveal weaknesses in the system that can be used to break into the system.

Encryption of data can solve the problem of attackers sniffing the network for valuable data. Encryption involves converting the readable data into unreadable data. Only those knowing the decryption key can read the data. A problem here is that some network operating systems don't start encryption until the user has been authenticated (i.e. the password is sent unencrypted). Most systems employ re-usable passwords for authenticating users, which allows an attacker to monitor the network, extract the login information and access the system posing as that user. Even if the password is encrypted the intruder can just inject that packet into the network and gain access. The problem is compounded when, to maintain that single system illusion, only one login is required to access all servers on the network. Customers want a "single system image" of all networked computing resources, in which all systems management and administration can be handled within a single pool of system resources.

2.1.2. Mutual Confidence/Trust does not already exist

In this case, the users do not have previous knowledge/confidence of each other or the application they are using does not require such existing familiarity (e.g. electronic commerce). In such a situation, the security of client side, server side and their communication medium (such as network) are all indispensable. Secure socket layer (SSL) that is used for managing the security of a message transmission on the Internet does not provide mutual authentication because unlike the enterprises (servers), the customers (clients) do not possess digital certificates. The enterprises use their certificates to prove their authenticity; whereas the customers provide specific information (credit card number, social security number, etc.) to prove their identity; however, in the absence of some mutual trust the provision of such information is prone to attacks. This issue is a hot topic of Business to Consumer (B2C) systems [40].

The network security vulnerabilities have discussed above. The client and server security vulnerabilities are described below:

The Client

The client machines pose a threat to security as they can connect to the servers, and access their data, that are elsewhere in an organization. One large problem is that they are easily accessible and easy to use. They are usually located in open plan offices that present a pleasant environment for users (and intruders) making it impossible to lock them away when unattended.

One of the greatest risks with the client workstations is that the operating system is easily and directly accessible to the end user, which exposes the whole system to a number of risks. The workstation operating system assumes that the person who turns it on is the owner of all files on the computer, including the configuration files. Even if the client/server application has good security, that security might not be able to counteract attacks at the operating system level, which could corrupt data passed to other tiers of the client/server system. The tighter security now being offered with Windows NT addresses some of these issues.

The Server

The first line of defense for the server(s) is to have the server center in a secure location where access, by authorized personnel only, can be supervised and administration can be performed simply. Some database systems, e.g. Oracle, can validate database users without database passwords by using information in the host’s operating system authentication mechanism. This simplifies the database security administration as it is centralized at a

system level. However, the problem with this is that often the database user does not have to start a host session on the database server before using the database across the network, effectively bypassing the operating system security mechanisms.

Many client/server database systems do not have adequate password management facilities similar to those found in operating systems. This means that there is no easy way for the database administrator to ensure that users have chosen good passwords and will change them frequently. Most database systems allow users to change their passwords using simple SQL utilities. Unfortunately these SQL utilities do not force the user to verify the current password prior to changing it. Therefore, it is easy to change another user's password. Third party database password management utilities are starting to appear which address the above deficiencies.

2.2. Peer-to-Peer (P2P)

P2P networking has grown faster than the Internet itself, and has reached a much broader audience at this stage of its development. Part of the attraction of P2P networks is their dynamic nature. P2P technology creates flexible *ad hoc* networks that span the globe, connecting end users in a peer-wise architecture that is both resilient and efficient. Search engines built into P2P clients are powerful and intuitive. They put a staggering volume and variety of digital content at a user's fingertips. But there is a downside to placing such a potent technology in the hands of novice users. A P2P client can turn a computer into a server, exposing it to a new range of threats.

Installation and operation is so easy that most do not fully appreciate the risks. And deceptive practices of the vendors of P2P file sharing software who are trying to stay one step ahead of copyright owners and network administrators have made the situation much worse.

2.2.1. Spyware and Adware

The prevalence of embedded spyware and adware in P2P clients is but one example. Spyware monitors user behavior and tracks web browsing habits. The information collected by spyware is typically sold to companies and/or used by adware to conduct targeted web marketing. Based on an individual's browsing patterns, adware opens web pages promoting a particular product or service.

P2P developers bundle spyware and adware in their clients to generate revenue. One P2P Company maintains that its embedded spyware is "integral" to the operation of their product. Of course, there is no inherent functional dependency between advertising and file sharing. In fact, lightweight implementations of P2P software have been developed that leave the spyware out. "Integral" means that the P2P software has been deliberately engineered so it will not function without the spyware active.

Spyware and adware are, by construction, difficult to detect and may be impossible to disable or remove from a client. Common tactics include hiding in system folders and running in the background from system startup. Amazingly, some spyware components remain on a system long after the original application is removed, and will even embed themselves in a host despite an aborted installation of the carrier application.

Spyware not only poses a threat to user privacy, it can also create additional vulnerabilities on a user's system. Spyware products embedded in the most popular P2P clients download executable code without user knowledge. Even if the code is not malicious, it may contain flaws that render a system open to attack. The clandestine nature of the software makes detection and remediation extremely difficult.

2.2.2. Circumventing Security

P2P software is commonly designed to circumvent network security services. Enterprises and institutions wishing to stem the tide of media piracy on their networks often find that P2P file sharing traffic is disguised as or hidden amongst normal network activity. Techniques

such as tunneling, port hopping and push requests make it difficult to detect and filter P2P traffic. That is their intent; to foment user participation in spite of an enterprise's security policy. One consequence (intended or not) is that these techniques dramatically weaken an organization's security posture.

Tunneling embeds P2P messages within another protocol so that they blend in with other traffic, making them more difficult for firewalls and filters to detect. A common scheme is HTTP tunneling, in which P2P communications are disguised as web browsing traffic. This variation is popular because web traffic is so common and typically travels freely across enterprise networks. To this end, tunneling not only helps violate a network security policy by enabling forbidden applications but also expands the network security perimeter in ways unknown and unpredictable to system administrators.

Another commonly used trick is for P2P clients to vary their communication ports – a technique called port hopping. This thwarts blocking and scanning software that identifies network services based on well-known port assignments. Port hopping is built into the latest versions of the most popular P2P clients – and there is no reason for it other than to allow network software clients to avoid detection.

Developers of the Gnutella protocol devised a special solution that permits clients to circumvent firewalls configured to block its file request messages. In this scheme, a 'push-request' message is sent through the Gnutella network to the system behind the firewall, which then knows to initiate a file upload to the requesting host. So instead of a client 'pulling a file to it,' it asks the serving system to 'push the file out.' To the user, the net effect is the same – they get the file – but to the firewall, which usually has looser restrictions on out-bound traffic, it makes all the difference in the world. And once again, an enterprise's network security policy is violated.

2.2.3. Software Vulnerabilities

Another major concern is how software flaws in P2P networking clients can greatly increase the exposure in a network, leaving it vulnerable to intruders and hackers. All software has flaws, and some flaws create exposures that can be exploited to violate the security of a system.

Exploitable weaknesses in P2P software have been identified. Buffer overflow and cross-site scripting vulnerabilities were reported in early iMesh and Gnutella clients, respectively. P2P clients that use the Fasttrack protocol are known to be susceptible to Denial of Service attacks due to its client-to-client messaging architecture. Sometimes the shared files themselves enable an attack. MP3s contain special meta-data that in the past has been used to exploit buffer overflow vulnerabilities in media players. In this particular attack, a P2P network is simply a distribution mechanism for the malicious payload, but it is an incredibly effective one.

There is nothing special about P2P software that makes it inherently more flawed than other software. It is built for the same platforms and developed in the same programming languages as other computer and network applications. However, several factors conspire to make the risks induced by security vulnerabilities in P2P file sharing clients much more serious. The first factor is that P2P clients engender massive *ad hoc* connectivity across organizational and enterprise domains. P2P file sharing networks are well beyond the administrative control of any one company or organization. A system running a P2P client may be behind a firewall, but it is exposed through the client to every user on that P2P network, regardless of their location. Simply put, P2P clients can dramatically amplify exposures to external threats.

A related factor deals with trust. P2P file trading networks are open environments that allow anyone to share files pseudo-anonymously. Trust in this circumstance is hard to come by. Users are connected to and download files from hosts they know very little about. In many cases, the P2P client itself is installed in a bootstrap process that downloads it from a peer on the network. P2P file sharing networks expose systems to untrusted hosts and software, and offer little in the way of protection.

Enterprise security management in the presence of contraband P2P file sharing software is a supreme challenge. The dynamic nature of P2P networks, the stealth tactics employed by the software and the tendency of individuals to hide its use makes a complete inventory of P2P clients on a large network virtually impossible. This again magnifies any security vulnerabilities because inventories are essential for security remediation processes. It is very difficult to address problems on a network if you cannot find the software that is causing them.

2.2.4. Worms and Viruses

No discussion of security threats to P2P networks is complete without covering the potential for viruses and worms. Viruses and worms are self-replicating code that may or may not contain a malicious payload. The difference between the two is that a virus typically requires some form of human participation to propagate while a worm can spread across a network without human intervention. Both are viable modes of attack in P2P networks.

A P2P virus needs a carrier file to contain its payload. The obvious choices are audio, video and executable files traded over the network. Buffer overflow vulnerabilities have already been exploited in media players by maliciously crafted MP3 files. A virus can leverage such a weakness to execute code that replicates itself in the shared folder directory of a user. The act of downloading an infected file spreads the virus to a new host.

The recent integration of executable content in media formats creates a richer entertainment experience, but also offers a limitless space for viral code. Likewise, e-mail attachments became the preferred mode of virus transmission after the introduction of active content in word processing documents and web pages. Scripting means a user no longer has to break an application with a buffer overflow attack. Instead, he can exploit weak security policies and input validation processes to achieve the same effect.

Likewise, a self-propagating P2P worm could infect almost every host on the P2P network, crossing enterprise network boundaries with blazing speed. More importantly, the previously discussed obstacles to efficient remediation indicate that a P2P worm would have tremendous staying power, re-infecting unpatched hosts and infecting new ones as they came online.

There is a role for technology to play in addressing these problems. Tools and systems can be developed to better monitor and secure hosts running P2P clients. Of course technology is only one piece of the solution. Users must be made aware of the risks of participating in open P2P file sharing networks. Developers must be held accountable and live up to higher standards of integrity and transparency for the P2P software they build.

In a very real sense, peer-to-peer file trading software exposes individuals and enterprises to risks above and beyond those of other software. The technology itself is beautiful in its design, but developer and user practices conspire to create a dangerous operational environment. On its current evolutionary track, threats to security and privacy posed by P2P file sharing technology will get worse, not better.

2.3. Distributed Applications

2.3.1. Multiple Application Multiple Users (Videoconferencing)

Videoconferencing sessions are primarily transmitted using ISDN and IP signaling standards. Each signaling standard has its own set of security weaknesses. The brand and model of videoconferencing equipment used also contributes in a major way to the vulnerability of videoconferencing in general.

Videoconferencing over IP utilizes the TCP/IP protocol, often the Internet as a transmission medium, and can be easily monitored or recorded using any off the shelf packet-sniffing tool installed on a computer in the LAN or at a switch. The inherent vulnerability of IP video transmissions traveling over the Internet or a private IP network is linked to the features of the Internet Protocol.

To understand some of the vulnerabilities of an ISDN video call, it is helpful to look at the path of an ISDN call. Although the ISDN signal is originally digital, it is soon converted at the phone company's switch. There, it may be converted to analogue for routing over conventional phone lines or satellite, where it may be reconverted and sent over a fiber optic network. Once near its destination, it is converted back to digital at a local switch location, where it can be routinely monitored by the owner of the switch. Since ISDN is switched to a variety of formats, it is not inherently safer than any other broadband communications. The path of the information is uncertain, which also is not conducive to maximum privacy. ISDN video calls can also be monitored and recorded with a simple and inexpensive device called an ISDN line tester.

Both IP and ISDN video systems are vulnerable to attacks that can give attackers remote control of the videoconferencing device. Through dial-up management hackers can gain access to the listen command and monitor calls. This type of remote control can also allow unauthorized individuals to gather information about the device, retrieve files, crash the device, or stream the video session to another domain on the Internet.

2.3.2. Single Applications, Multiple Users (Multicasting)

In addition to the general threats to security (eavesdropping, impersonation, data manipulation, denial of submission, denial of receipt, denial of service, repudiation, replay attacks, theft of service, theft of content, etc.), multicast communication is subject to some additional security threats and vulnerabilities:

- **Uncontrolled Multicast Group Membership:** IP multicast protocols provide no means to specify, control, or limit the membership of a multicast session group.
- **Leakage of Security State:** In multicast environments, the security state of a multicast session may be shared among multiple participants, thereby increasing the risk of security state leakage.
- **Security State Revocation:** There is no simple mechanism to revoke the validity of a security state and to notify all multicast participants of the change. This may be necessary in order to restrict access of a multicast group member that has left the group.
- **Security Management:** Multicast protocols do not provide mechanism s for managing the security attributes of multicast group members. In particular, the problems of key distribution and of re-keying a group are of major concern. Another significant problem is the lack of synchronization among the group members.
- **Multi-Point Attacks:** In multicast, it is easier for an attacker to pose as one of many users, or to attack at several points in the network at the same time, thereby increasing the vulnerability of the system.

2.3.3. Grid Architecture

The security and privacy issues are coming to the fore with the growing size and profile of the grid community. The forthcoming generations of the computational grid will make available a huge number of computing resources to a large and wide variety of users. The diversity of applications and mass of data being exchanged across the grid resources will attract the attention of hackers to a much higher extent. A detailed Grid security risks analysis is carried-out in [4]. This analysis is based on the static and fix Grids; however, some features of mobile and pervasive Grids (such as Security Gaps) are also considered. The threats and vulnerabilities pertaining to the mobility are elaborated in section E. A concise account of Grid analysis is presented below:

Cryptography

Public key encryption does not depend on keeping the algorithms secret. In fact they are public knowledge. One could therefore easily write a program to decrypt a message via the *brute force cracking* method of trying all the possible keys. Although a vast amount of

computer power is needed to run such a program. However, computers are getting faster all the time, roughly doubling in speed every 18 months (Moore's Law). Besides this, public-key encryption has two major drawbacks: (a) public-key calculations take much longer than symmetric key calculations. Public key calculations involve the exponentiation of very large numbers, and these operations take a hundred to a thousand times longer to compute than symmetric key calculations; and (b) public key encryption lends itself to a cryptographic *man-in-the-middle* attack. (where two persons Alice and Bob want to communicate in secret, but an attacker Charles pops up between them and convinces Alice that he is Bob and Bob that he is Alice.)

Proxies and Delegation

There is a private key (for the proxy) that is on a remote system outside the user's direct control, but can be used to sign messages that the grid infrastructure will trust as if coming from him. If a remote system were compromised (or spoofed), the user proxy's private key would no longer be private. Even if he becomes aware of this he would not be able to revoke the proxy because he doesn't have a revocation process.

Authorization

Most grid systems available today assert that authorization to use a resource should be granted by the resource owner, on the basis that the owner should retain full control of what happens to their resources. Many grid systems implement this by simply mapping a remote grid user identity onto a local account on the resource. The local system administrator can then define which resources the local account is able to access. It is obviously right and proper that the owner or operator of a resource should be able to control it, but there are several problems with the account-mapping approach as a way to achieve this. The obvious problem is that the number of grid users is likely to scale with the number of organizations participating in the grid, while the number of system administrators at each site does not. At some point the number of accounts that need to be created and managed to control access by remote users will become too great, and then the system administrator will have no option but to give all grid users the same *standard* access rights.

Security Gaps

The security gaps are introduced in any secure path going through one or more middleboxes that need to perform some processing on passing data packets. These middleboxes include Network Address Translation (NAT) gateways, packet or content filters, proxy firewalls, and Wireless Application Protocol (WAP) gateways.

GSI [28] provides secure authentication and communication for grids, it does not attempt to discover middleboxes and negotiate security with them. As a result, security gaps could surface, particularly in cases where some grid resources and nodes exist in a local network behind a firewall. Further, the adaptability of GSI is limited making it hard to port it to lightweight devices (e.g. PDAs) with limited capabilities.

Grid Applications

Even if the grid infrastructure is made safe, it is still possible that the underlying applications made available to grid users may be insecure. Giving the user access to a command shell (even within predefined scripts or applications) is clearly extremely dangerous.

Legal issues

Wide spread grid applications have raised unique legal concerns. The existing cyber laws do not comprehensively address this new range of issues. The spread of various grid resources across a number of political frontiers will encourage the potential attackers to launch their attacks from safe havens – countries where the corresponding laws are either

nonexistent or are comparatively less strict. Examples include inconsistencies in the copyright laws, intellectual property laws, personal information confidentiality laws, etc.

Ethical issues

Grid computing is not simply a giant leap in technology; it has social and ethical impacts as well. Its wide scale acceptance by the general public heavily lies in its ability to convince them that their information will not be used (directly or indirectly) for any unsolicited purpose. The resource owners would also like to be assured that their resources will not be a part of some malicious activity.

2.4. Mobility

Security concerns come to the forefront in highly dynamic mobile environments. Threats and vulnerabilities associated with the mobility can be broadly classified into physical, centralized and decentralized mobility sub-domains. However, in the real world situations, a mobile system may consist of several sub-domains. For example, a fleet of robots has characteristics of

- Physical mobility – physical infrastructures of the individual robots,
- Centralized mobile architecture – connections of various robots at one site to their counterparts at some other location through GSM/GPRS/UMTS, and
- Decentralized mobile architecture – connections of various robots to their host and with each other through ad-hoc network.

Threats and vulnerabilities associated with these mobility sub-domains are described below:

2.4.1. Physical Mobility

Mobility and ubiquitous computing are key technologies of the decade. But protection of mobile systems is not comparable with protection of stationary architectures. The mobility aspect implies a lot of new threats and attacks that have to be covered in a security concepts and protocols.

The technical opportunities strengthen the user mobility and encourage the deployment of the mobile technologies for the development of various (mobile) applications providing information, orientation (routing) and other helpful services. Therefore, mobile users will take advantage of multi-interface (e.g. WLAN, Bluetooth, UMTS) PDAs and Smart Phones, participating in countless Ad-Hoc networks and using push services for orientation and information. Thus the user (and his device) is frequently switching networks, confronting him with interchanging security threats in a fast order. Coming home from a longer business trip and finally re-entering the home/corporate (W)LAN environment doesn't mean the user is secure – in contradiction to the feeling of entering a save harbor, one must regard the homecoming device as a considerable security threat itself being a potential carrier of malicious code or Trojans. Such threats dwarf the user acceptance, which is necessary for the success of this technology. The fundamental factors to gain and improve the user acceptance are e.g. transparent services and security aspects. The very nature of most wireless communications makes security a significant factor that must be understood and addressed for wireless communication to achieve its vast potential.

The repeated change of participation of mobile users in un-trusted networks is a challenge for both the direct security and integrity of the mobile device (attacks launched directly against it, like DoS depletion, misuse of the air interfaces, stealing of computational resources etc.) and the security of the home LAN (insertion of Trojans, malicious code and viruses behind the firewall and anti-virus gateway). With the increase of information that is send to the device, the probability of malicious content being inserted also grows.

Another example for serious threats that already emerge in today's GSM/GPRS networks is the usage of JAVA enabled phones and the problems arising from the data transfer when

downloading JAVA content over the air. The JAVA environment on mobile or embedded devices, utilizing MIDP, kJAVA or proprietary subsets of the above, makes powerful applications, applets and games available to user. Some loopholes have been identified in the security of java-enabled phones, which can be exploited by virus-like alterations of code. These problems, now being a mere annoyance due to their little impact on usage, can and will evolve, when devices become more powerful, networks offer higher bandwidth and applications get higher privileges. But devices will still not be capable to provide scan-engines like the ones offered for desktop machines! Thus, the need for server-based or network-based solutions is evident and research has to be conducted to improve the security of both networks and end-user devices.

Besides the security regarding the device itself and it's associated home LAN, issues of privacy vs. authentication have to be taken into account. On the one hand, providers for premium services need to be able to identify their customers for billing purposes or provision of tailor-made applications and services. On the other hand, user preference profiling and location tracing/tracking bring up privacy issues of considerable importance. Some ideas regarding the use of multiple pseudonyms have been introduced theoretically circumventing the above-mentioned problems but introducing new ones with regard to multi-identity management and billing.

When considering technical solutions for the above introduced set of issues, the user himself has to be integrated in each concept right from the start. Security for applications, services and devices is mandatory, but the user and his behavior will once again be the weakest point. Therefore, easy-to-use mechanism for extra authorization, change of pseudonyms etc. have to be implemented, allowing the user to adapt to the more complex environment of multi-service (multi-)wireless networks.

A major challenge for user e.g. SMEs (small and medium enterprises) and vendors of the information and communication technology is to implement security in a way that meets business needs cost-effectively, both in the short term and as the enterprise needs to expand. In order to meet this challenge, the improvement of the existing methods of identifying and analyzing threats and security risks, and of specifying, designing and implementing security policies.

2.4.2. Software Mobility

Software mobility is carried out by the mobile agents. Mobile agents simply offer a greater opportunity for abuse and misuse, broadening the scale of threats significantly. An agent is comprised of the code and state information needed to carry out some computation. Mobility allows an agent to move, or hop, among agent platforms. The agent platform provides the computational environment in which an agent operates. The platform from which an agent originates is referred to as the home platform, and normally is the most trusted environment for an agent.

Four threat categories for the mobile agents can be identified: threats stemming from an agent attacking an agent platform, an agent platform attacking an agent, an agent attacking another agent on the agent platform, and other entities attacking the agent system. The last category covers the cases of an agent attacking an agent on another agent platform, and of an agent platform attacking another platform, since these attacks are primarily focused on the communications capability of the platform to exploit potential vulnerabilities. The last category also includes more conventional attacks against the underlying operating system of the agent platform.

Agent-to-Platform

The agent-to-platform category represents the set of threats in which agents exploit security weaknesses of an agent platform or launch attacks against an agent platform. This set of threats includes masquerading, denial of service and unauthorized access.

Masquerading

When an unauthorized agent claims the identity of another agent it is said to be masquerading. The masquerading agent may pose as an authorized agent in an effort to gain access to services and resources to which it is not entitled. The masquerading agent may also pose as another unauthorized agent in an effort to shift the blame for any actions for which it does not want to be held accountable. A masquerading agent may damage the trust the legitimate agent has established in an agent community and its associated reputation.

Denial of Service

Mobile agents can launch denial of service attacks by consuming an excessive amount of the agent platform's computing resources. The mobile computing paradigm requires an agent platform to accept and execute an agent whose code may have been developed outside its organization and has not been subject to any a priori review. A rogue agent may carry malicious code that is designed to disrupt the services offered by the agent platform, degrade the performance of the platform, or extract information for which it has no authorization to access. Depending on the level of access, the agent may be able to completely shut down or terminate the agent platform.

Unauthorized Access

Access control mechanisms are used to prevent unauthorized users or processes from accessing services and resources for which they have not been granted permission and privileges as specified by a security policy. Each agent visiting a platform must be subject to the platform's security policy. Applying the proper access control mechanisms requires the platform or agent to first authenticate a mobile agent's identity before it is instantiated on the platform. An agent that has access to a platform and its services without having the proper authorization can harm other agents and the platform itself. A platform that hosts agents representing various users and organizations must ensure that agents do not have read or write access to data for which they have no authorization, including access to residual data that may be stored in a cache or other temporary storage.

Agent-to-Agent

The agent-to-agent category represents the set of threats in which agents exploit security weaknesses of other agents or launch attacks against other agents. This set of threats includes masquerading, unauthorized access, denial of service and repudiation. Many agent platform components are also agents themselves. These platform agents provide system-level services such as directory services and inter-platform communication services. Some agent platforms allow direct inter-platform agent-to-agent communication, while others require all incoming and outgoing messages to go through a platform communication agent. These architecture decisions intertwine agent-to-agent and agent-to-platform security. This section addresses agent-to-agent security threats and leaves the discussion of platform related threats to sections E.2.1 and E.2.3.

Masquerade

Agent-to-agent communication can take place directly between two agents or may require the participation of the underlying platform and the agent services it provides. In either case, an agent may attempt to disguise its identity in an effort to deceive the agent with which it is communicating. An agent may pose as a well-known vendor of goods and services, for example, and try to convince another unsuspecting agent to provide it with credit card numbers, bank account information, some form of digital cash, or other private information. Masquerading as another agent harms both the agent that is being deceived and the agent whose identity has been assumed, especially in agent societies where reputation is valued and used as a means to establish trust.

Denial of Service

In addition to launching denial of service attacks on an agent platform, agents can also launch denial of service attacks against other agents. For example, repeatedly sending messages to another agent, or spamming agents with messages, may place undue burden on the message handling routines of the recipient. Agents that are being spammed may choose to block messages from unauthorized agents, but even this task requires some processing by the agent or its communication proxy. If an agent is charged by the number of CPU cycles it consumes on a platform, spamming an agent may cause the spammed agent to have to pay a monetary cost in addition to a performance cost. Agent communication languages and conversation policies must ensure that a malicious agent doesn't engage another agent in an infinite conversation loop or engage the agent in elaborate conversations with the sole purpose of tying up the agent's resources. Malicious agents can also intentionally distribute false or useless information to prevent other agents from completing their tasks correctly or in a timely manner.

Repudiation

Repudiation occurs when an agent, participating in a transaction or communication, later claims that the transaction or communication never took place. Whether the cause for repudiation is deliberate or accidental, repudiation can lead to serious disputes that may not be easily resolved unless the proper countermeasures are in place. An agent platform cannot prevent an agent from repudiating a transaction, but platforms can ensure the availability of sufficiently strong evidence to support the resolution of disagreements. This evidence may deter an agent that values its reputation and the level of trust others place in it, from falsely repudiating future transactions. Disagreements may arise not only when an agent falsely repudiates a transaction, but also because imperfect business processes may lead to different views of events. Repudiation often occurs within non-agent systems and real-life business transactions within an organization. Documents are occasionally forged; documents are often lost, created by someone without authorization, or modified without being properly reviewed. Since an agent may repudiate a transaction as the result of a misunderstanding, it is important that the agents and agent platforms involved in the transaction maintain records to help resolve any dispute.

Unauthorized Access

If the agent platform has weak or no control mechanisms in place, an agent can directly interfere with another agent by invoking its public methods (e.g., attempt buffer overflow, reset to initial state, etc.), or by accessing and modifying the agent's data or code. Modification of an agent’s code is a particularly insidious form of attack, since it can radically change the agent's behavior (e.g., turning a trusted agent into malicious one). An agent may also gain information about other agents’ activities by using platform services to eavesdrop on their communications.

Platform-to-Agent

The platform-to-agent category represents the set of threats in which platforms compromise the security of agents. This set of threats includes masquerading, denial of service, eavesdropping, and alteration.

Masquerade

One agent platform can masquerade as another platform in an effort to deceive a mobile agent as to its true destination and corresponding security domain. An agent platform masquerading as a trusted third party may be able to lure unsuspecting agents to the platform and extract sensitive information from these agents. The masquerading platform can harm both the visiting agent and the platform whose identity it has assumed. An agent that masquerades as another agent can harm other agents only through the messages they

exchange and the actions they take as a result of these messages, but a malicious platform that masquerades as an authorized platform can do more harm to the duped agent than a single agent can do on its own. The threat of a malicious platform altering an agent's code, state, or data is discussed in more detail in section E.2.3.4.

Denial of Service

When an agent arrives at an agent platform, it expects the platform to execute the agent's requests faithfully, provide fair allocation of resources, and abide by quality of service agreements. A malicious agent platform, however, may ignore agent service requests, introduce unacceptable delays for critical tasks such as placing market orders in a stock market, simply not execute the agent's code, or even terminate the agent without notification. Agents on other platforms waiting for the results of a non-responsive agent on a malicious platform must be careful to avoid becoming deadlocked. An agent can also become live-locked if a malicious platform, or programming error, creates a situation in which some critical stage of the agent's task is unable to finish because more work is continuously created for it to do. Agent live-lock differs from agent deadlock in that the live-locked agent is not blocked or waiting for anything, but is continuously given tasks to perform and can never catch up or achieve its goal.

Eavesdropping

The classical eavesdropping threat involves the interception and monitoring of secret communications. The threat of eavesdropping, however, is further exacerbated in mobile agent systems because the agent platform can not only monitor communications, but also can monitor every instruction executed by the agent, all the unencrypted or public data it brings to the platform, and all the subsequent data generated on the platform. Since the platform has access to the agent's code, state, and data, the visiting agent must be wary of the fact that it may be exposing proprietary algorithms, trade secrets, negotiation strategies, or other sensitive information. Even though the agent may not be directly exposing secret information, the platform may be able to infer meaning from the types of services requested and from the identity of the agents with which it communicates. For example, someone's agent may be communicating with a travel agent, although the content of the message may not be exposed, this communication may indicate that the person on whose behalf the agent is acting is planning a trip and will be away from their home in the near future. The platform may share this information it has inferred with a suitcase manufacturer that may begin sending unsolicited advertisements, or even worse, the platform administrators may share this information with thieves who may target the home of the traveler.

Alteration

When an agent arrives at an agent platform it is exposing its code, state, and data to the platform. Since an agent may visit several platforms under various security domains throughout its lifetime, mechanisms must be in place to ensure the integrity of the agent's code, state, and data. A compromised or malicious platform must be prevented from modifying an agent's code, state, or data without being detected. Modification of an agent's code, and thus the subsequent behavior of the agent on other platforms, can be detected by having the original author digitally sign the agent's code. Detecting malicious changes to an agent's state during its execution or the data an agent has produced while visiting the compromised platform does not yet have a general solution. The agent platform may be running a modified virtual machine, for example, without the agent's knowledge, and the modified virtual machine may produce erroneous results.

A mobile agent that visits several platforms on its itinerary is exposed to a new risk each time it is in transit and each time it is instantiated on a new platform. The party responsible for the malicious alteration of an agent's code, state, or data if not immediately detected may be impossible to track down after the agent has visited other platforms and undergone countless changes of state and data. Although checkpointing and rollback of mathematical

computations may be possible in non-agent environments, mobile agent frameworks make this task extremely difficult since an agent's final state and data on a platform may be the result of a series of non-deterministic events that depend on the behavior of autonomous agents whose previous behavior cannot be recreated.

The security risks resulting from an agent moving from its home platform to another is referred to as the "single-hop" problem, while the security risks resulting from an agent visiting several platforms is referred to as the "multi-hop" problem. The risks associated with the single-hop problem are easier to mitigate than the risks associated with a multihop scenario, since the protection mechanisms within the trust environment of the home platform are more difficult to use in the latter situation.

Agent platforms can also tamper with agent communications. Tampering with agent communications, for example, could include deliberately changing data fields in financial transactions or even changing a "sell" message to a "buy" message. This type of goal-oriented alteration of the data is more difficult than simply corrupting a message, but the attacker clearly has a greater incentive and reward, if successful, in a goal-oriented alteration attack.

Other-to-Agent Platform

The other-to-agent platform category represents the set of threats in which external entities, including agents and agent platforms, threaten the security of an agent platform. This set of threats includes masquerading, denial of service, unauthorized access, and copy and replay.

Masquerade

Agents can request platform services both remotely and locally. An agent on a remote platform can masquerade as another agent and request services and resources for which it is not authorized. Agents masquerading as other agents may act in conjunction with a malicious platform to help deceive another remote platform or they may act alone. A remote platform can also masquerade as another platform and mislead unsuspecting platforms or agents about its true identity.

Unauthorized Access

Remote users, processes, and agents may request resources for which they are not authorized. Remote access to the platform and the host machine itself must be carefully protected, since conventional attack scripts freely available on the Internet can be used to subvert the operating system and directly gain control of all resources. Remote administration of the platform's attributes or security policy may be desirable for an administrator that is responsible for several distributed platforms, but allowing remote administration may make the system administrator’s account or session the target of an attack.

Denial of Service

Agent platform services can be accessed both remotely and locally. The agent services offered by the platform and inter-platform communications can be disrupted by common denial of service attacks. Agent platforms are also susceptible to all the conventional denial of service attacks aimed at the underlying operating system or communication protocols.

Copy and Replay

Every time a mobile agent moves from one platform to another it increases its exposure to security threats. A party that intercepts an agent, or agent message, in transit can attempt to copy the agent, or agent message, and clone or retransmit it. For example, the interceptor can capture an agent’s "buy order" and replay it several times, having the agent buy more

than the original agent had intended. The interceptor may copy and replay an agent message or a complete agent.

2.4.3. Centralized Mobile Architectures

This section lists possible security threats to the centralized mobile architectures particularly 3G systems, detailing what the threats achieve, how they are carried out and where in the system they could occur. Threats are categorized as:

Unauthorized access to sensitive data (violation of confidentiality)

- **Eavesdropping:** An intruder intercepts messages without detection.
- **Masquerading:** An intruder hoaxes an authorized user into believing that they are the legitimate system to obtain confidential information from the user; or an intruder hoaxes a legitimate system into believing that they are an authorized user to obtain system service or confidential information.
- **Traffic analysis:** An intruder observes the time, rate, length, source, and destination of messages to determine a user's location or to learn whether an important business transaction is taking place.
- **Browsing:** An intruder searches data storage for sensitive information.
- **Leakage:** An intruder obtains sensitive information by exploiting processes with legitimate access to the data.
- **Inference:** An intruder observes a reaction from a system by sending a query or signal to the system. For example, an intruder may actively initiate communications sessions and then obtain access to information through observation of the time, rate, length, sources or destinations of associated messages on the radio interface.

Unauthorized manipulation of sensitive data (Violation of integrity)

- **Manipulation of messages:** Messages may be deliberately modified, inserted, replayed, or deleted by an intruder

Disturbing or misusing network services (leading to denial of service or reduced availability)

- **Intervention:** An intruder may prevent an authorized user from using a service by jamming the user's traffic, signaling, or control data.
- **Resource exhaustion:** An intruder may prevent an authorized user from using a service by overloading the service.
- **Misuse of privileges:** A user or a serving network may exploit their privileges to obtain unauthorized services or information.
- **Abuse of services:** An intruder may abuse some special service or facility to gain an advantage or to cause disruption to the network.
- **Repudiation:** A user or a network denies actions that have taken place.

Unauthorized access to services

- Intruders can access services by masquerading as users or network entities.
- Users or network entities can get unauthorized access to services by misusing their access rights.

Threats Associated with Attacks on the Radio Interface

The radio interface between the terminal equipment and the serving network represents a significant point of attack in 3G. The threats associated with attacks on the radio interface are split into the following categories:

Unauthorized access to data

- **Eavesdropping user traffic:** Intruders may eavesdrop user traffic on the radio interface.
- **Eavesdropping signaling or control data:** Intruders may eavesdrop signaling data or control data on the radio interface. This may be used to access security management data or other information, which may be useful in conducting active attacks on the system.
- **Masquerading as a communications participant:** Intruders may masquerade as a network element to intercept user traffic, signaling data or control data on the radio interface.
- **Passive traffic analysis:** Intruders may observe the time, rate, length, sources or destinations of messages on the radio interface to obtain access to information.
- **Active traffic analysis:** Intruders may actively initiate communications sessions and then obtain access to information through observation of the time, rate, length, sources or destinations of associated messages on the radio interface.

Threats to integrity

- **Manipulation of user traffic:** Intruders may modify, insert, replay or delete user traffic on the radio interface. This includes both accidental and deliberate manipulation.
- **Manipulation of signaling or control data:** Intruders may modify, insert, replay or delete signaling data or control data on the radio interface. This includes both accidental and deliberate manipulation.
  NB: Replayed data, which cannot be decrypted by an intruder, may still be used to conduct attacks against the integrity of user traffic, signaling data or control data.

Denial of service attacks

- **Physical intervention:** Intruders may prevent user traffic, signaling data and control data from being transmitted on the radio interface by physical means. An example of physical intervention is jamming.
- **Protocol intervention:** Intruders may prevent user traffic, signaling data or control data from being transmitted on the radio interface by inducing specific protocol failures. These protocol failures may themselves be induced by physical means.
- **Denial of service by masquerading as a communications participant:** Intruders may deny service to a legitimate user by preventing user traffic, signaling data or control data from being transmitted on the radio interface by masquerading as a network element.

Unauthorized access to services

- **Masquerading as another user:** An intruder may masquerade as another user towards the network. The intruder first masquerades as a base station towards the user, then hijacks his connection after authentication has been performed.

Threats Associated with Attacks on Other Parts of the System

Although attacks on the radio interface between the terminal equipment and the serving network represent a significant threat, attacks on other parts of the system may also be conducted. These include attacks on other wireless interfaces, attacks on wired interfaces, and attacks, which cannot be attributed to a single interface or point of attack. The threats associated with attacks on other parts of the system are split into the following categories:

Unauthorized access to data

- **Eavesdropping user traffic:** Intruders may eavesdrop user traffic on any system interface, whether wired or wireless.
- **Eavesdropping signaling or control data:** Intruders may eavesdrop signaling data or control data on any system interface, whether wired or wireless. This may be used to access security management data, which may be useful in conducting other attacks on the system.
- **Masquerading as an intended recipient of data:** Intruders may masquerade as a network element in order to intercept user traffic, signaling data or control data on any system interface, whether wired or wireless.
- **Passive traffic analysis:** Intruders may observe the time, rate, length, sources or destinations of messages on any system interface, whether wired or wireless, to obtain access to information.
- **Unauthorized access to data stored by system entities:** Intruders may obtain access to data stored by system entities. Access to system entities may be obtained either locally or remotely, and may involve breaching physical or logical controls.
- **Compromise of location information:** Legitimate user of a 3G service may receive unintended information about other users locations through (analysis of) the normal signaling or voice prompts received at call set up.

Threats to integrity

- **Manipulation of user traffic:** Intruders may modify, insert, replay or delete user traffic on any system interface, whether wired or wireless. This includes both accidental and deliberate manipulation.
- **Manipulation of signaling or control data:** Intruders may modify, insert, replay or delete signaling or control data on any system interface, whether wired or wireless. This includes both accidental and deliberate manipulation.
- **Manipulation by masquerading as a communications participant:** Intruders may masquerade as a network element to modify, insert, replay or delete user traffic, signaling data or control data on any system interface, whether wired or wireless.
- **Manipulation of applications and/or data downloaded to the terminal or USIM:** Intruders may modify, insert, replay or delete applications and/or data, which are downloaded to the terminal or USIM. This includes both accidental and deliberate manipulation.
- **Manipulation of the terminal or USIM behavior by masquerading as the originator of applications and/or data:** Intruders may masquerade as the originator of malicious applications and/or data downloaded to the terminal or USIM.
- **Manipulation of data stored by system entities:** Intruders may modify, insert or delete data stored by system entities. Access to system entities may be obtained either locally or remotely, and may involve breaching physical or logical controls.

Denial of service attacks

- **Physical intervention:** Intruders may prevent user or signaling traffic from being transmitted on any system interface, whether wired or wireless, by physical means. An example of physical intervention on a wired interface is wire cutting. An example of physical intervention on a wireless interface is jamming. Physical intervention involving interrupting power supplies to transmission equipment may be conducted on both wired and wireless interfaces. Physical intervention may also be conducted by delaying transmissions on a wired or wireless interface.
- **Protocol intervention:** Intruders may prevent user or signaling traffic from being transmitted on any system interface, whether wired or wireless, by inducing protocol failures. These protocol failures may themselves be induced by physical means.
- **Denial of service by masquerading as a communications participant:** Intruders may deny service to a legitimate user by preventing user traffic, signaling data or

control data from being transmitted by masquerading as a network element to intercept and block user traffic, signaling data or control data.

- **Abuse of emergency services:** Intruders may prevent access to services by other users and cause serious disruption to emergency services facilities by abusing the ability to make USIM-less calls to emergency services from 3G terminals. If such USIM-less calls are permitted then the provider may have no way of preventing the intruder from accessing the service.

Repudiation

- **Repudiation of charge:** A user could deny having incurred charges, perhaps through denying attempts to access a service or denying that the service was actually provided.
- **Repudiation of user traffic origin:** A user could deny that he sent user traffic.
- **Repudiation of user traffic delivery:** A user could deny that he received user traffic.

Unauthorized access to services

- **Masquerading as a user:** Intruders may impersonate a user to utilize services authorized for that user. The intruder may have received assistance from other entities such as the serving network, the home environment or even the user himself.
- **Masquerading as a serving network:** Intruders may impersonate a serving network, or part of a serving network's infrastructure, perhaps with the intention of using an authorized user's access attempts to gain access to services himself.
- **Masquerading as a home environment:** Intruders may impersonate a home environment perhaps with the intention of obtaining information, which enables him to masquerade as a user.
- **Misuse of user privileges:** Users may abuse their privileges to gain unauthorized access to services or to simply intensively use their subscriptions without any intent to pay.
- **Misuse of serving network privileges:** Serving networks may abuse their privileges to gain unauthorized access to services. The serving network could e.g. misuse authentication data for a user to allow an accomplice to masquerade as that user or just falsify charging records to gain extra revenues from the home environment.

Threats Associated with Attacks on the Terminal and UICC/USIM

- **Use of a stolen terminal and UICC:** Intruders may use stolen terminals and UICCs to gain unauthorized access to services.
- **Use of a borrowed terminal and UICC:** Users who have been given authorization to use borrowed equipment may misuse their privileges perhaps by exceeding agreed usage limits.
- **Use of a stolen terminal:** Users may use a valid USIM with a stolen terminal to access services.
- **Manipulation of the identity of the terminal:** Users may modify the IMEI of a terminal and use a valid USIM with it to access services.
- **Integrity of data on a terminal:** Intruders may modify, insert or delete applications and/or data stored by the terminal. Access to the terminal may be obtained either locally or remotely, and may involve breaching physical or logical controls.
- **Integrity of data on USIM:** Intruders may modify, insert or delete applications and/or data stored by the USIM. Access to the USIM may be obtained either locally or remotely.
- **Eavesdropping the UICC-terminal interface:** Intruders may eavesdrop the UICC-terminal interface.

- **Masquerading as an intended recipient of data on the UICC-terminal interface:** Intruders may masquerade as a USIM or a terminal in order to intercept data on the UICC-terminal interface.
- **Manipulation of data on the UICC-terminal interface:** Intruders may modify, insert, replay or delete user traffic on the UICC-terminal interface.
- **Confidentiality of certain user data in the terminal or in the UICC/USIM:** Intruders may wish to access personal user data stored by the user in the terminal or UICC, e.g. telephone books.
- **Confidentiality of authentication data in the UICC/USIM:** Intruders may wish to access authentication data stored by the service provider, e.g. authentication key.

2.4.4. Decentralized Mobile Architectures

Decentralized mobile architectures include wireless connected peers using ad-hoc networks. Threats to these systems are typically divided into *passive* and *active* classes. These two broad classes are then subdivided into other types of threats.

Passive Attack

An attack in which an unauthorized party gains access to an asset and does not modify its content (i.e., eavesdropping). Passive attacks can be either eavesdropping or traffic analysis (sometimes called traffic flow analysis). These two passive attacks are described below:

- **Eavesdropping:** The attacker monitors transmissions for message content. An example of this attack is a person listening into the transmissions on a LAN between two workstations or tuning into transmissions between a wireless handset and a base station.
- **Traffic analysis:** The attacker, in a more subtle way, gains intelligence by monitoring the transmissions for patterns of communication. A considerable amount of information is contained in the flow of messages between communicating parties.

Active Attack

An attack whereby an unauthorized party makes modifications to a message, data stream, or file. It is possible to detect this type of attack but it may not be preventable. Active attacks may take the form of one of four types (or combination thereof): masquerading, replay, message modification, and denial-of-service (DoS). These attacks are defined below:

- **Masquerading:** The attacker impersonates an authorized user and thereby gains certain unauthorized privileges.
- **Replay:** The attacker monitors transmissions (passive attack) and retransmits messages as the legitimate user.
- **Message modification:** The attacker alters a legitimate message by deleting, adding to, changing, or reordering it.
- **Denial-of-service:** The attacker prevents or prohibits the normal use or management of communications facilities.

The risks associated with 802.11 are the result of one or more of these attacks. The consequences of these attacks include, but are not limited to, loss of proprietary information, legal and recovery costs, tarnished image, and loss of network service.

Loss of Confidentiality

Confidentiality is the property with which information is not made available or disclosed to unauthorized individuals, entities, or processes. This is, in general, a fundamental security requirement for most organizations. Due to the broadcast and radio nature of wireless technology, confidentiality is a more difficult security requirement to meet in a wireless network. Adversaries do not have to tap into a network cable to access network resources.

Moreover, it may not be possible to control the distance over which the transmission occurs. This makes traditional physical security countermeasures less effective.

Passive eavesdropping of native 802.11 wireless communications may cause significant risk to an organization. An adversary may be able to listen in and obtain sensitive information including proprietary information, network IDs and passwords, and configuration data. This risk is present because the 802.11 signals may travel outside the building perimeter or because there may be an "insider." Because of the extended range of 802.11 broadcasts, adversaries can potentially detect transmission from a parking lot or nearby roads. This kind of attack, performed through the use of a wireless network analyzer tool or *sniffer*, is particularly easy for two reasons: 1) frequently confidentiality features of WLAN technology are not even enabled, and 2) because of the numerous vulnerabilities in the 802.11 technology security, as discussed above, determined adversaries can compromise the system.

Wireless packet analyzers are readily available on the Internet today. They are commonly used for breaking into wireless networks. They can take advantage of flaws in the key-scheduling algorithm that was provided for implementation of RC4, which forms part of the original WEP standard. To accomplish this, they require only a computer running the Linux operating system and a wireless network card. The software passively monitors the WLAN data transmissions and computes the encryption keys after at least 100 MB of network packets have been *sniffed*. On a highly saturated network, collecting this amount of data may only take three or four hours; if traffic volume is low, it may take a few days. For example, a busy data access point transmitting 3,000 bytes at 11 Mbps will exhaust the 24-bit IV space after approximately 10 hours. If after ten hours the attacker recovers two cipher texts that have been using the same key stream, both data integrity and confidentiality may be easily compromised. After the network packets have been received, the fundamental keys may be guessed in less than one second. Once the malicious user knows the WEP key, that person can read any packet traveling over the WLAN.

Another risk to loss of confidentiality through simple eavesdropping is broadcast monitoring. An adversary can monitor traffic, using a laptop in promiscuous mode, when an access point is connected to a hub instead of a switch. Hubs generally broadcast all network traffic to all connected devices, which leaves the traffic vulnerable to unauthorized monitoring. Switches, on the other hand, can be configured to prohibit certain attached devices from intercepting broadcast traffic from other specified devices. For example, if a wireless access point was connected to an Ethernet hub, a wireless device that is monitoring broadcast traffic could intercept data intended for wired and wireless clients. Consequently, agencies should consider using switches instead of hubs for connections to wireless access points.

WLANs risk loss of confidentiality following an active attack as well. Sniffing software as described above can obtain user names and passwords (as well as any other data traversing the network) as they are sent over a wireless connection. An adversary may be able to masquerade as a legitimate user and gain access to the wired network from an access point (AP). Once "on the network," the intruder can scan the network using purchased or publicly and readily available tools. The malicious eavesdropper then uses the user name, password, and IP address information to gain access to network resources and sensitive corporate data.

Lastly, rogue APs pose a security risk. A malicious or irresponsible user could, physically and surreptitiously, insert a rogue AP into a closet, under a conference room table, or any other hidden area within a building. The rogue AP could then be used to allow unauthorized individuals to gain access to the network. As long as its location is in close proximity to the users of the WLAN, and it is configured so as to appear as a legitimate AP to wireless clients, then the rogue AP can successfully convince wireless clients of its legitimacy and cause them to send traffic through it. The rogue AP can intercept the wireless traffic between an authorized AP and wireless clients. It need only be configured with a stronger signal than the existing AP to intercept the client traffic. A malicious user can also gain access to the wireless network through APs that are configured to allow access without authorization. It is

also important to note that rogue access points need not always be deployed by malicious users. In many cases, rogue APs are often deployed by users who want to take advantage of wireless technology without the approval of the IT department. Additionally, since rogue APs are frequently deployed without the knowledge of the security administrator, they are often deployed without proper security configurations.

Loss of Integrity

Data integrity issues in wireless networks are similar to those in wired networks. Because organizations frequently implement wireless and wired communications without adequate cryptographic protection of data, integrity can be difficult to achieve. A hacker, for example, can compromise data integrity by deleting or modifying the data in an e-mail from an account on the wireless system. This can be detrimental to an organization if important e-mail is widely distributed among e-mail recipients. Because the existing security features of the 802.11 standard do not provide for strong message integrity, other kinds of active attacks that compromise system integrity are possible. As discussed before, the WEP based integrity mechanism is simply a linear CRC. Message modification attacks are possible when cryptographic checking mechanisms such as message authentication codes and hashes are not used.

Loss of Network Availability

A denial of network availability involves some form of DoS attack, such as jamming. Jamming occurs when a malicious user deliberately emanates a signal from a wireless device in order to overwhelm legitimate wireless signals. Jamming may also be inadvertently caused by cordless phone or microwave oven emissions. Jamming results in a breakdown in communications because legitimate wireless signals are unable to communicate on the network. Non-malicious users can also cause a DoS. A user, for instance, may unintentionally monopolize a wireless signal by downloading large files, effectively denying other users access to the network. As a result, agency security policies should limit the types and amounts of data that users are able to download on wireless networks.

Other Security Risks

With the prevalence of wireless devices, more users are seeking ways to connect remotely to their own organization's networks. One such method is the use of untrusted, third party networks. Conference centers, for example, commonly provide wireless networks for users to connect to the Internet and subsequently to their own organizations while at the conference. Airports, hotels, and even some coffee franchises are beginning to deploy 802.11 based publicly accessible wireless networks for their customers, even offering VPN capabilities for added security.

These untrusted public networks introduce three primary risks: 1) because they are public, they are accessible by anyone, even malicious users; 2) they serve as a bridge to a user's own network, thus potentially allowing anyone on the public network to attack or gain access to the bridged network; and 3) they use high-gain antennas to improve reception and increase coverage area, thus allowing malicious users to eavesdrop more readily on their signals.

Lastly, by connecting to their own networks via an untrusted network, users may create vulnerabilities for their company networks and systems unless their organizations take steps to protect their users and themselves.

2.5. Applications

2.5.1. Synchronous Applications

File Transfer Protocol (FTP) – an example of synchronous applications is considered in this section. A variety of FTP servers incorrectly manage buffers in a way that can lead to remote intruders executing arbitrary code on the FTP server. Currently, many of the most common FTP vulnerabilities are using the *FTP Bounce Attack*. The Bounce attack involves the user (attacker) opening a control connection with an FTP server.

FTP Bounce Attack

One of the most alarming things about a Bounce Attack is that it is RFC complaint, as FTP was initially promoted to be easy and robust to use, not secure. By allowing the user (in this case attacker) to define the parameters of the data connection, the writers of the RFC (and many subsequent programs implementing FTP) have left themselves open to this attack.

Bounce Attacks occur when a user (attacker) is connected to an FTP server by his control connection and uses the ftp PORT command to get the FTP server to open what the FTP server believes is a data connection on the specified port. Any other machine receiving a *data* connection on TCP port 23, however, will probably believe that it is receiving a telnet request. Also, it should be kept in mind that FTP was specifically written to allow transfer between two hosts, both remote to the *user*. So, the RFC clearly explains that a compliant implementation will allow for redirection of the data connection to a different host than the one that initiated the control connection!

PORT Command

The PORT command in Active FTP connections is perhaps the biggest problem with any FTP server. According to RFC959:

If this command is used, the argument is the concatenation of a 32-bit Internet host address and a 16-bit TCP port address. This address information is broken into 8-bit fields and the value of each field is transmitted as a decimal number (in character string representation).

A logical extension of the above leads us to understand that simply by connecting to an FTP server, one can tell that FTP server to connect to another host, and on what port to connect. In the RFC, there are NO restrictions on this behavior. For this reason, FTP PORT command will be filled with vulnerabilities in the future. It is easy to see, for example, how easy it would be to complete a port scan of a network by using an FTP server that was strictly following the RFC. Or how a server on a LAN might be convinced to *respond* to a *connection* from an external host. By providing us with functionality that is simple and robust, the RFC has provided almost all networks with big holes in security.

GLOB Vulnerability

Filename “globing” is the common practice of using wildcard characters (like “*” in Unix/Linux) to perform operations on lots of files with common strings in their names. The essence of this problem is that, although not required by the RFC, many FTP implementations allow for file name “globing.” While in itself this is not a bad thing, it can be exploited by creating very large amounts of data being passed to the main command processing routines. This can lead to buffer overflows. Depending on how the system is *trained* to handle the overflow, arbitrary code can be run on the server at this time. (Using the permissions of the FTP process daemon)

While this vulnerability has been fixed on a large amount of FTP packages at the time of this writing, it illustrates another vulnerability commonly found in FTP software – the software itself. Again, this demonstrates the need to keep the servers well patched.

Stateful Firewalls and FTP

CheckPoint Firewall-1 is of particular interest, as it operates on the “stateful inspection” principle, which commonly examines the source and destination addresses and port numbers. In the case of PASV FTP, CheckPoint is required to keep track of another bit of information: the PASV port number sent to the client from the FTP server.

During a normal PASV-FTP connection, the user (client) sends the FTP server a PASV command. This is related to the PORT command in that it is used to determine which TCP port the data connection will be established on. The difference here is that when the PASV command is issued, the SERVER now is responsible to identify a non-default data port. In a common FTP situation, the client (on TCP port 21) initiates the control connection. The client then proceeds (on TCP port 20) to open the data transfer connection. It is perhaps easier to think of active FTP sessions as “Client-Driven” and PASV-FTP sessions as “Server-Driven”. Normally, the client creates all connections, and a stateful inspection type firewall can “watch” the source port in the first packet the firewall lets through. In any event, with PASV-FTP the CheckPoint firewall must watch for the PASV port that is passed from the server to the client, and dynamically allow that port number through. This behavior is on by default in Checkpoint 4.0, as many common programs (like Microsoft Internet Explorer and Netscape Communicator) use PASV-FTP in their FTP implementations.

CheckPoint Firewall-1, then, looks for the string “227” (an FTP message code meaning “Entering Passive Mode”) and extracts the destination IP and port given in the packet payload. The issue is that the destination IP and port can be that of the firewall and there is no logic in the firewall to prevent this. A crafty person, therefore, can take control of firewall by using this vulnerability to run arbitrary code on it.

Again, the blame for this PASV-FTP problem is perhaps half CheckPoint, half RFC, but that will not help users with FTP servers on their networks. It should also be pointed out that there are various patches for CheckPoint, Operating systems as well as FTP implementations that will reduce or eliminate this particular vulnerability.

2.5.2. Asynchronous Applications

Asynchronous web services (WS) are considered in this section. In order to illustrate the nature of the threat, it is worthwhile discussing some of the types of attacks that are likely as Web Services architectures get deployed. Here are ten of the most likely techniques, employing multiple classes of input or target vulnerabilities, which will be used to attack the technology:

Coercive Parsing

XML is already recognized as a standard file format for many applications. As the obvious successor to legacy ASCII and presentation-oriented html, its position is unchallenged. This is easily seen by the number of grammars that claim XML as their parent.

The basic premise of a coercive parsing attack is to exploit the *legacy bolt-on* - XML-enabled components in the existing infrastructure that are operational. Even without a specific Web Services application these systems are still susceptible to XML based attacks that whose main objective is either to overwhelm the processing capabilities of the system or install malicious mobile code.

Parameter Tampering

Parameters are used to convey client-specific information to the Web service in order to execute a specific remote operation. Since instructions on how to use parameters are explicitly described within a WSDL document, malicious users can play around with different parameter options in order to retrieve unauthorized information. For example by submitting special characters or unexpected content to the Web service can cause a denial of service condition or illegal access to database records. An attacker can embed, for example,

command line code into a document that is parsed by an application that can create a command shell to execute the command.

Recursive Payloads

One of the strengths of XML is its ability to nest elements within a document to address the need for complex relationships among elements. The value is easy to see with forms that have a form name or purpose that contains many different value elements, such as a purchase order that incorporates shipping and billing addresses as well as various items and quantities ordered. One can intuitively acknowledge the value of nesting elements three or four levels, perhaps more. An attacker can easily create a document that attempts to stress and break an XML parser by creating a document that is 10,000 or 100,000 elements deep.

Oversize Payloads

XML is verbose by design in its markup of existing data and information, so file size must always be considered. While an enterprise's programmers and analysts will work to limit the size of a document, there are a number of reasons to have XML documents that are hundreds of megabytes or gigabytes in size. Sometimes this is a function of converting a batch file transfer process into real-time. It may also be anticipated in the multimedia (e.g. digital video) world where gigabyte files are the norm. Or, it could be an attacker again exercising the parser to execute a denial-of-service attack. Parsers based on the Document Object Model (DOM) are especially susceptible to this attack given its need to model the entire document in memory prior to parsing

Schema Poisoning

XML Schemas provide formatting instructions for parsers when interpreting XML documents. Schemas are used for all of the major XML standard grammars coming out susceptible to poisoning. An attacker may attempt to compromise the schema in its stored location and replace it with a similar but modified one.

Denial-of-service attacks against the grammar are straightforward if the schema is compromised. In addition, the door is open to manipulate data if data types are compromised, like changing dates to numbers when the application is performing arithmetic operations, or modifying the encoding to allow for data obfuscation that eventually gets through to a parser and re-formed into an attack, in the same way a Unicode attack can traverse directories through web servers.

WSDL Scanning

Web Services Description Language (WSDL) is an advertising mechanism for web services to dynamically describe the parameters used when connecting with specific methods. These files are often built automatically using utilities. These utilities, however, are designed to expose and describe all of the information available in a method.

In addition, the information provided in a WSDL file may allow an attacker to guess at other methods. For example, a service that offers stock quoting and trading services may advertise query methods like requestStockQuote, however also includes an unpublished transactional method such as tradeStockQuote. It is simple for a persistent hacker to cycle thru method string combinations (similar to cryptographic cipher unlocking) in order to discover unintentionally related or unpublished application programming interfaces.

Routing Detours

The WS-Routing specification provides a way to direct XML traffic through a complex environment. It operates by allowing an interim way station in an XML path to assign routing instructions to an XML document. If one of these web services way stations is compromised, it may participate in a man-in-the-middle attack by inserting bogus routing instructions to point a confidential document to a malicious location. From that location, then, it may be

possible to forward on the document, after stripping out the malicious instructions, to its original destination.

External Entity Attack

Another benefit of XML is its ability to build documents dynamically at the time of insertion by pointing to a uniform resource identifier (URI) where the actual data exists. These external entities may not be trustworthy. An attacker can then replace the data being collected with malicious data.

SQL Injection

Database parsers are aimed at native database languages in the same fashion as SQL injection, SQL injection could allow an attacker to execute multiple commands in an input field by using native command separators like ‘;’ or pipes. This capability may allow an attacker to execute native stored procedures or invalidated SQL commands.

Replay Attack

Similar to the “network ping of death” a hacker can issue repetitive SOAP message requests in a bid to overload a Web service. This type of network activity will not be detected as an intrusion because the source IP is valid, the network packet behavior is valid and the HTTP request is well formed. However, the business behavior is not legitimate and constitutes an XML-based intrusion. In this manner, a completely valid XML payload can be used to issue a denial of service attack.

Chapter 3

Towards a Comprehensive Security Services Model

During the course of this chapter, we will be going over many important security terms. While some of the terms covered provide the background as to how security works, there are some important concepts that should be highlighted. This is due to the fact that some areas within security require a precise understanding of their concepts. Also, some security components may work slightly different within a large-scale environment as opposed to a standard network.

3.1. Fundamental Concepts

Below are some important security concepts that will be described in greater detail throughout the chapter.

- **Symmetric encryption:** Using the same secret key to provide encryption and decryption of data.
- **Asymmetric encryption:** Using two different keys for encryption and decryption. The public key encryption technique is the primary example of this using a “public key” and a “private key” pair.
- **Secure Socket Layer/Transport Layer Security (SSL/TLS):** These are essentially the same protocol, but are referred to one another differently. TLS has been renamed by the IETF, but they are based on the same RFC.
- **Public Key Infrastructure (PKI):** The different components, technologies, and protocols that make up a PKI environment.
- **Mutual Authentication:** Instead of using an LDAP repository to hold the public key (PKI), two parties who want to communicate with one another use their public key stored in their digital certificate to authenticate with one another.

These are all important concepts to remember and will give you a head start in understanding how grid security works.

3.1.1. Symmetric Key Encryption

Symmetric key encryption is based on the use of one shared secret key to perform both the encryption and decryption of data. To ensure that the data is only read by the two parties (sender and receiver); the key has to be distributed securely between the two parties and no others. If someone should gain access to the secret key that is used to encrypt the data, they would be able to decrypt the information. This form of encryption is much faster than asymmetric encryption.

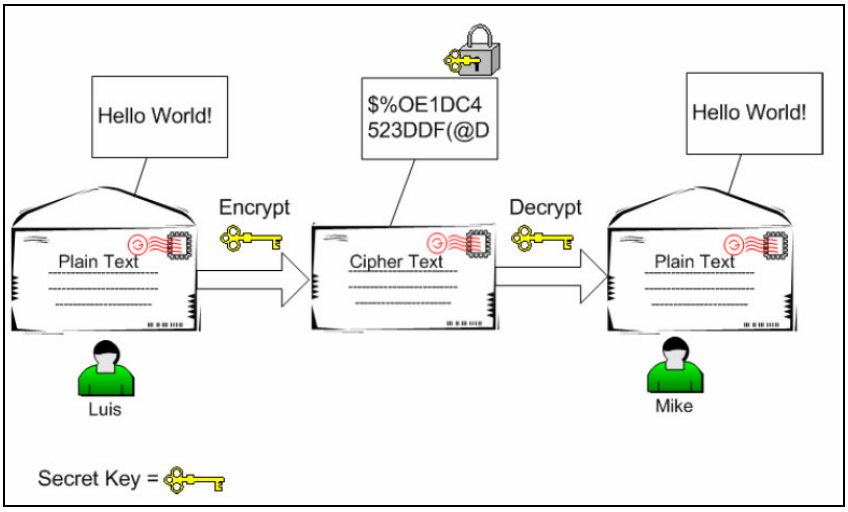

Figure 3.1: Symmetric key encryption using a shared secret key

Here are some commonly used examples of a symmetric key cryptosystem:

- Data Encryption Standard (DES): 56-bit key plus 8 parity bits, developed by IBM in the middle 1970s
- Triple-DES: 112-bit key plus 16 parity bits or 168-bit key plus 24 parity bits (that is, two to three DES keys)
- RC2 and RC4: Variable-sized key, often 40 to 128 bits long

To summarize, secret key cryptography is fast for both the encryption and decryption processes. However, secure distribution and management of keys is difficult to guarantee.

3.1.2. Asymmetric Key Encryption

Another commonly-used cryptography method is called public key cryptography. The RSA public key cryptography system is a prime example of this. In public key cryptography, an asymmetric key pair (a so-called a public key and a private key) is used. The key used for encryption is different from the one used for decryption. Public key cryptography requires the key owners to protect their private keys while their public keys are not secret at all and can be made available to the public. Normally, the public key is present in the digital certificate that is issued by the Certificate Authority.

The computation algorithm relating the public key and the private key is designed in such a way that an encrypted message can only be decrypted with the corresponding other key of that key pair, and an encrypted message cannot be decrypted with the encryption key (the key that was used for encryption). Whichever (public/private) key encrypts your data, the other key is required to decrypt the data. A message encoded with the public key, for instance, can only be decoded with the private key. One of the keys is designated as the public key because it is made available, publicly, via a trusted Certificate Authority, which guarantees the ownership of each of the public keys. The corresponding private keys are secured by the owner and never revealed to the public.

The public key system is used twice to completely secure a message between the parties. The sender first encrypts the message using his private key and then encrypts it again using the receiver's public key. The receiver decrypts the message, first using his private key and then the public key of the sender. In this way, an intercepted message cannot be read by

anyone else. Furthermore, any tampering with the message will make it not decrypt properly, revealing the tampering.

The asymmetric key pair is generated by a computation which starts by finding two vary large prime numbers. Even though the public key is widely distributed, it is practically impossible for computers to calculate the private key from the public key. The security is derived from the fact that it is very difficult to factor numbers exceeding hundreds of digits.
This mathematical algorithm improves security, but requires a long encryption time, especially for large amounts of data. For this reason, public key encryption is used to securely transmit a symmetric encryption key between the two parties, and all further encryption is performed using this symmetric key.

3.1.3. The Certificate Authority

A properly implemented Certificate Authority (CA) has many responsibilities. These should be followed diligently to achieve good security. The primary responsibilities are:

- Positively identify entities requesting certificates
- Issuing, removing, and archiving certificates
- Protecting the Certificate Authority server
- Maintaining a namespace of unique names for certificate owners
- Serve signed certificates to those needing to authenticate entities
- Logging activity

Within some PKI environments, a Registrant Authority (RA) works in conjunction with the CA to help perform some of these duties. The RA is responsible for approving or rejecting requests for the certificate of public keys and forwarding the user information to the CA. The RA normally has the responsibility of validating that the user's information is correct before the signed digital certificate is sent back to the user.

One of the critical issues within a PKI environment is guaranteeing the system's trustworthiness. Before a CA can sign and issue certificates for others, it has to do the same thing to itself so that its identity can be represented by its own certificate. That means a CA has to do the following:

1. The CA randomly generates its own key pair.
2. The CA protects its private key.
3. The CA creates its own certificate.
4. The CA signs its certificate with its private key.

The CA's Private Key

The CA's private key is one of the most important parts in the whole public key infrastructure. It is used, for example, by the CA to sign every issued digital certificate within the system. Thus, it is especially susceptible to attacks from hackers. If someone were to gain access to the CA's private key, they would be able to impersonate anyone within the environment. Therefore, it is very important to protect this key. Knowing how sensitive the private key is to the rest of environment, it is important to provide CA server with any available security measures. This includes restricting physical and remote access and monitoring and auditing of the server.

CA Cross Certification

Generally within a single environment, a CA will provide certificates to a fixed group of users. If two companies or virtual organizations (VOs) need to communicate and trust one another, this may require that both CAs trust one another or participate in cross certification. For example, Alice, an employee belonging to an organization with its own CA, may want to

run a job on grid computer Mike, who is outside the organization, and who belongs to a different CA.

In order to do so, the following should be considered:

- Alice and Mike need a way to obtain each other's public key certificates.
- Mike needs to be sure that he can trust Alice's CA. Alice needs to be sure that she can trust Mike's CA.

Resources from different security domains or VOs will need to trust each others' certificates, so the roles and relationships between CAs have to be defined. The purpose of creating such trust relationships is to eventually achieve a global, interoperable PKI and enlarge the distributed infrastructure. Once the relationship is established, both of the CA's can be configured to work with the entire system.

3.1.4. Digital Certificates

Digital certificates are digital documents that associate a resource with its specific public key. A certificate is a data structure containing a public key and pertinent details about the key owner. A certificate is considered to be a tamper-proof electronic ID when it signed by the Certification Authority for the grid environment.

Digital certificates, also called X.509 certificates, act very much like passports; they provide a means of identifying resources. Unlike passports, digital certificates can (and should) be distributed and copied without restriction, while people are normally very concerned about handing their passports to someone else. Certificates do not normally contain any confidential information and their free distribution does not create a security risk.

The important fact to know and understand about digital certificates is that the CA certifies that the enclosed public key belongs to the entity listed in the certificate. The technical implementation is such that it is considered extremely difficult to alter any part of a certificate without easy detection. The signature of the CA provides an integrity check for the digital certificate.

When a client wants to start a session with a resource, he/she does not attach the public key to the message, but the certificate instead. The recipient receives the communication with the certificate and then checks the signature of the Certificate Authority within the certificate. If the signature was signed by a certifier that he/she trusts, the recipient can safely accept that the public key contained in the certificate is really from the sender. This prevents someone from using a fraudulent public key to impersonate the public key owner.

Digital certificate contains the information about its user and his/her public key. When the user communicates with another party, the recipient will use his/her public key (contained in his/her digital certificate) to decrypt the SSL session ID, which is used to encrypt all data transferred between the nodes.

A digital certificate is made up of a unique distinguished name (DN) and certificate extensions that contain the information about the individual or host that is being certified. Some information in this section may contain the subject's e-mail address, organizational unit, or location.

Figure 3.2 is a graphical depiction of the digital certificate.

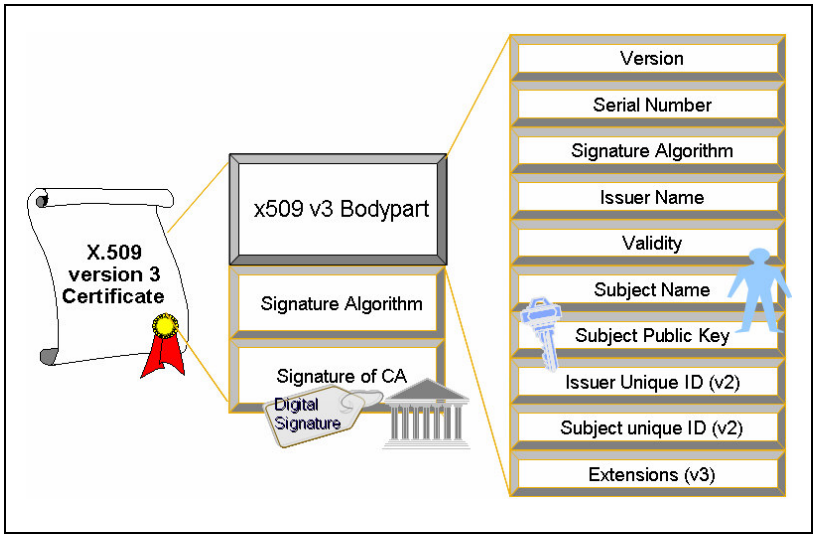

Figure 3.2: Digital certificate

Obtaining a client or a server certificate from a CA involves the following steps:

1. The user requiring certification generates a key pair (private key and certificate request containing the public key).
2. The user signs its own public key and any other information required by the CA. Signing the public key demonstrates that the user does, in fact, hold the private key corresponding to the public key.
3. The signed information is communicated to the CA. The private key remains with the client and should be stored securely. For instance, the private key could be stored in an encrypted form on a Smartcard, or on the user's personal computer.
4. The CA verifies that the user does own the private key of the public key presented.
5. The CA (or optionally an RA) needs to verify the user's identity. This can be done using out-of-band methods, for example, through the use of e-mail, telephone, or face-to-face communication. A CA (or RA) can use its own record system or another organization's record system to verify the user's identity.
6. Upon a positive identity check, the CA creates a certificate by signing the public key of the user, thereby associating a user to a public key. The certificate will be forwarded to the RA for distribution to the user.

Verification of the User

The authentication described above is a one-time authentication for the purpose of certificate issuance. This can be compared to the process when a government authority issues a passport to an individual. The passport then serves as an authentication mechanism when this individual travels to foreign countries. Just like passports, digital certificates can subsequently be used in daily operations for authenticating subjects to other parties that require authentication.

Certificate Revocation List

In other PKI environments that use directory services to store the public key, a certificate revocation list (CRL) is a means of notifying clients who wish to verify the revocation of certificates. CRLs are issued to mark some certificates unusable, even though their expiration has not come yet.

Path Validation

In order to verify that a certificate is valid, a check must be done to ensure that whoever signed the certificate is valid. This is how the path validation of a certificate is done. This is done to verify that the certificate path from the Root CA is valid and up the chain between the CA and client/server. This is especially important when explaining why delegated certificates are valid within the environment. This delegation is an extension to PKI and is not normally allowed [43]. As long as the path is valid within the delegated certificate, the certificate will not be rejected.

PKI Directory Services

Within some PKI environments, the signed keys are published to a public directory for easy retrieval. Instead of having the clients handle the mutual authentication, an external server is responsible for handling the authentication process. A good example of this process is the MyProxy server [44]. In this example, the user would authenticate to the Web portal, which would request the user's online credentials that are stored in the directory. Upon this authentication, the proxy would extract the DN within their digital certificate and match their credentials with the public key stored within the directory. If they two keys matched up, the user would be given access to resources within the system.

3.2. Security Objectives

They reflect the stated intent to counter identified threats and/or comply with any organizational security policies and usage assumptions.

3.2.1. Availability

It is the property of being accessible and useable upon demand by an authorized entity [41]. Availability functionality of security is responsible for ensuring that the system is available to authorized users. Availability is sometimes confused with reliability. The latter is a measure of how few failures happen to the system components. Naturally, the more reliable system components are the higher availability of the system. The reverse is not always true because availability can be (and usually is) achieved by other means than increase of reliability. Availability is a functionality of system security because most of security breaches potentially decrease overall system availability.

3.2.2. Confidentiality

It is the property that information is not made available or disclosed to unauthorized individuals, entities or processes [41]. Information confidentiality functionality is responsible for protecting information from unauthorized disclosure. Sending letters in sealed envelopes as opposed to postcards is a well known computer unrelated example of confidentiality services. By enclosing a letter in an envelope, one protects its contents from being accessed by anyone else but its intended reader. In computer communications, confidentiality is usually achieved by encrypting information and making only sender in a position of decrypting the received data. Making sure that information left in a system after an application is not read by any other application is also responsibility of confidentiality services.

Confidentiality function of security services is sometimes confused with confidential. They are not equivalent! Confidential is used to express the sensitivity level of particular information. Many with the healthcare background use terms *privacy* and *confidentiality* interchangeably. It is due to the fact that the healthcare domain puts very different meaning in the word confidentiality than the technically oriented computer security world. The Massachusetts Medical Society Policy on Patient Privacy and Confidentiality explains the difference in [42]:

Although the words privacy and confidentiality often are used interchangeably, they are related but not synonymous terms. Privacy derives from the concepts of personal freedom and autonomy, and involves the ability of an individual to control the release or dissemination of information that relates to him/herself. Confidentiality, on the other hand, arises in a relationship, when an individual gives private information to another on the condition of or with the understanding that the other will not further disclose it, or will disclose it only to the extent that the individual directs.

In this thesis, we will use the term confidentiality only in the context of distributed system security. To avoid any confusion, we do not use the terms privacy or confidentiality in the healthcare domain meanings.

3.2.3. Integrity

It is the property that data has not been altered or destroyed in an unauthorized manner [41]. Integrity service is responsible for providing the protection of data from unauthorized modifications. Since it is almost impossible to enforce access control over information traveling through multiple intermediate hops in inherently insecure networks, integrity becomes a very important asset of secure communications in distributed systems. In most of traditional systems that provide secure communications, integrity is achieved by signing messages digitally. The idea of digital signatures comes from check-sum computation in communication protocols. The main difference between check-sums and digital signatures is the ability to ensure that the signature was generated by the original sender.

3.3. Security Functions

They are the implementation of security policies defined to withstand certain security threats and risks.

3.3.1. Authentication

It is the evidence that an entity is the one claimed [41]. Authentication functionality of system security is responsible for making sure that a user or a service is who they claim to be. Sometimes, the word identification is used instead of authentication to mean the same. Authentication part of security deals only with user or service identities. It is not responsible for access control, confidentiality or any other security functionalities. Though, it might use confidentiality and integrity to protect information exchanged between the system and say the user during the authentication phase.

3.3.2. Authorization

It is the granting of rights, which includes the granting of access based on access rights [41]. Authorization functionality is responsible for making decisions about what users and what services can access what system services and for endorsing those decisions. Authorization cannot be enforced without reliable authenticating functionality of a system. Before access rights decisions can be made, it is critical to identify a user or a service. Authorization decisions are based on access control policies. Such policies can be very rudimentary ("grant access to anyone") or very complex ("Give access to HIV information of patient X only to a user who has status of 'Attending physician for patient X' when such a user is located at her hospital office and only if the patient X gave a consent to disclose her HIV information and when it is before 2 weeks after the patient X was discharged").

3.3.3. Access Control

It is often interchanged with the term authorization. Access control policies are expressed in the form of access control rules. A set of access control rules constitute access control

language that allows mapping of the application system business model into the access control model supported by the particular distributed system authorization services.

UNIX access control rules are a good example of a basic access control language. In UNIX, each resource including processes, files and devices is owned by some user (owner) and group. UNIX access rules specify what type of access right (read, write, execute) is granted to the resource owner, group, and the rest of the world in regard to this resource. In order for a user to perform access operations granted to its owner, the user has to have the same identity as the resource owner; to perform group access operations, the user has to be a member of the same group as the resource; to perform operations allowed to "the rest of the world" the user does not have to have any special rights. UNIX access rules are therefore simple:

If you are the *owner* this what access operations you can invoke,
otherwise
if you are a *member of the same group*, this is what access operations you can invoke,
otherwise
you have the same access rights as anyone else in the system.

3.3.4. Accountability

It is the property that ensures that the actions of an entity can be traced [41]. Accountability functionality is responsible for making users accountable for their security-relevant actions. Accountability service is an important part of any security system since it provides virtually the only way to monitor security activities in the system and to detect security breaches as well as to provide proof that a particular action was requested and/or a particular message was sent/received later in court. Accountability requires authentication to have reliable information about identity of involved parties.

Accountability is generally achieved via security audit and non-repudiation functionalities. Security audit is to facilitate an independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established polices and operational procedures, to detect security breaches and to recommend any indicated changes in control policy and procedures [41]. Non-repudiation functionality is to protect against originator of a message or action denying that it originated the message or the action as well as against the recipient of a message or action denying that he or she has received the message or was requested action.

3.4. Contemporary Issues

These issues have been raised due to the big scale, heterogeneous and mobile nature of the modern days systems and applications. These factors exacerbate the state of security and hence they should be adequately addressed to triumph over the security chinks in the system's armor.

3.4.1. Interoperability

Services that traverse multiple domains and hosting environments need to be able to interact with each other, thus introducing the need for interoperability at multiple levels:

- At the *protocol level*, some mechanisms are required to allow domains to exchange messages. This can be achieved via SOAP/HTTP, for example.
- At the *policy level*, secure interoperability requires that each party be able to specify any policy it may wish in order to engage in a secure conversation – and that policies expressed by different parties can be made mutually comprehensible. Only then can the parties attempt to establish a secure communication channel and security context upon mutual authentication, trust relationship, and adherence to each other's policy.

- At the *identity level*, mechanisms are required for identifying a user from one domain in another domain. This requirement goes beyond the need to define trust relationships and achieve federation between security mechanisms (e.g., from Kerberos tickets to X.509 certificates). Irrespective of the authentication and authorization model, which can be group-based, role-based or other attribute-based, many models rely on the notion of an identity for reasons including authorization and accountability [31].

3.4.2. Extensibility

A security policy model always evolves; accordingly, the design of a security system using that policy model should reflect the changes. Using role-based access control (RBAC) as an example, currently it supports role hierarchy, static separation of duty relations, and dynamic separation of duty relations. As research on RBAC progresses, more concerns have been and will be covered. So the model hierarchy of RBAC is quickly becoming more and more complicated, which requires that the security system supporting RBAC be flexible and extensible. To address this issue at the design level, we propose an aspect-oriented approach to designing flexible and extensible security systems where the user is provided with future proof functionality in the form of an extensible security architecture that allows alternative security services to be *plugged-in* as required. A unique feature is that communities of users with different security services will be able to securely interact/ collaborate with each other.

3.4.3. Adaptability

In the today's security architectures, it is getting indispensable to implement dynamically adaptable security services, where the security mechanisms are changed at runtime in reaction to changed security requirements (e.g., suspected intrusion) or changes in available resources. The security framework should make it easy to activate and deactivate micro-protocols at runtime, and the coordination mechanisms should allow adaptations across machines and across system layers to occur smoothly without interrupting normal operation. Our proposed security model use fine-grain configurability and fast adaptation ability as the basis for an *inherently survivable security architecture* that can automatically react to threats in the execution environment.

3.4.4. Mobility

The promise of anywhere, anytime access to critical information and the adoption of mobile data devices are propelling the development of mobile applications. As businesses begin extending information to the mobile channel, they seek guarantees that the information will be transmitted securely to the end user. Highly sensitive information, such as financial and proprietary, will form the foundation of successful mobile applications on the mobile Internet. Security, therefore, is an essential element for the continued adoption of mobile applications. It should scale security levels to satisfy today's needs, incorporate new standards that will be adopted by the mobile market and make secure access to the mobile devices through encryption depending upon the memory limitation of the mobile device. For example, the level of security on the mobile devices is bound to increase with higher versions of WAP that support new innovative encryption technology and improve the level of security currently on the WTLS. Figure 3.3 depicts the zone to be covered by the security of mobility.

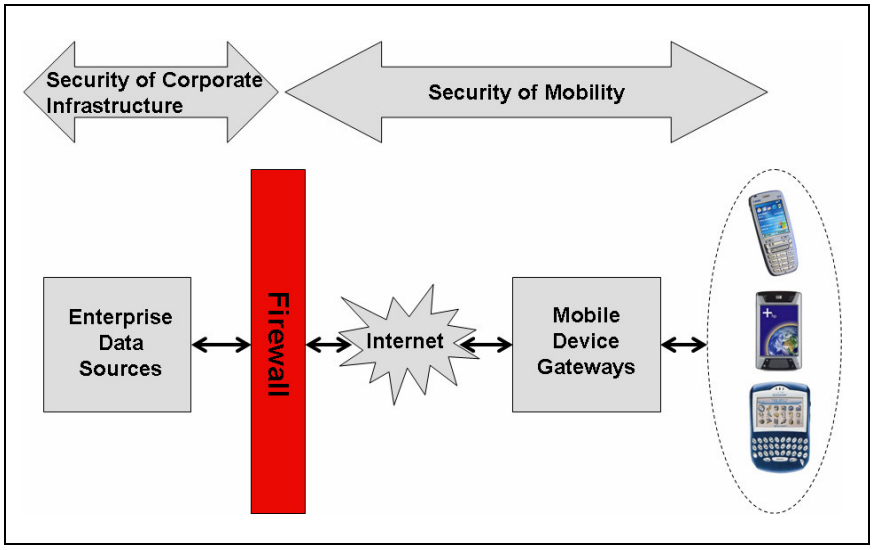

Figure 3.3: Mobility Security Zone

3.4.5. Abstraction

In the modern days heterogeneous systems, an abstraction layer is needed on the top of the security architecture to provide homogeneous and harmonized access to various security functionalities irrespective of the underlying technologies. We have proposed the idea of virtualization of security services to handle the complex problem of security in the large scale open heterogeneous distributed systems and applications. The virtualization of a service definition encompasses the security requirements for accessing that service. The need arises in the virtualization of security semantics to use standardized ways of segmenting security components (e.g., authentication, access control, etc.) and to provide standardized ways of enabling the federation of multiple security mechanisms. This concept is discussed in details in chapter 5.

3.4.6. Scalability

Scalable security is a vital issue for large-scale wide-area systems. There are several issues to be solved for scalable security architecture such as mapping from global subjects to local subjects, centralized certificate authority center, large number of users, many heterogeneous security policies. Generally proxies are used to handle the problem of scalability; however, the proxies are themselves prone to some security risks as there is a private key (for the proxy) that is on a remote system outside the user's direct control, but can be used to sign messages that the overall infrastructure will trust as if coming from him. If a remote system was compromised (or spoofed), the user proxy's private key would no longer be private. Even if he becomes aware of this he would not be able to revoke the proxy because he doesn't have a revocation process. In the absence of some rigorous mechanism of revocation of proxies, the security risks are minimized by making proxies short-lived, so if compromised they cannot be misused for long. Nevertheless, this represents another way in which authentication may be temporarily unreliable.

3.4.7. Resilience

There is a need of self-healing security mechanisms to assure the survivability of the overall system. Resilient security architecture makes the system to regain its original security configurations after the attack scenario is over and therefore it improves the quality of service of the entire system. Current research focuses on system research, development, and use-case expansion to adaptive, real-time, and resilient security systems enabled by new

technologies, services, and methods targeted at improving the survivability and trustworthiness of the IT infrastructure at high operational capacity.

3.5. Security Policy

Before investigating the specifics of a security architecture, it is important to identify the security objectives, the participating entities, and the underlying assumptions. In short, we must define a *security policy*, a set rules that define the security subjects (e.g., users), security objects (e.g., resources) and relationships among them. While many different security policies are possible, we present a specific policy that addresses the security requirements of large-scale systems. A set of common terminologies that is used in the policy description is given below:

- A *subject* is a participant in a security operation. A subject is generally a user, a process operating on behalf of a user, a resource (such as a computer or a file), or a process acting on behalf of a resource.
- A *credential* is a piece of information that is used to prove the identity of a subject. Passwords and certificates are examples of credentials.
- *Authentication* is the process by which a subject proves its identity to a requestor, typically through the use of a credential. Authentication in which both parties (i.e., the requestor and the requestee) authenticate themselves to one another simultaneously is referred to as *mutual authentication*.
- An *object* is a resource that is being protected by the security policy.
- *Authorization* is the process by which we determine whether a subject is allowed to access or use an object.
- A *trust domain* is a logical, administrative structure within which a single, consistent local security policy holds. Put another way, a trust domain is a collection of both subjects and objects governed by single administration and a single security policy.

An example security policy for the Grid is defined in [28]. The key features of this policy are quoted here:

1. The grid environment consists of multiple *trust domains*.

   Remark: This policy element states that the grid security policy must integrate a heterogeneous collection of locally administered users and resources. In general, the grid environment will have limited or no influence over local security policy. Thus, we can neither require that local solutions be replaced, nor are we allowed to override local policy decisions. Consequently, the grid security policy must focus on controlling the inter-domain interactions and the mapping of inter-domain operations into local security policy.

2. Operations that are confined to a single trust domain are subject to local security policy only.

   Remark: No additional security operations or services are imposed on local operations by the grid security policy. The local security policy can be implemented by a variety of methods, including firewalls, Kerberos, and SSH.

3. Both global and local subjects exist. For each trust domain, there exists a partial mapping from global to local subjects.

   Remark: In effect, each user of a resource will have two names, a global name and a potentially different local name on each resource. The mapping of a global name to a local name is site-specific. For example, a site might map global user names to: a

predefined local name, a dynamically allocated local name, or a single \group" name. The existence of the global subject enables the policy to provide single sign-on.

4. Operations between entities located in different trust domains require mutual authentication.

5. An authenticated global subject mapped into a local subject is assumed to be equivalent to being locally authenticated as that local subject.

   Remark: In other words, within a trust domain, the combination of the grid authentication policy and the local mapping meets the security objective of the host domain.

6. All access control decisions are made locally on the basis of the local subject.

   Remark: This policy element requires that access control decisions remain in the hands of the local system administrators.

7. A program or process is allowed to act on behalf of a user and be delegated a subset of the user's rights.

   Remark: This policy element is necessary to support the execution of long-lived programs that may acquire resources dynamically without additional user interaction. It is also needed to support the creation of processes by other processes.

8. Processes running on behalf of the same subject within the same trust domain may share a single set of credentials.

   Remark: Grid computations may involve hundreds of processes on a single resource. This policy component enables scalability of the security architecture to large-scale parallel applications, by avoiding the need to create a unique credential for each process.

3.6. Security Models

Security models are often regarded as a formal presentation of the security policy enforced by the system and are used to test a policy for completeness and consistency. They describe what mechanisms are necessary to implement a security policy and deal with the fundamental security functionalities of a particular system or application. However, they are yet to be fully developed and assessed for their effectiveness and feasibility to support open large-scale IT architectures. A range of these models for different systems are discussed in detail in chapter 4.

Chapter 4

State-of-the-Art Security Mechanisms in Existing Systems

4.1. Grid Computing

The vision of the computational Grid [10] is to provide high performance computing and data infrastructure supporting flexible, secure and coordinated resource sharing among dynamic collections of individuals and institutions known as *virtual organizations (VO)* [45]. Grid computing is rapidly emerging from the scientific and academic area to the industrial and commercial world. It is intended to offer seamless and uniform access to substantial resources without having to consider their geographical locations. Resources can be high performance supercomputers, massive storage space, sensors, satellites, software applications, and data belonging to different institutions and connected through the Internet. Grids can enable collaboration between several organizations. The Grid provides the infrastructure that enables dispersed institutions (commercial companies, universities, government institutions, and laboratories) to form virtual organizations (VOs) that share resources and collaborate for the sake of solving common problems.

4.1.1. Introduction to Grid Security Problems

Grid applications are characterized by the coordinated use of resources from different administrative domains. Figure 4.1 depicts a Grid environment. Each site in the VO is independently administered and has its own local security solutions such as Kerberos and PKI. These solutions are built on top of different platforms such as UNIX, Windows and OS2.

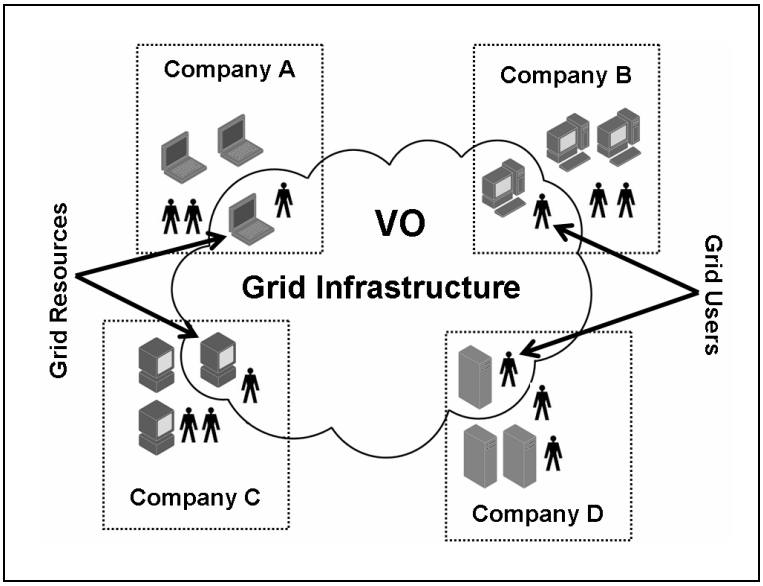

Figure 4.1: A Grid Computing Scenario

When these companies/institutions are brought together to collaborate on a common project in this heterogeneous environment, many security problems arise:

Interoperability

It is a key issue on the Grid. It is impractical to change the security mechanisms at each site in the VO due to technical, financial and political reasons. Thus, the security of the Grid project must be able to interoperate with the local security solutions at different levels:

Policy level

Each partner in the VO has its own security policy, which is carefully tailored to maximize the protection of its valuable resources. The main issues to be addressed are:

- Reconciliation of global security policy with local security policy.
- Solution of conflicts between local and global policy.

Authentication level

VO sites require mechanisms for identifying users from one security domain to another. For example, the identity of a user from company **A** (**U.A**) and his credential as expressed in Policy **A** are meaningless in the other VO sites. Therefore, how does **U.A** authenticate (e.g. UNIX login) to site **B** to access resource (**R.B**) (e.g. Kerberos)?

Authorization level

Access control mechanisms used vary from one VO site to another depending on the type and value of the resource accommodated. For example, site **A** may use an Access Control List (ACL) or a Role Based Access Control (RBAC) as mechanisms in order to gain access to its resources. The first problem is how to determine whether a user, **U.A,** authenticated in site **B,** is allowed access to resource, **R.B** in **B**. The second is who decides what the access rights of **U.A** are?

Scalability

The number of users and resources in the VO is dynamic. New users/resources can be added/removed to the project as required. A scalable way to dynamically manage users' authentication and their access rights to access project resources is required.

Confidentiality and integrity issues

On the Grid, users transmit data over the Internet and access remote data resources that may be very sensitive. Moreover, Grid users can run programs on remote sites. Therefore, confidentiality and integrity are required to:

- Protect transmitted data over a public network such as the Internet
- Ensure the privacy and accuracy of the results of programs executed on remote sites.
- Ensure the secrecy and correctness of the shared data resources.

Trust

Scientists and commercial companies want to know whom they are trusting with their data and commodities. The question that arises: Who to trust individuals/sites/third parties?

Usability

Grid users are from different types of organizations such as academic, government and financial institutions. Thus, they may not be security experts. Therefore, usability is required so that access to the VO resources is as smooth and seamless as access to local resources.

Firewall

A frequently encountered problem on the Grid is firewalls. VO members want to share resources with other partners but also, want to keep their other resources private. Collaborating partners on the Grid have to allow requests from and replies to jobs initiated from other sites to pass through their firewall to access their resources. This requires opening a port in the firewall to access those resources, which could introduce another vulnerability to the local security of the VO partner's organization. For commercial companies, it is unthinkable to compromise local security so they may end up without collaboration!

4.1.2. Grid Security – State-of-the-Art

In this section, prominent grid projects are examined with special considerations of their security mechanisms.

Globus

Globus [46] is the best known and probably the most widely-used end-to-end grid infrastructure available today. The philosophy of Globus is to enable sharing of computational resources across sites that have a relatively high level of trust in each other. To this end, the default security model in Globus provides rather large privileges to remote users, and depends heavily on authentication ('the only authenticated user is a good user'). The Toolkit, developed by the Globus Project, offers authentication, authorization, and secure communications through its Grid Security Infrastructure (GSI) [28]. The GSI uses public key cryptography, specifically public/private keys and X.509 certificates, as the basis for creating secure grids.

A time-stamped proxy, based on the user's private key is created in GSI for a secure authentication. Users cannot submit jobs to run or transfer data without creating the proxy. This proxy is used to grant or deny access to the grid resources. The user authorization in GSI is handled by mapping the user to a local user on the system being accessed. The system receiving the request reads the user's name from the proxy, and then accesses a local file to map that name to a local user. System administrators can assign users to virtual groups to avoid creating scores of extra user IDs on various grid systems. All users from a particular domain can be mapped to a single, common user ID when accessing a given grid resource. GSI is designed this way to help administrators separate outside users running grid computations from local users in need of local administration and support.

GSI uses digital certificates for mutual authentication and SSL/TLS for data encryption. The Toolkit contains OpenSSL, which it uses to create an encrypted tunnel between grid clients and servers, whereas GSI-Enabled OpenSSH is suggested for secure remote access to the grid. Secure Shell (SSH) establishes an encrypted session between the user's client and the grid server.

UNiform Interface to COmputing REsources (UNICORE)

The goal of UNICORE [47] is to deliver software that allows users to submit jobs to remote high performance computing resources without having to learn details of the target operating system, data storage conventions and techniques, or administrative policies and procedures at the target site. Fujitsu originally developed this, with contributions from commercial partners and the European academic and government research community.

UNICORE provides access to heterogeneous resources at remote sites through internet. This allows switching between the systems without changing the job. Another function is to perform synchronization and switching without any user intervention. UNICORE implements the same architecture as that of the web. In this way, UNICORE brings the power of supercomputing and the data resources involved, made available through World Wide Web (WWW).

The major concern while accessing resources via the web is security. UNICORE users and servers are authenticated by means of certificate (X509 compliant) issued by the UNICORE Certification Authority (CA). It also uses Secure Socket Layer (SSL) to provide network integrity for all control mechanisms and optionally for confidential data transfer.

Overall security within UNICORE heavily depends on:

- the security within the UNICORE PKI (CA security and RA authentication policy),
- the security of the private key-stores within the user clients and servers, and
- the diligence with which the individual certificates and certificate chains are validated before trust is granted.

The current PKI model is based upon a single central CA which signs the certificates of all UNICORE users. This model is good for a limited number of users. As soon as the number of users increases, the load for the CA steps up too. A higher CA load means:

- increasing delays in issuing certificates
- increasing number of RAs which results in a higher administrative load and possible security problems due to more frequent RA status changes (new RAs, diminishing RAs, changing RA representatives, etc.)
- in case the CA certificate expires or gets compromised (stolen private key) all subordinated certificates have to be exchanged against new ones. This would cause a total freeze of the whole UNICORE sphere. This would be a knock-out criterion for commercial, high availability applications.
- a single CA leaves no space for redundancy (no backup certificates from a separate CA).

In a distributed environment normally only partial outages occur. For commercial and/or very important applications there could be backup certificates from a different CA, so that those jobs could be re-submitted immediately.

Secure Highly Available Resource Peering (SHARP)

SHARP [48] tries to define new ways to share grid resources and delegate authority for using those resources. It proposes a new type of grid security infrastructure, a *policy server*, which controls when, where, and to what extent users can access grid resources. These policy servers issue a ticket to users that prove to a resource owner that this particular authorized policy server has granted access.

One of the key features of SHARP is its method for making secure sharing possible without creating a central authority to manage resource requests. Valid principals within a SHARP grid obtain *claims* to control a share of grid resources; varied principals can exchange claims in the same manner that Internet Service Providers (ISPs) do in exchanging network bandwidth for routing. Within the SHARP model, each site acts as a central authority to certify keys, validate signatures, and detect conflicts for claims on its local resources. Claims are cryptographically signed to make them unforgeable, nonrepudiable, and independently verifiable by third parties. Once established, the claims are managed by *agents*, pluggable modules which subdivide the claims and allocate them to their clients. These agents are designed to make the resource claim process more efficient. To avoid tying up excess system resources, these claims are timed, and expire after a specified period, so the system can recover the resource if the claim holder doesn't exercise their option. In some situations, agents may oversubscribe resources with extra claims, a method which makes sure that the resource pool is fully used even with some claims failing to materialize or timing out [49].

Condor and Condor-G Systems

Condor [50] is a software system that creates a High-Throughput Computing (HTC) environment by harnessing the power of clusters and workstations. It can manage dedicated clusters. However its main appeal is that it can make use of pre-existing resources which may be computers sitting on people's desks. When jobs are submitted to Condor, it finds an available machine in its organization's pool to run the job. Machines become available once they have been idle for a specified period of time. Jobs are migrated over the network to the machine. If the machine becomes unavailable and the job has not finished, Condor checkpoints it and either migrates the job to another machine or queues it to disk until a machine becomes free.

Whilst Condor can be treated as a resource provider at the lowest level of the grid, its overall architecture fits into the component layers of the grid model. There are however some fundamental differences between Condor and the grid. As a project it began development before the concept of the grid came into existence. Therefore some parts of the system (e.g. the communication system), use older technologies (RPC: Remote Procedure Call) and Condor uses its own proprietary systems for resource description, discovery and integration.

The inter-domain resource management protocols of the Globus Toolkit and the intra-domain resource management methods of Condor are combined in Condor-G System. Condor-G gets its name from how it talks to the resource management part. Condor-G uses the Globus Toolkit to start the job on a remote machine instead of using the Condor-developed protocols to start running a job on the remote machine. Condor-G provides a *window to the grid* for users to both access resources and manages jobs running on remote resources. In other words, Condor-G allows the user to harness multi-domain resources as if they all belong to one personal domain [51].

Condor-G incorporates GSI to answer its security needs. We have already discussed the GSI in the preceding Globus section.

Legion

Legion [52] is an object-based grid system developed at the University of Virginia. Its architecture was designed to address the challenges of using and managing wide-area resources. The Legion system is an implementation of a software architecture for grid computing. The basic philosophy underlying this architecture is the presentation of all grid resources as components of a single, seamless, virtual machine.

Legion programs and objects run on top of host operating systems, in user space. They are thus subject to the policies and administrative control of the local Operating System. The Legion objects running on a particular host must trust that host. This trust does not necessarily extend to objects running elsewhere, however. A critical aspect of Legion security is that the security of the overall Legion system cannot rely on every host being trustworthy. A large Legion system will span multiple trust domains, and even within one trust domain, some of the hosts may be compromised or may even be malicious.

There are two main types of credentials in Legion: *delegated credentials*, and *bearer credentials*. A delegated credential specifies exactly who is granted the listed rights, whereas simple possession of a bearer credential grants the rights listed within it. A Legion credential specifies the period the credentials are valid, who is allowed to use the credential, and the rights. The credential also includes the identity of its maker, who digitally signs the complete credential.

The Globus and Legion share a common base of target environments, technical objectives, and target end users, as well as a number of similar design features. Both systems abstract access to processing resources: Legion via the host object interface; Globus through the Globus Resource Allocation Manager (GRAM) interface. Both systems also support applications developed using a range of programming models, including popular packages such as Message Passing Interface (MPI). Despite these similarities, the systems differ significantly in their basic architectural techniques and design principles. Whereas Legion builds higher-level system functionality on top of a single unified object model, the

Globus implementation is based on the combination of working components into a composite meta computing toolkit for low-level services.

4.1.3. Open Grid Services Architecture (OGSA)

During the past years, Grid Computing and Web Services have started to merge and to benefit from the synergy of both paradigms. The Global Grid Forum (GGF) presented Open Grid Services Architecture (OGSA) as the fusion between Grid Computing and Web Services. Moreover, Grid Services require security mechanisms. The OGSA Security Architecture identifies the security requirements in a Grid environment, and based on them the Web services security model, defines a security model to secure Grid services.

Grids, as any computing environment, require some degree of system management, and especially security management. It is a potentially complex task given that resources are often heterogeneous, distributed, and cross multiple management domains. Currently, the Common Management Model Work Group (CMM-WG) is working on the specification of a management framework for OGSA. CMM-WG points to Common Information Model (CIM) as an interesting model for the management of the security services, but it does not include any further work in this line.

OGSA Security MODEL

The Security Model for OGSA is formed by the set of security components shown in figure 4.2.

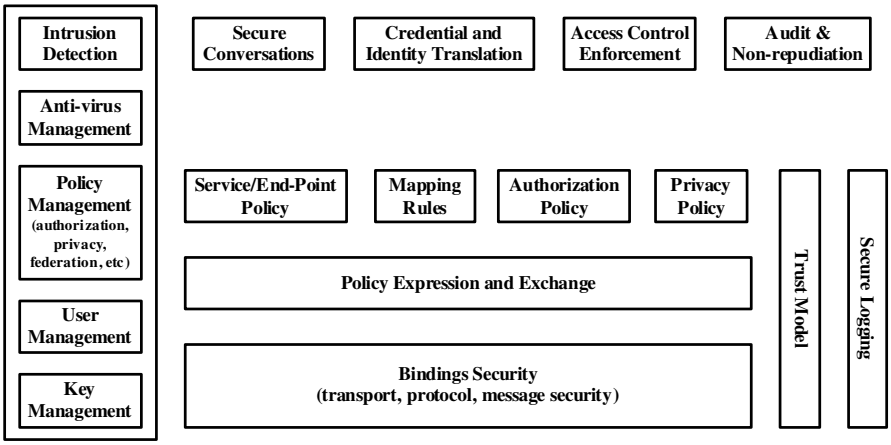

Figure 4.2: Components of the OGSA Security Model

In this layering, top components such as secure conversation, credential and identity translation, access control enforcement, and audit and non-repudiation, are application-specific components that depend on rules and policies for authorization, privacy, identity/credential mapping, and service/end-point provision. These grid policies are specified and defined based on a language for policy expression and exchange. In the bottom layer, the security of the bindings is based on the security characteristics of the used transport protocol and message format. On the right-hand side, the trust model component defines and establishes trust relationships for the grid environment, i.e. defining VO membership. The secure logging component is a requirement for the auditing of any policy decision. Finally, the left box groups all security management functions such as key management for cryptographic functions, user registry management, authorization, privacy and trust policy management and management of mapping rules. It also includes the management of anti-virus and intrusion detection services.

The OGSA security model is a framework that is extensible and flexible enough to allow the use of existing security technologies and standards, such as IPsec or SSL/TLS in the case of the network and transport layer, HTTPS in the binding layer, or security standards based on use of XML and assertion languages (e.g., SAML) in the message-level. Therefore, given that OGSA is a service-oriented architecture based on Web services (i.e. WSDL-based service definitions), the OGSA security model needs to be consistent with Web services security model that is currently being defined for the Web services framework. Figure 4.3 shows the set of Web services security specifications.

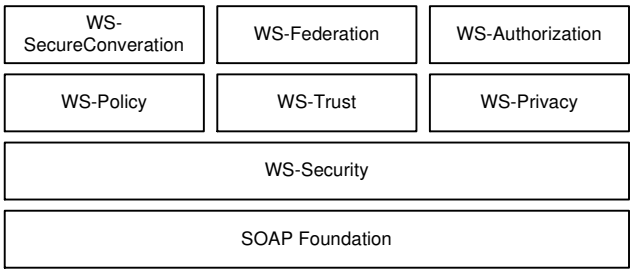

Figure 4.3: Web Services Security Specifications

In the bottom layer, several technologies such as SOAP, WSDL, XML Digital Signatures, XML Encryption and SSL/TLS are the core of the Web services model. Built upon this, the message security model (WS-Security) provides the base for the other security specifications, which include a Web service endpoint policy (WS-Policy), a trust model (WS-Trust), a privacy model (WS-Privacy), a model for secure conversations (WS-SecureConversation), a federated trust model (WS-Federation), and an authorization model (WS-Authorization).

These specifications serve as building blocks for the OGSA security specifications and can be used to implement the OGSA security model.

Shortcomings of OGSA

The OGSA is a novel initiative and there is a long way to go before we have a complete architecture specification where all of the desired properties of Grids are addressed. This can only happen by having reference implementations and deployments of OGSA-compliant Grid middleware that will eventually expose the strengths and weaknesses of the architecture [53]. Some shortcomings of the OGSA model are summarized below:

Availability and Robustness

The mechanisms of OGSA might greatly improve on the availability of services by introducing the Factory pattern but there needs to be further discussion of services that deal with failing or unavailable instances and start up new ones automatically. By introducing Factories OGSA lays the ground for automated service startup and thus increases robustness and availability. However, it would greatly enhance this aspect of OGSA if some service data elements were defined in the Factory that would deal with failing instances and policies on how to restart them. More discussion is needed on how the Grid Services should behave if some kind of failure occurs. What happens if a service has been unavailable for a given time? How to deal with service overload? What if the network of a Virtual Organization (VO) becomes partitioned? What happens if the Factory or the Registry is suddenly unavailable? This also touches a bit on the desired property of scalability.

Measurability

In the OGSA model, each VO needs certain levels of QoS to be achieved and that they may be measured in many different ways. The ability to set up VOs fulfilling many different QoS requirements is highlighted as one of the most desirable properties of Grids. OGSA does not elaborate further on QoS metrics. There need to be not just agreed metrics of QoS but each Grid Service needs to define how it enhances or decreases certain QoS metrics. There might be the need to define a QoS namespace to be able to query this property of services more easily. Each Service also needs to declare its own internal QoS metrics and give a value in a specific instance if different instances of the same service can be set up such that the given metrics can change. Measurability is also very important when a VO defines its own set of necessary services or when it analyzes its own state and optimizes its performance due to changes in its nature or requirements. In order to define, bootstrap and optimize a VO it is essential to have QoS metrics by which the VO can measure itself.

Integration

OGSA starts out with this point on integration of services. There is a need to integrate services not just within but also across VOs. OGSA solves this problem by defining the Grid Service interface and requiring all services to implement it. But how this is achieved with OGSA in detail and especially across VO boundaries is not detailed. A lot of effort still needs to be put into the exact mechanisms and definition of common semantics that integration of services (across VOs) may be achieved.

Interoperability and Compatibility

Interoperability is explicitly mentioned as a requirement and it is one of the driving concepts behind OGSA. Web Services, including Grid Services, are designed such that the modules are highly interoperable. There is no uniform protocol required that each service has to speak. Web Services Resource Framework (WSRF) descriptions are there to ensure interoperability. Interoperability is very closely related to discovery because services that need to interoperate have to discover among other things which common protocols they can use and whether there are issues of compatibility. OGSA addresses compatibility by having Service Description elements that declare compatibility to past versions. What is missing is the capability to declare forward compatibility, not just backward (in)compatibility.

Service Discovery

Users of many services and services that want to interoperate need to get hold of the service descriptions to discover which services meet their needs or which services are still missing to achieve a given QoS. But how does the user know how to get hold of these descriptions? The answer of OGSA is the Registry and the HandleMap. The Registry needs to be searched to find the Grid Service Handles of the services that fulfill the user requirements – formulated in a query if necessary. The HandleMap then can be contacted to retrieve the detailed description of the services in question. But more fundamentally, this touches again on unified QoS metrics and Service Data elements. How is it possible to get the handles of the registries that we can query to get a set of services that we might be interested in i.e. how do we find the registry or registries relevant for a given query? How is a query formulated to do so? OGSA briefly mentions the use of XQuery [54] to query the Service Data of the Registry. In a large Grid spanning administrative domains, service data is partitioned over one or more registries nodes, for reasons including autonomy, scalability, availability, performance and security. This needs to be clarified in the future.

Another interesting design decision of OGSA is that the Grid Service Handle is strongly coupled to the home HandleMap. This has several implications. The home HandleMap

needs to point to a valid URL at all times. If the service is moved then all GSH will have to deal with the update latency in the DNS that might also impact on service discovery (mapping of GSH into GSR). It is amazing why the GSH is a restricted URL and may not have extensions (i.e. point to a server page or have predefined HTTP GET options).

Manageability

Manageability is also just touched upon as a desired element, but is deferred to a later time. The idea of unified monitoring and controlling mechanisms is there but not further exploited. Higher-level services are supposed to deal with this, but how are they to interact and be integrated with each other? Having uniform monitoring and control mechanisms might impose stricter requirements on the architecture so this is worth giving more thought. The questions one might ask include:

- How can a VO be created and bootstrapped?
- How can it be changed?
- How can individual users monitor and control their Grid sessions?

Security

This issue is touched but not elaborated on sufficiently. The hosting environment gets the burden of authentication, which is reasonable; but there is no discussion on how local and VO-wide security policies are enforced also on authentication. Is there the need for a Grid Service that deals with these issues or should each of the services have an interface addressing this, making it part of the GridService base interface? New developments in this area are necessary.

For Data Grids, authorization and accounting will be particularly complex for certain VOs. In this context, global authorization schemes don't work because local resources refuse to trust the global authorization authority to perform the authorization in accordance with the local policies – which may change over time. Also for logging, auditing and accounting purposes the local resource managers will always want to know exactly who has done what to their resource. An issue is how to delegate rights to automated Grid services that need to use the resources on behalf of the user even if the user did not initiate their usage explicitly.

Security has to be dealt with within OGSA since it will depend on the success of the underlying security framework. Open questions include: How are VO-wide policies applied? How is local security policies enforced? What is the role of hosting environments? How is an audit performed? Can a user belong to more than one VO and use both resources even if the security mechanisms differ?

Support for the Mobile Devices

There has been no considerable contribution on actual approaches to accommodate mobile and wireless smart devices such as PDAs and smart phones in the computational Grids. Mobile electronic devices and wearable computers are increasingly common. Individuals will frequently own a collection of these mobile devices. Yet, these devices are often resource limited: processing power is low, battery life is finite, and storage space is constrained. These restrictions slow application execution, and hinder operability. Arguably, applications executing on devices must be made aware of concurrently-executing applications in order to optimally use the limited resources.

Currently, OGSI implementations exist for several platforms, or runtimes. Sandholm et al. implement OGSI for the Java Virtual Machine runtime [55]. Humphrey et al. implement OGSI for the Microsoft .NET Framework runtime [56]. However, very few mobile devices can support either of these runtimes. Rather, many mobile devices run Windows CE with the .NET Compact Framework, a substantially stripped-down version of the .NET

Framework. In addition, neither of these implementations considers the addition of mobile device constraints, such as limited resources and intermittent network connectivity.

Several efforts combine grid computing and mobile devices. Gonzalez-Castano incorporates mobile devices into Condor as client front-ends for job submission and job querying to traditional supercomputer grids [57]. Phan et al. suggest proxy based cluster architecture for introducing mobile devices into traditional grids [58], though provides no implementation for evaluation. Clarke and Humphrey investigate the challenges of integrating mobile devices into the Legion grid computing system [59]. While addressing some of the particular concerns of mobile devices, none of these efforts embraces the community-adopted OGSI specification.

The characteristics of wireless and mobile devices must be considered if these devices are to be integrated in the Grid world. The system must have the ability to operate on power-, memory-, and even bandwidth-constrained hardware. The software must be sensible to resource consumption in these environments. Moreover, the sporadic and dynamic network environment must be handled gracefully. These concerns imply several capabilities that are useful in a Grid system, such as a Grid system should be both flexible and reflective, allowing users to make tradeoffs and select the combination of services that is best suited for their purpose. The ability to dynamically query and adjust, at runtime, the implementation and execution characteristics of the system software is important when dealing with these devices.

4.2. Cluster Computing

Cluster is a collection of interconnected computers working together as a single system. The initial idea leading to cluster computing was developed in the 1960s by IBM as a way of linking large mainframes to provide a cost-effective form of commercial parallelism. However, cluster computing did not gain momentum until three trends converged in the 1980s: high performance microprocessors, high-speed networks, and standard tools for high performance distributed computing. A possible fourth trend is the increased need of computing power for computational science and commercial applications coupled with the high cost and low accessibility of traditional supercomputers. These building blocks are also known as killer-microprocessors, killer-networks, killer-tools, and killer-applications, respectively. The recent advances in these technologies and their availability as cheap and commodity components are making clusters or networks of computers (PCs, workstations, and SMPs) an appealing vehicle for cost-effective parallel computing.

4.2.1. Introduction to Clusters Security

Cluster systems are finding increasing deployment in academic, research, and commercial settings. Over the last several years, the trend has been towards an increase in both the absolute number of cluster installations and in the average number of nodes per cluster. The increase in the average sizes of clusters introduces a new set of challenges for system administrators. While a great deal of effort has been expended in creating tools to aid in the installation, administration, and monitoring of clusters, very little effort has been expended in creating tools that address the unique issues of cluster security, particularly for very large cluster installations.

When commodity clusters were still a new technology, most of the development focus was centered on simply getting them to work; the issue of cluster security was given relatively little consideration for at least two reasons. First, many people thought it was unlikely that hackers would disrupt scientific systems and jobs. Second, many people believed that the issues related to cluster security were the same as for general computer security. (“What works for one system should work for a collection of 100 systems”) [81]. However, as cluster systems have become more widespread and powerful, they have become increasingly desirable targets to attackers due to a few functional characteristics:

1. *High bandwidth connections* – To facilitate its computational goals, a cluster must have high bandwidth connections to the outside world, allowing interactive use by many users, transfer of large datasets into and out of the cluster, and fast inter-node communication. These high bandwidth connections are attractive to attackers because the attacker can subsequently leverage them for purposes such as launching denial-of-service flood attacks against other sites.
2. *Extensive computational power* – Legitimate cluster users marshal the aggregate processing power of multiple machines with the goal of solving grand challenge scientific problems. In contrast, this computational power could be used by an attacker for purposes such as carrying out brute-force attacks against authentication mechanisms on other network resources to which the attacker wishes to gain unauthorized access.
3. *Massive storage capacity* – Many high-performance cluster environments include storage capacity measured in terabytes, used for storing large scientific datasets and the results produced by computations involving these datasets. To a hacker, large-capacity disk storage is an attractive target for use in creating repositories of stolen copyrighted software and multimedia files.

The issues related to cluster security are not the same as those related to general computer security. Even though the behavior of individual nodes may be simple and could be approached with traditional computer security techniques, effective security management in the context of cluster systems requires tools that evaluate the state of the cluster as a whole. ("A 100-node cluster is different from 100 standalone systems"). For example, a traditional security monitoring tool, that examines the flow of communication into and out of individual cluster nodes, is limited to evaluating security based only on streams of data that it considers independently of any cluster-specific context. On the other hand, a cluster-aware security monitoring tool could evaluate whether a given node should even be communicating at all, based on information from sources such as the cluster's job management system. That is, if no job is currently scheduled for execution on a given node, that node should most likely not be sending or receiving data on the network.

The idea that cluster security must be considered as a whole is further underscored by realizing that while the behavior of individual cluster components may be simple, the combined interactions of multiple components may result in complex, unintended, and non-intuitive behaviors that are difficult or impossible to predict. That is, even if certain hardware or software components that make up a cluster are certified as *assured*, these components must co-exist in a cluster environment that most likely consists of *non-assured* components. Furthermore, even if a cluster was built entirely from certified components, it is unlikely that the entire cluster, considered as a single entity, would have been evaluated in any kind of certification process. Simple combinatorics makes it infeasible to use formal methods to identify and protect against all known vulnerabilities from component interactions. For example, attack trees [82] are a good technique for prioritizing risks from known attacks. However, despite work that generates attack trees for limited, and some would say artificial, scenarios, attack trees have been shown not to scale to practical environments where new attacks cannot be modeled in advance and where the scale of components and their interactions are intractable for realistic computation [83].

4.2.2. Cluster Security – State-of-the-Art

This section provides a list of presently used techniques used to guard high-performance clusters.

Network Considerations

To reduce the risk of unauthorized access, a site can adopt an enclosed cluster design. In extreme cases, this can be achieved by keeping a cluster on a physically isolated network. A more common and convenient approach is to limit direct user access to dedicated login or

head nodes. The compute nodes can then be placed in a private non-routable address space, or alternately kept behind a firewall. In situations where it is feasible, this approach limits the scope of outside threats, and correspondingly lessens the work of administrators. A Grid computing environment can present problems with this type of enclosed cluster, however, as Grid jobs can be allocated nodes on multiple clusters, all of which may have to intercommunicate. A more open design, with all nodes Internet accessible, is necessary to support this functionality.

To prevent a potentially compromised machine from sniffing cluster communications, no unmanaged machines or machines with different security models should be allowed on the same network segments as any cluster computers.

Centralized Software Configuration

A tightly-constrained software environment on clusters is important for both performance and security. Only specific software should be installed on clusters and permitted programs must be current and patched. Recognizing the distinct types of cluster nodes as equivalence classes with regards to their configuration can ease administration and bolster cluster security. By restricting the available software on a given node type, fewer computers may be affected by the update of a given package. Moreover, certain classes of cluster nodes may receive higher priority based on the security impact of a compromise on them. Central configuration management can be implemented by either network mounting common files or by utilizing a mechanism for automatic distribution of such files to various subsets of the cluster as needed. Tools such as Cfengine [84], which exist for the purpose of centralized configuration management and repair in a general network setting, have been adapted for use in a cluster environment.

Authentication

Authentication on cluster machines is another area of security concern. Traditional means of authentication, like /etc/passwd and shadow files, present some configuration issues in any distributed system. The number of machines in a large cluster can create a problem with synchronizing these files in a timely manner. Therefore, when new users are added, or someone has a password change, all machines need to be updated.

Public key mechanisms such as RSA authentication using SSH provide another means of security. Here the user manages their own keys which are kept in their home directories. Public key systems such as these, while cryptographically secure, rely on the users protecting their private key, and adding additional protection with the use of a passphrase on their key. Many users will forego this last step, instead preferring the ease of a passwordless login.

Centralized authentication methods like Kerberos are typically used in cluster environments. Using a service like Kerberos users can authenticate once and then have access to any cluster resource they are authorized to use. Kerberos and related systems also provide better protection of user authenticators and can enforce varying policies on passwords for users (length, character classes, expiration, etc.).

PKI systems provide another means for maintaining cluster security. It is becoming more common in cluster environments for users to authenticate with something like X.509 certificates to authenticate to services. One of the current drawbacks of PKI is that you are placing the responsibility on the users to protect their keys, and users may not be very security conscious.

It is always possible for an intruder to masquerade as an authorized user. This can be achieved by exploiting protocol flaws, or by local keyboard sniffing for passwords. Thus root access to a cluster should demand a higher standard of security. Under no circumstances should remote root logins be permitted, only direct console access.

Intrusion Detection Systems

Host-Based Intrusion Detection Systems (HIDS) such as Tripwire [85] are commonly used to monitor high-value assets such as clusters. Tripwire is typically configured to report file and operating systems changes once every 24 hours. While Tripwire is reliable, it has usability problems due to the cryptic nature of its reports as well as false positives. In the context of clusters, Tripwire makes no priority distinctions between different nodes, so that security staff has a difficult time obtaining situational awareness of file/operating system changes when considering a cluster as one system. Since Tripwire reports all file/operating system changes, many of the alerts it generates are actually legitimate user or system administration activity. Faced with a large volume of false positives, a cluster administrator making changes across a large number of nodes will often disable an HIDS for significant periods of time.

While an HIDS is capable of detect signs of intrusion, ultimately their reports must be validated since it is possible (and likely) that upon a successful root-level compromise, an intruder will replace the binaries used by that system. Network-based Intrusion Detection Systems (NIDS) can be used to verify the output from individual hosts, in addition to scanning for generally suspicious traffic. NIDS passively monitor network flows, and can be configured to send alerts if traffic matching attack signatures are detected. Neither HIDS or NIDS have been adapted for the unique cluster environment.

Packet Filtering

It is possible to individually firewall each host to specifically tailor cluster node access. Pfilter [86] compiles security policies into either iptables or ipchains rule sets for Linux. While the advantages in using an automated tool like Pfilter in larger cluster environments may be clear, cluster administrators are often reluctant to firewall cluster nodes aggressively due to concerns of either performance or user inconvenience. In a Grid-enabled cluster, or a cluster where the policy is to allow users relatively free reign in the used of allocated nodes, firewalling individual nodes may be unacceptable without some provision for dynamically adjusting firewall rules on a per-host basis.

4.2.3. Cluster Security – Open Challenges

A cluster encompasses a collection of distributed resources: multiple layers including applications, middleware, operating systems, and network interconnects must all be coherently protected. While locking down a cluster by disabling services is desirable from a security perspective, cluster resources are meant to be used, so there is the resource management challenge of allowing users to consume resources in an authorized way.

Clusters represent a heterogeneous management environment composed of different hardware and software node configurations, presenting the challenge of integrating different security solutions (vendor or open source) with a goal of comprehensive security solutions across the entire cluster. Further, there are large scale management requirements. As the size of clusters continues to increase, installing, monitoring, and maintaining clusters becomes a challenge since any misconfiguration or inconsistency potentially becomes an exploitable vulnerability. We are beyond the point where typically-sized clusters can be managed manually without automation support. The current state-of-the-art has automated cluster tools available for performance management; the challenge is developing automated cluster tools for security management.

Distributed Security Infrastructure (DSI)

The DSI project [87] targets the distributed access control service. DSI began as a research project to support different security mechanisms to address the needs of telecommunications applications running on carrier-class Linux clusters. For the time being, DSI provides distributed mechanisms for access control, security management, and authentication.

The Distributed Security Infrastructure contains one security server (SS) and a security manager (SM) on the remaining cluster nodes. The SS is responsible for distributed security management of the cluster. It will propagate the security policy and communicate via alarms and messages with the SMs on the nodes. Communication is done over the Secure Communication Channel (SCC). The SCC communications are encrypted using SSL/TLS over CORBA.

The versatility of DSI is in fine-grained control that can be enforced on the node by the SMs. Various structures in the kernel such as sockets and processes can be assigned a *security context identifier* (ScID). ScIDs are global over the cluster and persistent. ScIDs are meant to group together processes that have the same security context. So, contrary to PIDs, SsIDs do not uniquely identify processes but security contexts. Similarly, each node is assigned a *security node identifier* SnID. Hence, the distributed security policy (DSP) consists of a list of rules to be applied to (SnID, ScID) pairs.

For security mechanisms to be effective, users should not be able to bypass them. Hence, the best place to enforce security is at kernel level. Therefore, when necessary, all security decisions are implemented at kernel level, in the *DSI Security Module* (DSM). DSM is a set of kernel functions enforcing distributed security policy, and is implemented using LSM [88] as a Linux kernel module. As future work, in order to use the mainstream Linux tools, we consider using SELinux instead of our internally developed DSM Linux kernel module.

As presented in Section 3, there is need for compartmentalization in large distributed applications. In order to compartmentalize large applications, DSI uses ScIDs to implement different virtual security zones. These security zones are defined with a process level granularity across the entire cluster. They are based on the process type and the node on which they are executing. A process instance can belong to different security zones. For example, the instances of the same process type can be defined in different security zones depending on which cluster node they are running. ScIDs do not identify different instances of a process type, but rather define the security zone they belong to. The security rules are defined in a central security policy file: Distributed Security Policy (DSP). They define the possible interactions between different security zones in the entire cluster. The DSP file can be used by the administrator to define a homogeneous view of the cluster. This is particularly convenient for the carrier-class clusters which are not running a wide range of applications – this makes it possible to predefine interactions between different zones. This flexible mechanism can be used to confine untrusted software or in an extreme case run them inside a sandbox. DSP changes are automatically propagated to all nodes of the cluster. The security managers are in charge of communicating these new rules to the local DSM providing a dynamic evolution of security behavior of the cluster.

4.3. Peer-to-Peer (P2P) Computing

A peer-to-peer system is defined as a distributed system in which network-addressable computing elements called *peers* have comparable roles and responsibilities, communicate information, share or consume services and resources between them. The ability of peer-to-peer systems to harness vast amounts of storage from a scalable collection of autonomous peers and its emphasis on de-centralization and lack of a central authority have made it an attractive systems solution to everyday home computer users, who seem empowered by the ability to independently select and change their own policies, roles, and responsibilities. By allowing peers to share a portion of the authority, these systems also possess other interesting technical characteristics such as self-organization and adaptation.

Peer-to-peer communities provide a method for arranging large numbers of peers in a self configuring peer relationship based on declared attributes (or interests) of the participating peers. This method is expected to have an impact on sharing of resources, pruning of search spaces, and trust relationships amongst peers in the network.

4.3.1. Introduction to P2P Security Problems

Security is one of most important considerations when architecting, deploying, or integrating P2P applications or products. The very nature of P2P indicates that nodes are organized in a flat structure. This means that there is no one single fixed super-server that is responsible for routing or servicing calls and requests from client applications. In the conventional setting, as is in the current web application model, a variety of security means and tools can be applied on the super-server, such as logging, filtering – both IP and domain level, authentication – username/password, token level, reciprocal, PKI, challenge keys, etc, OS hardening, and frozen image – like Tripwire, etc. However, it will be much more difficult to apply similar measures to a P2P environment, since there is no one single fixed super-server or a guaranteed repository of metadata. Similarly, the difficulty to set up a uniform security policy and/or security standard makes P2P computing even more prone to security issues and hacking. Therefore, P2P security issues must analyzed with utmost care.

First to understand how data is transferred on the internet, we need to look at the SS7 (Signaling System 7) stack and protocol widely used by the telecommunication industry. As shown in figure 4.4, the SS7 separates data communication into 7 layers namely Physical, Data, Network, Transport, Session, Presentation, and Application.

The Physical layer depends on the underlying hardware and network connections like wiring and switches. The Data Link layer talks about packet specifications and issues, such as packet format, header, CRC checks (**C**yclic **R**edundancy **C**ode), error handling, data flow control, etc. The Network layer talks about message routing and control. For majority of P2P application developers and users, what's more interesting are the 4 upper layers which sit on the top of the network layer.

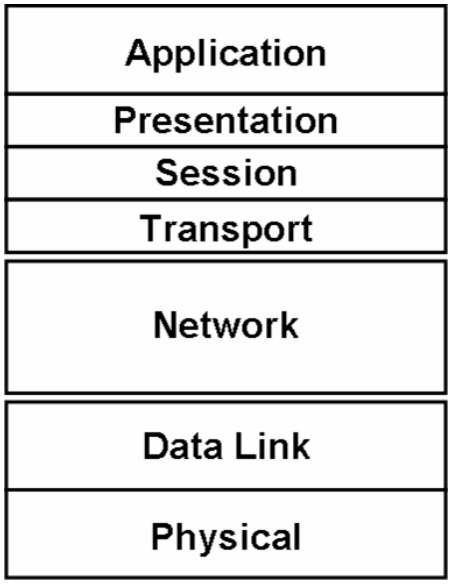

Figure 4.4: SS7 layers architecture

There are a variety of transport protocols available to guarantee the data integrity and validity. For example, there are HTTPS (secure hyper-text transfer protocol), SSL (secure socket layer), TLS (transport layer security), etc. Majority of which uses encryption to protect the data. A decision to use which or to not even use encryption will be up to the P2P application designer. Management of encrypted data requires more resources and consequently, performance penalty could be the counterweight to use encryption.

At the session level, a channel between P2P endpoints must be established, identified, and authenticated to the network. How long to keep a session valid or a connection alive will depend on the nature of the protocol running, the network refresh rate, availability and cost of other system resources, as well as fluctuation in the number of concurrent connections.

At the presentation and application layers, security requirements become more desirable as there are user interactions. An important point to consider for a P2P system is that nodes are transient and not constant meaning that peer and peer-groups can come and go as wish. The storage of the authentication tokens and user-identities is a complex issue, whether they should be stored locally, on the super-peer, broadcasted regularly, or even to be defined by default.

4.3.2. P2P Security – State-of-the-Art

Secure *nodeId* Assignment

In the original design of Pastry [60], and in many other P2P systems [61-63], nodeIds are chosen at random from the space of all identifiers (i.e., for Pastry, a randomly chosen 128-bit number). The problem with such a system is that a node might choose its identifier maliciously. A coalition of malicious nodes that wishes to censor a specific document could easily allocate itself a collection of nodeIds closer to that document's key than any existing nodes in the system. This would allow the coalition to control all the replica roots for that document, giving them the ability to censor the document from the network. Likewise, a coalition could similarly choose nodeIds to maximize its chances of appearing in a victim node's routing tables. If all the outgoing routes from a victim point to nodes controlled by the coalition, then *all* of the victim's access to the overlay network is mediated (and possibly censored) by the coalition. It's necessary, therefore, to guarantee that nodeIds are assigned randomly.

The simplest design to perform secure nodeId assignments is to have a centralized authority that produces cryptographic nodeId certificates, a straightforward extension to standard cryptographic techniques: rather than binding an e-mail address to a public key, these certificates instead bind a nodeId, chosen randomly by the server, to a public key generated by the client machine. The server is only consulted when new nodes join and is otherwise uninvolved in the actions of the P2P system. As such, such a server would have no impact on the scalability or reliability of the P2P overlay.

Regardless, to make such a design work, we must concern ourselves with Sybil attacks [64], wherein a hostile node or coalition of nodes might try to get a large number of nodeIds. Even if those nodeIds are random, a large enough collection of them would still give the attackers disproportionate control over the network. The best solution we currently have to this problem is to moderate the *rate* at which nodeIds are given out. Possible solutions include charging money in return for certificates or requiring some form of external authentication. While it may be possible to use some form of cryptographic puzzles [65], these still allow attackers with large computational resources to get a disproportionate number of nodeIds.

An open problem is assigning random nodeIds without needing a centralized authority. We considered a number of possibilities, including variations on cryptographic puzzles and multi-party bit-commitment schemes. Unfortunately, all such schemes appear to open the possibility that an attacker can rejoin the network, repeatedly, and eventually gain an advantage.

Ejecting Misbehaving Nodes

Existing models and simulations show Pastry can route successfully when as many as 30% of the nodes in the P2P overlay network are malicious. However, it would be preferable to have mechanisms to actively *remove* malicious nodes when they are detected. An interesting open problem is how to remove a malicious node from the overlay. While all P2P overlays must have provisions for recovering when a node fails, we would like these mechanisms to be invocable when a node is still alive and functioning. When one node accuses another of cheating, there needs to be some way that it can *prove* its accusation, in order to convince other nodes to eject the malicious node from the network.

While such a proof may be generated at the application layer, it's not clear how such a proof could be generated at the routing layer. If a node is simply dropping messages with some probability or is pretending that perfectly valid nodes do not exist, such behavior could also be explained by failures in the underlying Internet fabric. Addressing this, in general, is an interesting open problem.

Trust in P2P Overlays

P2p systems generally require a remarkable amount of trust from their participants. A node must trust that other nodes implement the same protocols and will respect the goals of the system. In previous sections, we have discussed how mechanisms can be developed to work around a certain percent of the nodes violating the rules, but there are many other aspects where trust issues arise.

Popularity When documents are requested based on keywords, rather than cryptographically strong hashes, it becomes possible for an adversary to spoof the results. The recording industry, in particular, has apparently been deploying “decoy” music files in p2p networks that have the same name as music files by popular artists. The decoy files have approximately the correct length, but do not contain the desired music. Similar issues have traditionally hurt search engines, where any page with a given search term inside it had an equal chance of appearing highly on the search results. The best solution to the search engine problem, as used by Google’s PageRank technology, has been to form a notion of popularity. For Google, pages that are linked from “popular” pages are themselves more popular. An interesting issue is how to add such a notion of popularity into a p2p storage system. It might be possible to extend the audit logs, from Section 4.2, to allow nodes to indicate the value, or lack thereof, of a given file. If users can then rank each others rankings, this could potentially allow the creation of a system comparable to Google’s PageRank.

Code Fundamentally, p2p systems require the user to install a program on their computer that will work with other p2p nodes to implement the system. Since many applications can be built on a generic p2p substrate, an interesting issue becomes how to distribute the code to support these p2p applications. Users should not necessarily trust arbitrary programs, written by third parties, to run on their system. Recently, some commercial p2p systems were discovered to redirect sales commissions from online purchases to the P2P developers [66] and might also sell the use of CPU cycles on a user’s computer to third parties, without the user getting any reimbursement [67]. Why should a user arbitrarily grant such privileges to P2P code? In many respects, this same problem occurred with active networks [68], except, in those systems, the computational model could be restricted [69]. For P2P systems, where applications can perform significant computations and consume vast amounts of disk storage, it would appear that a general-purpose mobile code security architecture [70] is necessary.

Other Works

P2P systems have been designed in the past to address numerous security concerns, providing anonymous communication, censorship resistance, and other features. Many such systems, including onion routing [71], Crowds [72], Publius [73], and Tangler [74], fundamentally assume a relatively small number of nodes in the network, all well-known to each other. To scale to larger numbers of nodes, where it is not possible to maintain a canonical list of the nodes in the network, additional mechanisms are necessary. Some recent P2P systems have also been developed to support censorship resistance [75] and anonymity [76, 77].

Sit and Morris [78] present a framework for performing security analyses of P2P networks. Their adversarial model allows for nodes to generate packets with arbitrary contents, but assumes that nodes cannot intercept arbitrary traffic. They then present a taxonomy of possible attacks. At the routing layer, they identify node lookup, routing table maintenance and network partitioning / virtualization as security risks. They also discuss issues in higher-level protocols, such as file storage, where nodes may not necessarily maintain the necessary invariants, such as storage replication. Finally, they discuss various classes of denial-of-service attacks, including rapidly joining and leaving the network, or arranging for other nodes to send bulk volumes of data to overload a victim’s network connection (i.e., distributed denial of service attacks).

Dingledine *et al.* [79] and Douceur [64] discuss address spoofing attacks. With a large number of potentially malicious nodes in the system and without a trusted central authority to

certify node identities, it becomes very difficult to know whether you can trust the claimed identity of somebody with whom you have never before communicated. Dingledine proposes to address this with various schemes, including the use of micro-cash that allow nodes to build up *reputations*.

Bellovin [80] identifies a number of issues with Napster and Gnutella. He discusses how difficult it might be to limit Napster and Gnutella use via firewalls, and how they can leak information that users might consider private, such as the search queries they issue to the network. Bellovin also expresses concern over Gnutella's "push" feature, intended to work around firewalls, which might be useful for distributed denial of service attacks. He considers Napster's centralized architecture to be more secure against such attacks, although it requires all users to trust the central server.

4.4. Pervasive/Ubiquitous Computing

Marc Weiser [89] painted a picture of computing technology weaving itself in to the very fabric of everyday life, to the point where it is impossible to define the boundaries of computing technology. Weiser's argument was that we need to get rid of the box to see a truly seamless integration of computing in people's working, domestic, and leisure lives. He put forward the view that ubiquity will have been achieved only when computing has become invisible (i.e., microprocessors are embedded in the everyday object we use but we are largely unaware of it) and there is "intelligent"communication between the objects that "anticipate" our next move. After that, technology has advanced along many dimensions, especially in hardware progress and wireless communication technologies. A number of leading technological organizations are exploring Pervasive Computing. But it is far from Weiser's vision become reality. Pervasive Computing will be the future. Pervasive computing will be a fertile source of challenging research problems in computer systems for many years to come.

4.4.1. Introduction to Security

The security of pervasive/ubiquitous computing refers to establish mutual trust between infrastructure and device in a manner that is minimally intrusive. In such environment, a smart device can recognize the user through a sort of Universal Remote Control which the user keeps secured. Secure transient association is used when the user is deploying devices and imprinting can be used to establish shared secret.

There are interesting and challenging problems in providing consistency in the management of security and in specifying authorization policies for pervasive/ubiquitous computing environments. Security can be implemented in heterogeneous components such as firewalls, different computer operating systems and multiple databases. The pervasive/ubiquitous computing system should support secure sensitive or high-value transactions and verifies that messages were not modified while in transit from queue to queue.

Authentication is one of the most important characteristics of ubiquitous computing security. Authentication provides confirmation of user access rights and privileges to the information to be retrieved. During the authentication process, a user is identified and then verified not to be an imposer. The authentication process is the assurance process that a party to some computerized transaction is not an impostor.

4.4.2. Pervasive/Ubiquitous Computing Initiatives

Both academia and industry have recently advanced pervasive/ubiquitous computing projects. Although our selection is far from exhaustive, it suggests the current state of the art in pervasive computing.

Aura

Carnegie Mellon University characterizes its Aura project [91] as “distraction free ubiquitous computing.” The project aims to design, implement, deploy, and evaluate a large scale computing system demonstrating a “personal information aura” that spans wearable, handheld, desktop, and infrastructure computers.

Aura is a large umbrella project with many individual research thrusts. Darwin is an intelligent network at Aura’s core. Coda is a distributed file management system that supports nomadic file access, and Odyssey provides operating system support for resource adaptation.

These products and others are evolving within the Aura project, which emphasizes pervasive middleware and application design.

Endeavour

The University of California at Berkeley’s Endeavour project [92] is an academic effort that focuses on the specification, design, and prototype implementation of a planet scale, self-organizing, and adaptive “information utility.” This smart environment is pervasive – everywhere and always there – with components that flow through the infrastructure, shape themselves to adapt to their usage, and cooperate on tasks.

Endeavour’s key innovative technological capability is its pervasive support for fluid software. It includes processing, storage, and data management functionality to arbitrarily and automatically distribute itself among pervasive devices and along paths through scalable computing platforms that are integrated with the pervasive networking infrastructure. The system can compose itself from pre-existing hardware and software components to satisfy a service request while advertising the services it can provide to others.

Oxygen

The Oxygen project [93], an MIT initiative, envisions a future in which computation will be freely available everywhere, like oxygen in the air we breathe. The project rests on an infrastructure of mobile and stationary devices connected by a self-configuring network. This infrastructure supplies abundant computation and communication, which are harnessed through system, perceptual, and software technologies to meet user needs.

The Oxygen project is focusing on eight environment-enabling technologies. Its emphasis is on understanding what turns an otherwise dormant environment into an empowered one to which users shift parts of their tasks.

Portolano

In its Portolano project [94], the University of Washington seeks to create a test-bed for investigating pervasive computing. The project emphasizes invisible, intent-based computing, which infers users’ intentions via their actions in the environment and their interactions with everyday objects.

Project devices are highly optimized to particular tasks so that they blend into the world and require little technical knowledge on the user’s part. In short, Portolano proposes an infrastructure based on mobile agents that interact with applications and users. Data-centric routing automatically migrates data among applications on the user’s behalf. Data thus becomes “smart,” and serves as an interaction mechanism within the environment.

Sentient Computing

AT&T Laboratories, Cambridge, UK, is collaborating with the Cambridge University Engineering Department on the Sentient Computing project [95]. The project explores user interfaces that employ sensors and resource status data to maintain a world model shared by users and applications.

The world model for the Sentient Computing project covers an entire building. Interfaces to programs extend seamlessly throughout the building. Computer desktops follow their owners

and reflect real-time updates for object locations. This project has led to some new kinds of applications, like context-aware filing systems and smart posters.

Cooltown

Hewlett-Packard's pervasive computing initiative, Cooltown [96], focuses on extending Web technology, wireless networks, and portable devices to create a virtual bridge between mobile users and physical entities and electronic services.

Cooltown uses URLs for addressing, physical beaconing and sensing of URLs for discovery, and localized Web servers for directories to create a location-aware system that supports nomadic users. It leverages Internet connectivity on top of this infrastructure to support communications services.

EasyLiving

The EasyLiving project [97] of Microsoft Research's Vision Group is developing an architecture and related technologies for intelligent environments. The project supports research addressing middleware, geometric world modeling, perception, and service description. Key system features include computer vision, multiple sensor modalities, automatic and semiautomatic sensor calibration, and device-independent communication and data protocols.

WebSphere Everyplace

IBM's pervasive computing work focuses on applications and middleware that extend its WebSphere software platform (www-3.ibm.com/software/pervasive/). The company is spearheading consortia and initiatives for open standards to support pervasive computing applications. It is also working with hardware vendors such as Palm (www.palm.com), Symbol Technologies (www.symbol.com), and Handspring (www.handspring.com) to develop a new generation of devices.

4.4.3. Issues and Challenges

As a superset of mobile computing, pervasive/ubiquitous computing subsumes mobile computing research issues while opening up new ones unique to itself. In all cases, pervasive applications should disappear into the environment.

Privacy

Protecting the privacy of users is of central importance. In a ubiquitous computing environment, sensors are actively collecting user data, much of which can be very sensitive and valuable. The data collected will often be streaming at high rates (video and audio) and it must be dealt with in real-time. In addition, there could be hundreds of tiny computers in every room, all capable of sensing people near them.

How is privacy maintained when location and activity are tracked (and predicted) by the environment? Imagining that there many computers linked by high-speed networks where messages can be intercepted and recorded by unauthorized people. Effective solutions for controlling access to data in such technology-rich environments remain to be a challenge for some time to come.

Scalability

Future pervasive/ubiquitous computing environments will likely face a proliferation of users, applications, networked devices, and their interactions on a scale never experienced before. As environmental smartness grows so will the number of devices connected to the environment and the intensity of human-machine interactions.

Traditional development requires recreating the application for each new device. Even if an enterprise could generate new applications as fast as it adds new devices, writing

application logic only once – independent of devices – would have tremendous value in solving the applications scalability problem.

Furthermore, applications typically are distributed and installed separately for each device class and processor family. As the number of devices grows, explicitly distributing and installing applications for each class and family will become unmanageable, especially across a wide geographic area.

Heterogeneity

Conversion from one domain to another is integral to computing and communication. Assuming that uniform and compatible implementations of smart environments are not achievable, pervasive computing must find ways to mask this heterogeneity – or uneven conditioning,3 as it has been called – from users. For instance, a sophisticated laboratory and a department store may always differ in their infrastructural smartness. Pervasive computing must fill this gap at some level, say middleware, to smooth out “smartness jitter” in the user’s experience.

For networking, developers have faced protocol mismatch problems and learned how to tackle the large dynamic range of architectural incompatibilities to ensure trans-network interoperability. Mobile computing has already achieved disconnected operation, thereby hiding the absence of wireless coverage from the user. Middleware may borrow similar concepts to dynamically compensate for less smart or dumb environments so that the change is transparent to users.

But the real difficulty lies at the application front. Today, applications are typically developed for specific device classes or system platforms, leading to separate versions of the same application for handhelds, desktops, and cluster-based servers. As heterogeneity increases, developing applications that run across all platforms will become exceedingly difficult.

Integration

Though pervasive computing components are already deployed in many environments, integrating them into a single platform is still a research problem. The problem is similar to what researchers in distributed computing face, but the scale is bigger. As the number of devices and applications increases, integration becomes more complex. For example, servers must handle thousands of concurrent client connections, and the influx of pervasive devices would quickly approach the host’s capacities. We need a confederation of autonomous servers cooperating to provide user services.

Integrating pervasive computing components has severe reliability, quality of service, invisibility, and security implications for pervasive networking. The need for useful coordination between confederation components is obvious. This coordination might range from traditional areas such as message routing or arbitrating screen usage to new challenges such as deciding which application can use a room’s light intensity to communicate with the user. For a wide area federation, message access is the primary requirement. Routing between servers introduces the possibility of messages from a single producer using multiple paths and, hence, arriving at a consumer out of order or duplicated.

Invisibility

A system that requires minimal human intervention offers a reasonable approximation of invisibility. Humans can intervene to tune smart environments when they fail to meet user expectations automatically. Such intervention might also be part of a continuous learning cycle for the environment. To meet user expectations continuously, however, the environment and the objects in it must be able to tune themselves without distracting users at a conscious level.

A smart environment can implement tuning at different system levels. For example, network-level devices will require auto-configuration. Current manual techniques for

configuring a device with addresses, subnet masks, default gateways, and so on are too cumbersome and time-consuming for pervasive computing.

Automated techniques to dynamically reconfigure the network when required are also crucial to realizing the pervasive computing vision.

Perception: Context awareness

Most computing systems and devices today cannot sense their environments and therefore cannot make timely, context-sensitive decisions. Pervasive computing, however, requires systems and devices that perceive context. Mobile computing addresses location- and mobility-management issues but in a reactive context – responding to discrete events. Pervasive computing is more complex because it is proactive. Intelligent environments are a prerequisite to pervasive computing.

Perception, or context-awareness, is an intrinsic characteristic of intelligent environments. Implementing perception introduces significant complications: location monitoring, uncertainty modeling, real-time information processing, and merging data from multiple and possibly disagreeing sensors. The information that defines context awareness must be accurate; otherwise, it can confuse or intrude on the user experience.

ComMotion, a location-aware computing environment that addresses these issues for mobile users, is under development at the MIT Media Lab [90] Microsoft Research is investigating Radar7 an in-building location-aware system.

Smartness: Context management

Once a pervasive computing system can perceive the current context, it must have the means of using its perceptions effectively. Richer interactions with users will require a deeper understanding of the physical space.

Smartness involves accurate sensing (input) followed by intelligent control or action (output) between two worlds, namely, machine and human. For example, a pervasive computing system that automatically adjusts heating, cooling, and lighting levels in a room depending on an occupant's electronic profile must have some form of perception to track the person and also some form of control to adjust the ventilation and lighting systems.

4.5. Mobile Computing

Mobile computing, that is the ability of having computing and communication abilities on the move, depends on the existence of a suitable distributed systems infrastructure. So, security considerations of mobile computing can be seen as extensions to those of distributed computing. The security issues in mobile computing are therefore examined on the basis of known security issues of information systems.

Security of distributed systems

Security of distributed systems is a critical issue, as it is difficult to provide in such an environment physically secure communication and to co-ordinate multiple management policies. A distributed system is susceptible to a number of threats both from legitimate users of the system and from intruders. Two general types of security threats are the *host compromise* and the *communication compromise*. Host compromise security threats refer to various degrees of subversion of individual hosts. Possible attack categories are the followings:

- *Masquerading*: when a user is masquerading as another to gain access to a system object to which he is not authorized.
- *Unauthorized use of resources*: when a user is accessing system object without having authorization. This situation may lead to theft of computing resources or improper use of information.
- *Disclosure of information*: unauthorized reading of stored information.

- *Alteration of information*: unauthorized writing into stored information.
- *Denial of service*: the attacker acts to deny resources or services to entities which are authorized to use them, e.g. by locking a file.

The communication compromise security threats refer to threats associated with message communication. Possible attacks can be categorized as follows:

- *Masquerading*: when a user is deceiving about its real identity. Masquerading may lead to impersonation.
- *Unauthorized use of resources*: when a user is accessing a network component without have being authorized. This situation may lead to theft or improper use of communication resources.
- *Interception*: The opponent gains access to the data transmitted over the communication link. Two types of interception are distinguished: *disclosure of information* (the opponent obtains information transmitted over the link), and *traffic analysis* (the opponent observes the message patterns and derives information about the identities and locations of the communicating parties, the message frequency and length, etc.).
- *Alteration of resources and information*: The opponent modifies the messages transmitted, alters their sequence or delays them. Unauthorized alteration of information may occur through active wire-tapping. This threat may also involve unauthorized introduction (removal) of resources into (from) a distributed system.
- *Fabrication*: The opponent inserts information into the communication link. A special type of this attack is replay of old messages in order to mislead the communicating parties.
- *Repudiation of actions*: This is a threat against accountability. A repudiation attack may occur whereby the sender (receiver) of a message denies having sent (received) it.
- *Denial of service/Interruption*: The attacker prevents the easy transmission of information.

Finally, the security functions and controls that can be used in distributed systems include:

- *Identification and Authentication*: Authentication information and mechanisms that involve trusted third parties (passwords, cryptographic techniques, challenge-response techniques).
- *Access control and Authorization*: Access control information, access control rules, delegation.
- Information *confidentiality*: Confidentiality mechanisms (encryption) and attributes (secret keys, public and private keys).
- Information *integrity*: Integrity mechanisms that provide generation and verification of integrity checks.
- *Non-repudiation*: (e.g. through digital signatures).
- *Auditing* and *Accountability*.
- *Availability* and *Prevention of Denial of Service*.

Security in mobile unit extensions

When distributed systems include mobile parts, we face several additional security problems. Usually those that stream from distributed systems gain interest, for example delegation, while others, as for example authentication and encryption, must eliminate the system load they produce as a result of their completeness. Some properties of mobile computing systems that also affect security are broadcast base communications (ease accessible to eavesdroppers), crossing boundaries of administration domains with high

heterogeneity, disconnections, physical constraints of mobile devices, high dependence on the infrastructure, highly distributed environment, etc.

Security and delegation

The security provisions used in mobile computers must operate in a dynamic and fluid communication environment. Furthermore, the sub-networks may be physically distributed and may not geographically overlap. Thus, a computer may have to switch communications between different kinds of sub-networks and it may become disconnected as it moves. The ability to *delegate* limited authority is essential to realize security in a ubiquitous computing environment. A *delegation* is a temporary permit issued by the delegator and given to the delegate who becomes limited authorized to act on the delegator's behalf.

Delegation is a well understood problem but there are special considerations for mobile systems that these existing approaches do not address. The usual requirements for a general delegation scheme include:

- *Revocation*: the delegator must have the ability to cancel delegations it has issued.
- *Cascading*: the delegate must have the option to create delegations on the delegator's behalf.
- *Restriction*: The delegator must be able to limit the rights granted by any delegation.

The delegation for mobile computers has, however, the following additional requirements:

- *Disconnected Delegates*: Since a client may disconnect after issuing a delegation, delegations must succeed even if the delegator is currently not attached.
- *Low Resource Usage*: It is crucial that bandwidth and host resource usage be minimized.
- *Frequent Creation and Revocation*: Delegations may be issued and revoked frequently as mobile hosts detach and reattach to the system.
- *Interoperability*: Delegation should be as independent of the underlying protocols and system software as possible to promote interoperability.

Security and mobility

In mobile computing it is sometimes difficult to achieve the required isolation and self-efficiency due to the relatively limited resources available to a mobile unit, which makes it necessary to communicate with the mobile support station. The mobility of users and data that they carry introduces, therefore, security problems from the point of view of the *location* of a user and the secrecy and authenticity of the data exchanged. A user on a mobile wireless network may choose, for example, to have the information concerning his existence treated as being confidential. That is a user may choose to remain *anonymous* to the majority of other users on the network, with the exception of a select number with whom the user often interacts.

Another potential security problem lies in the possibility of *information leakage*, through the inference made by an attacker *masquerading* as a mobile support station, who may issue a number of queries to the database at the user's home node or to database at other nodes, with the aim of deducing parts of the user's profile containing the patterns and history of the user's movements.

Related to the management of these databases is the issue of *replication* of certain parameters and user profiles with the aim of replicating the environments surrounding the user. Thus, as the user roams across zones, the user must not experience degradation in the access and latency times.

In general, as sensitive data is replicated across several sites, the security risks are also increased due to the multiplication of the *points of attack*.

Security and disconnections

Differing levels of disconnection may be introduced, ranging from the normal connection to connections using low bandwidth channels. A crucial aspect of disconnection is the elective or non-elective nature of a disconnection. Security and integrity problems may occur in the case of frequent disconnections caused by *hand-offs* that occur when the mobile unit crosses zones/cells.

The transition from one level of disconnection to another may present an opportunity for an attacker to *masquerade* either the mobile unit or the mobile support station. An attacker should not be able to 'hijack' the communications of a mobile unit which is *stepping-down* its level of connection and then masquerading as the mobile unit. Similarly, an attacker must not masquerade as a mobile support station to a mobile unit that is about to *step-up* its level of connection.

4.6. Security Shortcomings in Existing Systems

The severe problem in the various security mechanisms, described in the previous sections, is the non-existence of the consideration of all the typical parameters of large scale open networked systems. These security solutions are generally inherited from the previous technologies and the designers and developers have tried to adapt them to these contemporary computing systems.

Our vision to handle the security issues of the emerging large scale open networked systems and applications is to develop a comprehensive security architecture based on the specific security requirements of these systems. The characteristics of these systems include:

1. Heterogeneity of the system components
2. Diversity of the set of security services
3. Large set of users with varied security requirements
4. Extensive negotiations between the applications and the resources/stakeholders
5. Establishment of security relations without relying on central infrastructure
6. Etc.

Based on these specific characteristics, we have proposed a security architecture (described in chapter 5) that employs the concept of virtualization of security services to handle the problem of heterogeneity of the system components and diverse nature of the security services. A security handler is used to absorb the heterogeneity of the underlying architecture and to provide a homogeneous interface to the user applications. The varied security requirements of the users are satisfied by the configurable/pluggable nature of the security services so that different kind of users can invoke the set of services they are interested in. We have proposed the concept of ‘Security broker’ that performs the security negotiations like a resource broker in a distributed system dispatches various jobs to the most suitable resource of the system. We have also employed security bootstrapping to setup security relations in these environments without requiring the user to be an expert and without relying on central infrastructure.

The details of our proposed security architecture and its salient features are elaborated in the following chapter.

Chapter 5

Proposed Architecture

5.1. Overview

In the large scale distributed systems, like computational Grid, the need for efficient and secure data transportation over potentially insecure channels creates new security and privacy issues, which are exacerbated by the heterogeneous nature of the collaborating resources. Traditional security approaches require adequate overhauling to address these paradigms. In this thesis, we propose a new two-pronged approach to address these security issues – VIPSEC: Virtualized and Pluggable Security Services Architecture. First, the virtualization of security services provides an abstraction layer on the top of the security infrastructure, which harmonizes the heterogeneity of underlying security mechanisms. Second, the configurable/pluggable nature of the various security services permits the users and resource providers to configure the security architecture according to their requirements and satisfaction level. This approach allows the security infrastructure to develop with minimal impact on the resource management functionalities.

Since security implementations are more and more numerous and complex, it has become almost impossible for an inexperienced user to understand their meaning and especially how they should be used. Additionally, the heterogeneity of networks does not simplify the understanding and definition of a security system. Therefore, it is currently impossible to establish a security policy for a communication by using the low level properties of the different networks that are being crossed. The classical solution to this problem consists in setting up a secured high-level ciphered tunnel from end to end. This is acceptable in some situations, but it may not satisfy future evolutions of networks. The goal of virtualization is to reinstate security principles (transparency, responsibility, traceability, etc.), security objectives (integrity, availability, confidentiality, etc.), security policies (protection, deterrence, vigilance, etc.) and security functions (identification, authentication, access control, management of secret elements, privacy, etc.) in their rightful place. Virtualization aims at describing a policy and at refining it. Actually, a unique security policy cannot be implemented on several heterogeneous networks, architectures or environments. The current complexity of networks comes from the fact that on the one hand each element defines its own security policy in accordance with the security domain to which it pertains (a priori…), and on the other hand each security domain has its own security policy. In the virtual paradigm, the policy of the element (wherever it may be) shall be merged with the policy of the domain to which it belongs. Then, this policy will be automatically implemented depending on the available security functions.

Virtualization is a powerful principle used in Computer Science to conceive of a heterogeneous computerized reality in a different manner by reducing its visible complexity. Indeed, virtualization enables attaching physical or logical resources that are incompatible, heterogeneous and exploded in order to render their heterogeneity invisible to certain subjects (such as the users), while at the same time, rendering them more attachable to other subjects, especially to the security hooks that will be able to capture these resources and handle them more efficiently. In practice, virtualization is created by adapted mechanisms that are distinctive every time.

- In general, it is achieved by masking a variety of dissimilar worlds or mechanisms. This veil erases the roughness of the computerized reality and enables the creation of logical hooks to get hold of the system components in a different and more efficient manner.
- This could also take the form of a short-circuit. In this case, it will be an autonomous logical device superimposed on the existing system, enabling to smooth out or divert the normal functioning of the system.

Virtualization is not only an abstraction since it also implements appropriate security functions (at the lowest level possible). Security policies of communicating elements will possibly be merged and implemented independently on every security domain along the communication path. This must be carried out without disturbing the existing structures and without creating a unique and inflexible world. Thanks to virtualization, security becomes conformable, slipped into the existing systems by adapting to it each time. Therefore, security must be tailored according to risk-taking (following the mission that one has set), i.e., the potential threats versus the decisions that we take.

A virtualization engine is a black box, a “virtual machine” containing a mechanism that hides and dissimulates the complexity of several distinctive physical and logical IT resources. These virtual devices will operate on the original operating system. The virtual machines should be designed based on a “sandbox” security model, like the Java virtual machine.

5.2. Virtualization

The concept of virtualization in information technology finds its roots in the very earliest software. The first programmable digital computers dealt in the world of 0s and 1s – Programs were 0s and 1s, output consists of 0s and 1s. As a result, programming was very difficult and programs were quite opaque. Then the compiler programs came into existence that let programmers work with English-like (high-level) languages like COBOL. The compiler took the COBOL code, crunched it, and spit out the 0s and 1s object code that the computers actually understood. The COBOL compiler, therefore, virtualized the object code.

As computers grew more powerful and complex, virtualization and encapsulation techniques continued to provide additional levels of abstraction. Timesharing mainframe computers allowed users to have virtual control of the machines. Another example is the graphical user interface, which provided virtual access to underlying system resources. Component architectures also provide virtual representations of distributed computing infrastructures. At every step, software allowed people to work with relatively simple tools that accessed complex systems behind the scenes.

Service orientation, then, is an evolutionary step in this inexorable progression to the next level of abstraction for distributed computing. By encapsulating software components, applications, and underlying systems with Web Services Interfaces and then virtualizing these fine-grained functional Web Services into coarse-grained business Services, companies have agile IT infrastructures that provide business agility.

5.2.1. Virtualization in the Context of Security Architecture

From the security point of view, the virtualization of a service definition encompasses the security requirements for accessing that service. The need arises in the virtualization of security semantics to use standardized ways of segmenting security components (e.g., authentication, access control, etc.) and to provide standardized ways of enabling the federation of multiple security mechanisms. The benefits of having a loosely-coupled, language-neutral, platform-independent way of linking and securing applications within organizations, across enterprises, and across the Internet is fundamental to the problem set addressed by the large scale open services architecture.

5.2.2. Need of Virtualization

The secure interoperability between virtual organizations demands interoperable solutions using heterogeneous systems. Virtualization permits each participating end-point to express the policy it wishes to see applied when engaging in a secure conversation with another end-point. Policies can specify supported authentication mechanisms, required integrity and confidentiality, trust policies, privacy policies, and other security constraints. A security services handler is shown in figure 5.1

Figure 5.1 : Security Services Handler

The concept of virtualization of security services is needed to have the absolute freedom to choose the underlying security mechanisms. It could be extended to adapt country-specific legal requirements, population-based ethical issues, and the business-oriented interests. Moreover, virtualization could be used to achieve the best trade-off between security guarantees and processing capabilities.

5.2.3. Advantages of Virtualization

From a user's point of view, usability is a decisive factor [98]. Hence, security architecture must provide a way that the user merely needs to have technical knowledge of the underlying security infrastructure. Virtualization facilitates the development of flexible, custom-designed hierarchical security model that responds to the security needs. It dynamically delivers high performance and statistically consistent level of the accessibility to security services. Virtualization introduces an abstraction layer to manage the complexity by defining security rules for the system that are applied whenever needed.

In a virtualized security services environment, a user does not need to know which authentication mechanism (such as X.509 certificate, Kerberos ticket, etc.) he requires for accessing a remote node. If he is authorize to perform some operation(s) at the remote site then the security architecture will take care of the interpretation of credentials for the various sites.

Virtualization is a gimmick, an artifice enabling to keep the standards in place and improve them by providing additional functions, without, however changing anything in the existing setup. Virtualization makes it possible to coexist with the pre-existing structures that will be utilized by default, utilizing the diverted systems only by request. Virtualization also enables to act in the opposite way, by subordination to the additional functions by default, while relegating the normal and standard functions to exception status, somewhat like traffic relief using a bypass road on a holiday route on which those in seasonal migration naturally follow the added traffic signs to avoid traffic jams, while the natives follow the normal signs to travel locally. The existing architecture, still operating by default, is thus preserved while being outfitted with an add-on smart design and not with an add-on module.

Virtualization enables slipping into any system irrespective of its architecture without defacing it. It is a guarantee of fluidity and upgradeability. If more ambitious, this approach is not incompatible and does not conflict with the previous approaches based on addition of specific devices (firewall, intrusion detection, specific cryptographic module) intended to take charge of certain security functions within a known domain or to perform a very specific and proprietary security function of a component (secure operating system, GSM security, etc.).

It is clear that large-scale virtualization is an important lock. If classical abstraction such as XML is rapidly deployable, the case of virtualization is different, as it requires installation of specific virtual machines for all of those heterogeneous environments.

Universal types of virtualization (Java Virtual Machine, etc.) are usually heavy. The internal mechanism required to perform such virtualization may be quite voluminous (VPNs, distributed operating systems, etc.).

The cost of this approach undoubtedly involves additional computing resources needed to implement the virtualization engines in all the existing hosts (terminals, servers, routers, gateways). Moreover, additional entities will ultimately have to be created to perform this virtualization by means of virtualization gateways between two widely different adjoining networks.

Virtualization is abstract, but each time it is the implemented mechanism and not the concept that will have to prove itself. Virtualization must remain simple and effective.

5.2.4. Feasibility of Implementation

The virtualization framework can employ existing security techniques, protocols and models in a common way, by adapting/masking them through virtualization. True virtualization would allow the behaviors that exist in 'real life' to exist for the virtual world, without the people involved needing to be specialists. Finally, the framework can support the security principles, security objectives, security policies and the security functions in the evolving digital world.

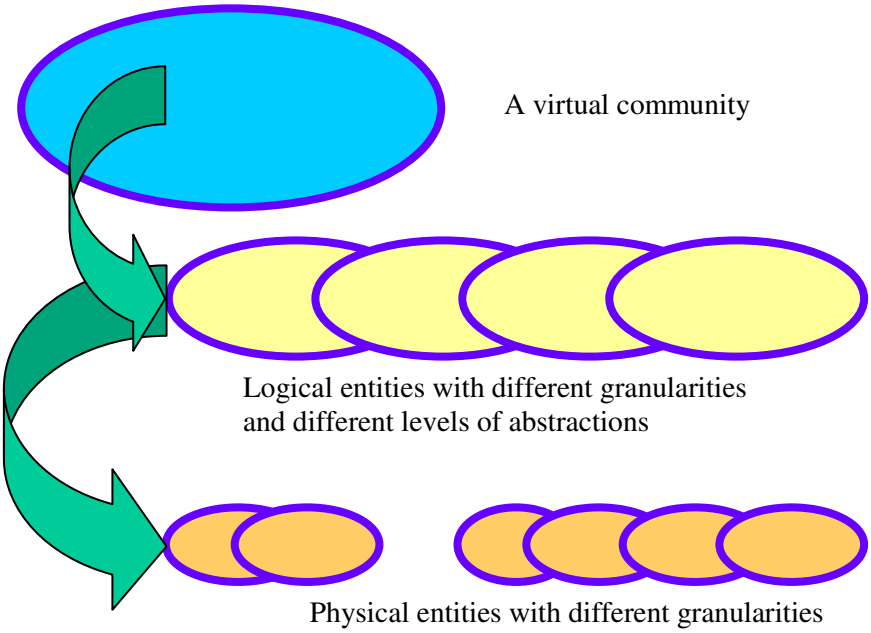

Figure 5.2: Virtualization of communities

Security in distributed systems is a field where virtualization seems to have untapped potential. If we want to use virtualization mechanisms, they will be an integral part of security and it will be necessary to make them secure.

To secure systems and/or objects, we introduce lightweight infrastructures (or middleware) in these systems and will inject objects for virtual machines (in a very general sense) to provide a different vision of things to foreign systems or external entities.

We use security functions (identification of a system or an object, authentication of a subject, key management, cryptographic protocols, etc.) by masking the heterogeneous architectures of the systems and networks, the complexity of services, etc. with other virtual concepts. Thus a system can have a group identity that appears in several machines or objects, without necessarily using a label (a name).

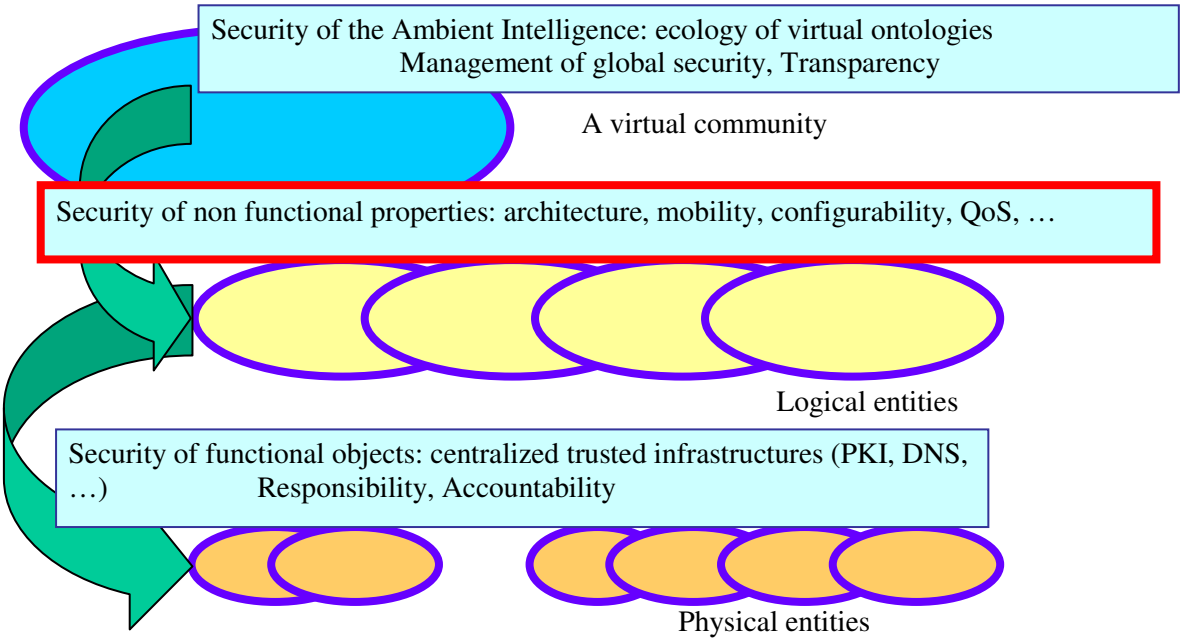

Figure 5.3: Security of Ambient Intelligence is split at 2 levels.

For example, security of Ambient Intelligence is split at 2 levels (figure 5.3). Security of non functional properties and security of functional properties. Security of non functional properties is rather placed at the articulation of the virtual mechanisms

Infospheres need to be mapped to the virtual paradigm. More has to be given to the individual, the organization and the state (rather than to devices or infrastructures). Each infosphere comprises a virtual ring. Two 'Virtual shields' can thus create a controlled zone that enhances infosphere security, not infrastructure security. A pervasive policy is simply expressed in high level, but refined in many dimensions to map to specific infrastructures, organizational or individual needs and real world events.

5.2.5. Examples of Virtualization

Virtuality is already present in computer systems, networks and distributed systems. Therefore, security must not be tied up with technology maturity as it is for the moment in GSM, or WiFi, in order to facilitate technology migration.

IPSec: An Example of Virtualization

IPSec is an example of interoperable authentication, integrity and encryption (Figure 5.4). However it is too static and does not provide security customization. Many fields in the IP header can be used to customize security and use these fields as a vehicle through heterogeneous worlds.

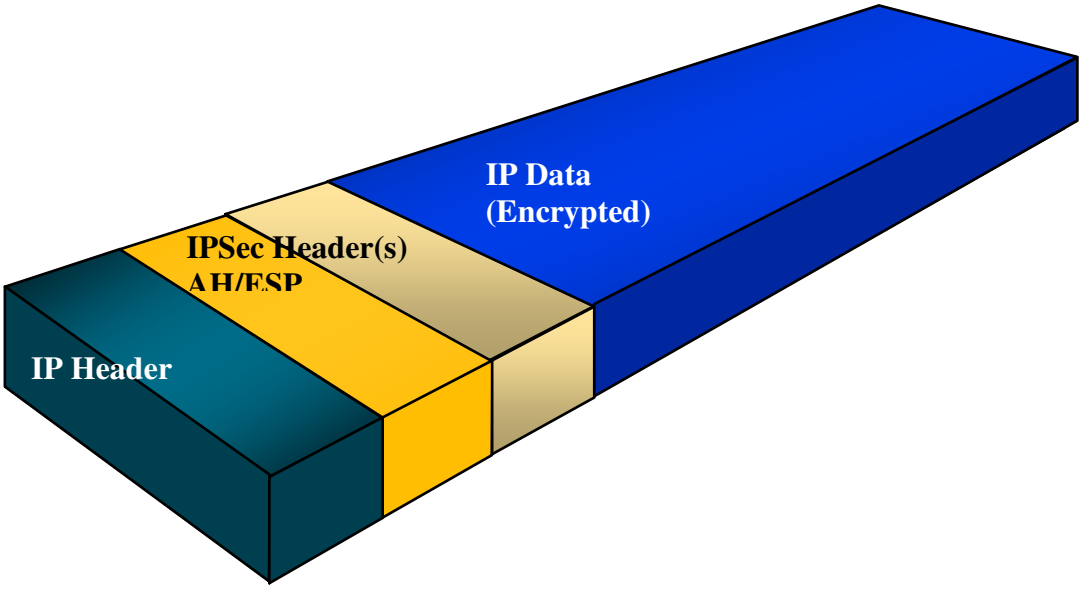

Figure 5.4: IPSec (from CISCO)

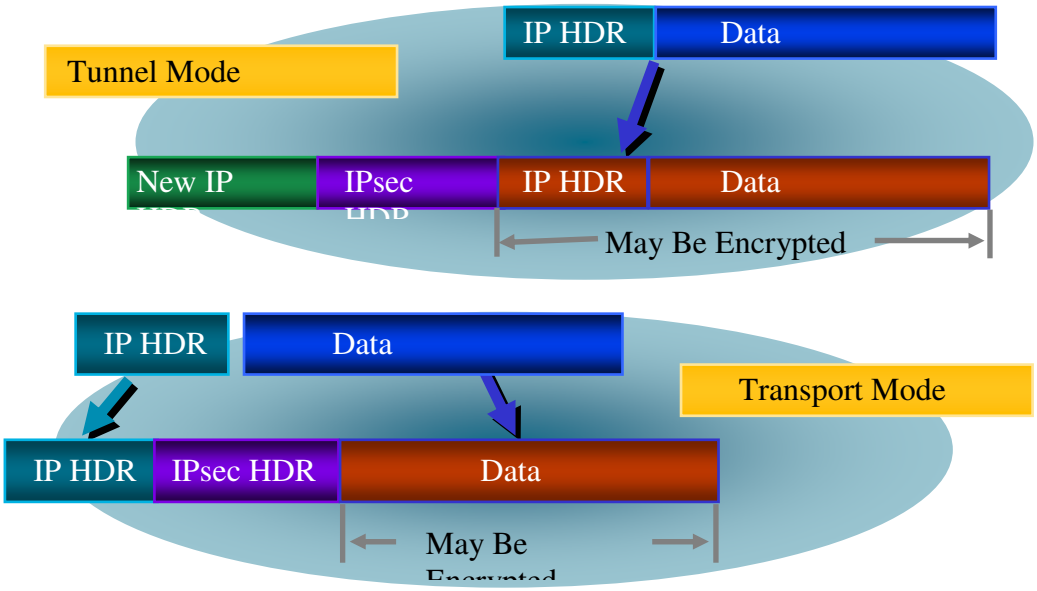

Figure 5.5: IPSec Tunnel and Transport Mode (from Cisco)

MPLS: An Example of Virtualization

MPLS (Multi-Protocol Label Switching), MPLS and GMPLS implement a virtualization of circuits with packet networks. The paradigm is a technology that integrates the **label swapping** forwarding paradigm with **network layer routing**. Label swapping is expected to improve the price/performance of network layer routing, improve the scalability of the network layer, and provide greater flexibility in the delivery of (new) routing services (by allowing new routing services to be added without a change to the forwarding paradigm). Packet

forwarding is based on labels (not IP information). It is based on the concept of label swapping similar to other layer-2 forwarding mechanisms (VPI/VCI). MPLS provides ability to have multiple labels (label stack). Other paradigms like destination based unicast routing, TE, QoS, or VPN may also be used to classify and forward packets. Once a packet is labeled it cannot be reclassified. Labels are assigned on entry into the MPLS domain. MPLS performs ingress classification function - referred to as label imposition and the location of the label within the packet/cell will depend on MPLS mode of operation

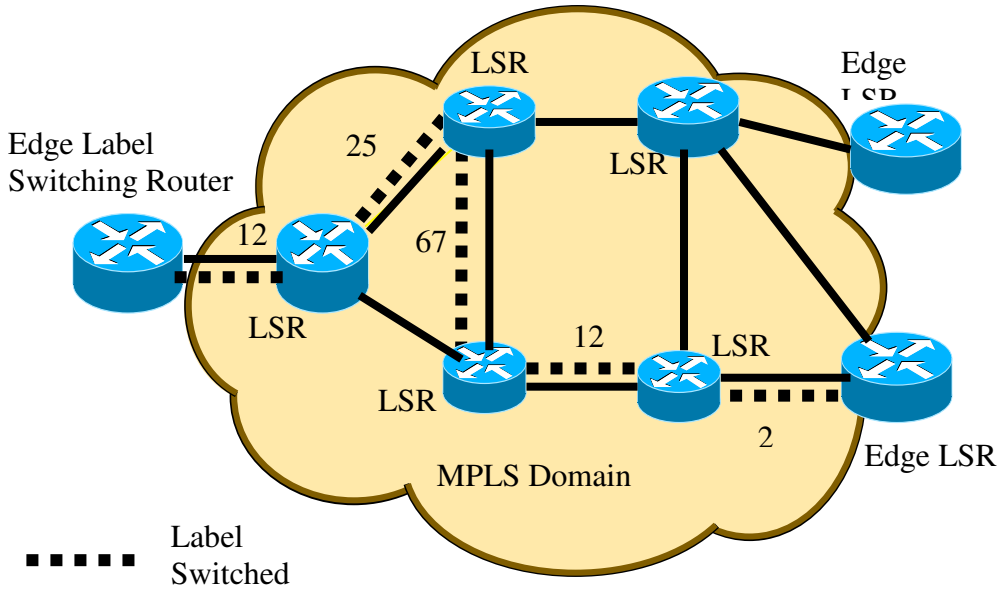

Figure 5.6: An MPLS network

Ad-hoc Networking

Ad hoc network are spontaneous wireless networks. By nature, wireless networks do not have routes, and implementing other functions like access control is not simple especially in a mesh network. Some major issues, such as service discovery, spectrum coexistence, management, and security are to be solved.

Figure 5.7: An example of ad-hoc network

There are 2 types of Ad-hoc routing algorithms that virtualize the non existing network layer either within layer 4 (robustness of the overall nodes morphology) or within the layer 2 (high mobility in the overall morphology of the graph of nodes)

5.3. Configurability

We propose a *configurable* mechanism for the invocation of security services to address security needs of the different kinds of users. This approach permits the evolution of security infrastructure with less impact on the resource management functionalities, which are still on the verge of evolution. Moreover, it permits the users and resource providers to configure the security architecture according to their requirements and satisfaction level. The set of these security services include core security services (such as authentication, authorization, identity mapping, audit, etc.) as well as contemporary security services (such as mobile access control, dynamic digital signature, etc.).

5.3.1. Pluggable Security Services (PSS)

Authentication Service is concerned with verifying proof of an asserted identity.
Authorization Service is concerned with resolving a policy based access control decision.
Identity Mapping Service provides the capability of transforming an identity that exists in one identity domain into an identity within another identity domain. This service is not concerned with the authentication of the service requestor; rather it is strictly a policy driven name mapping service.
Credential Conversion Service provides credential conversion between one type of credential to another type or form of credential. This service facilitates the interoperability of differing credential types, which may be used by services.
Policy Service is concerned with the management of policies. The policy service may be thought of as another primitive service, which is used by the authorization, audit, identity mapping and other services as needed.
Audit Service is responsible for producing records, which track security relevant events. The resulting audit records may be reduced and examined to determine if the desired security policy is being enforced.

Profile Service is concerned with managing service requestor's preferences and data which may not be directly consumed by the authorization service. This data will primarily be used by applications that interface with a person.
Privacy Service is primarily concerned with the policy driven classification of personally identifiable information (PII). Such a service can be used to articulate and enforce a privacy policy.
Encoded Communication Service is concerned with ciphering the data before it leaves the resource terminal. Participating nodes mutually agree on the encoding technique being employed to encrypt the data.
Nonrepudiation Service ensures that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.
Encrypted Data Storage Service is concerned with ciphering the data before it is stored in a storage repository. Authorized retrieving entities are aware of the encoding technique being employed to encrypt the data.
Communication Channels Service facilitates lossless transfer of communication parameters to a stand-by or a secondary communication channel in the case of main communication channel failure.

5.3.2. Requirements for Configurable/Pluggable Services

Definition of standard and flexible interfaces: To assure its proper functioning in various heterogeneous environments, PSS should define a set of standard, yet flexible, interfacing protocols. Most applications need not use these interfaces directly. Therefore, the PSS protocols interface should be exposed only to application developers interested in defining new protocols or in configuring them in novel ways.
Integration at application layer: To enable a user to invoke his desired set of security services in the beginning of a task session, PSS should offer its integration at application layer without requiring any low-level programming knowledge.
Coordinated invocation of Services: To assure secure links among the various nodes, there is a need for an identical set of invoked security services at all the ends. This coordination is important for flushing out any vulnerability that may be introduced due to mismatched services.
Usability by users and services: To assure the adaptability of the security architecture in the various operational situations, the security services invocation should be possible by not only the users but also by the appropriate computing services.
Simultaneous use of multiple services: To assure security in depth, various security services are needed simultaneously. Moreover, certain security services are dependent on other services; e.g., Invocation of Authorization service requires invocation of Authentication service. In such a situation, prerequisite services of a certain invoked service should be activated automatically.
Support for future enhancements: To maintain its usability, PSS should accommodate forthcoming enhancements to the Grid security infrastructure, such as allowing integration of semantic firewall, etc.
Optimization for various communication links: To ensure sustained communications, PSS should automatically adapt the current communication channel like wired network or wireless network (Bluetooth, 802.11, …).
Providing real-time invocation features: To make the security architecture flexible and adaptable to the needs of Grid users and services, real-time invocation and de-invocation of security services are indispensable.
Using standard programming interface: To allow the integration of security services with the various Grid systems, it should use standard programming interface. It will further enable the Grid programmers to extend it to meet their specific needs.

5.3.3. Description of the Architecture

To meet the requirements outlined in the section 5.3.2, we identified logical components, factored out common features, and defined general framework interfaces. Figure 5.8 depicts the PSS framework architecture. The security policy provides the fundamental guidelines for the various security operations. The Security Broker interacts between applications (more precisely the distributed applications) and the security services. The security broker has a security services handler (cf. figure 5.1), which is employed to absorb the heterogeneity. The layered architecture of the security broker is presented in the section 5.4. Invocation of various security services at one site requires coordination with the set of the services invoked at the other sites. This coordination is carried out by a special service called Coordination Service. The functioning of the coordination service is elaborated in section 5.4. However, it is worth mentioning here that the coordination service is guided by the security policy especially for the resolution of conflicts that arises if various grid sites try to invoke a dissimilar set of security services.

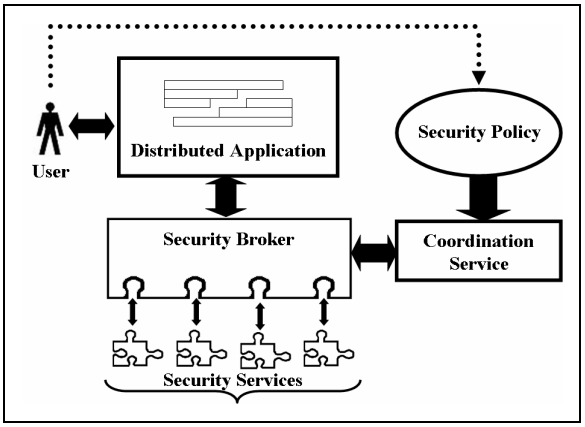

Figure 5.8: PSS Framework Architecture

5.3.4. Addition and Deletion of Security Services

Adding New Services Transparently

To add new services without making any changes to the rest of the system, a *registry* is used to maintain a collection of abstract factories. In the abstract factory pattern, a single class defines an interface for creating families of related objects, without specifying their concrete types. Subclasses of an abstract factory are responsible for creating concrete classes that collaborate among themselves. In the context of pluggable services, each abstract factory can create the *Connector*, *Acceptor*, *Profile*, and *Transport* classes for a particular service.

Adding New Services Dynamically

to configure new services dynamically, even while the system is running, a *configurator* is used that can decouple the implementation of a service from its configuration into the application. This configurator can be applied in either of the following ways:

a) The configurator can be used to dynamically load the registry class. This facade knows how to configure a particular set of services. To add new services, one must either implement a new registry class or derive from an existing one.

b) The configurator can be used to load the set of entries in a registry dynamically. For example, a registry can simply parse a configuration script and link the services listed in it dynamically. This is the most flexible design, but it requires more code to parse the configuration script and load the objects dynamically.

Services Synchronization

If the various communicating nodes invoke different sets of security services, a coordination service, as shown in figure 5.4, is used to resolve the invocation of unpaired services. A *log* of the services invoked at the various nodes is maintained and if a conflict is found in the set of invoked services in the collaborating nodes then it is resolved according to the rules set forth in the security policy. For example, if the security policy permits the

automatic invocation of corresponding services then if a VO member wishes to invoke *Audit Service* whereas the other prefers *Profile Service* then the coordination service will invoke the two services at both places.

5.3.5. Example Scenarios

In this section, we present two scenarios that depict the usefulness of our proposed security architecture. This state of security is otherwise unachievable with the existing security solutions presented in chapter 4.

These scenarios revolve around a number of parties who collaborate to create a distributed session:

Scenario 1: This scenario involves initiator-specified computation: in particular, grid-style computational offloading. In this scenario the initiator wishes to run an extensive but sensitive data mining query over a confidential data set, requiring computational and storage resources beyond those the initiator has available. For example, a pharmaceutical company wishes to measure the death rate in patients that are prescribed an experimental drug mix: the query is sensitive, in that there should be no external indication of the search parameters; and the patient data must remain confidential. Computational offloading scenarios are ideally suited for grid computing environments, where a goal is to create homogeneous, widely-available, distributed processing nodes to absorb excess local computational needs. However, although grid servers are gaining in popularity and popular usage, they currently offer few if any remotely verifiable guarantees about the security and integrity of their operating environments, making them unsuitable for application in this scenario. Our proposed security model eliminates this barrier, enabling such offloading scenarios to become commonplace for anyone who could benefit from them, while simultaneously addressing any security concerns. At role-acquisition time, the initiator might specify that it requires attestations to the effect of processing-time reservations, memory and communication isolation (with respect to other processes or entities running on the responder's system), encrypted on-disk storage of any swapped memory or source data, and confirmation that the responder's execution environment will be reset and zeroed upon completion of the service. The responder's virtual machine would then believably attest or assert that it will enforce each of these requirements. In an expanded scenario, the query may be provided by the initiator, who specifies one set of security requirements regarding the query text, whereas the patient data may come from a third-party source with much stricter requirements of verifying identity and ensuring confidentiality.

Scenario 2: This scenario involves responder-specified computations: in particular, online business services. The initiator identifies a responder who advertises that it is programmed and willing to accomplish the initiator's high level task. For example, a consumer wishes to order a book from an online broker, but desires to prevent the distributor of the book from learning any information about the consumer other than his or her address – in particular, preventing the exposure of bank or financial information that the consumer discloses in order to pay the broker. Online business services are in widespread use today, but suffer in that their usage is ad-hoc, with consumers relying only on past experience or reputation when verifying the expected behavior of different brokers, and in that users encounter different interfaces for each different broker for a given requested service. In this scenario, the consumer may desire to securely audit the broker's communications – in essence, obtaining a guarantee that the consumer will have knowledge of any unauthorized exposure – in lieu of specifying security parameters for each responder.

5.4. Security Brokering

The Security Broker mediates between applications (more precisely the distributed applications) and the security services. The security broker has a security services handler,

which is employed to absorb the heterogeneity of the underlying security services and to provide a homogeneous interface to the upper layer.

5.4.1. An Analogy of Resource Brokering

The idea of introducing a security services broker is actually inspired by the utilization of a brokering agent for the exploitation of suitable computing/storage resource (also known as the resource broker) in distributed applications.

The virtual security services handler shown in figure 5.1 could be seen as a part of the security broker (as shown in figure 5.9) that interacts between the core security architecture and the applications. Such an arrangement with no direct interaction between applications and core security architecture will raise the protection level of the security infrastructure from the malevolent applications.

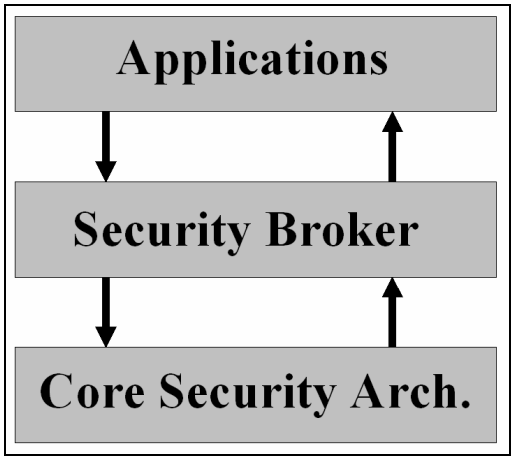

Figure 5.9 : Induction of a Security Broker

5.4.2. Distributed Virtual Engine

This concept of virtualization of security services can be realized through distributed virtual engines that will enable security service calls to be unified according to requirements and not according to the technologies to be supported.

Distributed virtual engines are implemented by using brokering agents for the security services.

5.4.3. Coordination between Applications and Core Security Architecture

This component of PSS is responsible for the surety that a coordinated set of security services are invoked at the various sites of the VO. It contains traces of all the services invoked at the various nodes (cf. figure 5.10). When a user invokes a set of services (default or user-defined) and it does not match with the set of services invoked at the other nodes then the mismatch is identified as *conflict* in the invoked services which is managed in the light of the security policy. Once the conflict is resolved, security services invocation is made to the security broker.

It is worth mentioning here that this security broker is not involved in the conflict management itself, rather it forwards the service invocation, made by a user/service, to the coordination service for its mapping and to look for any conflict(s) with the security services invoked at the other nodes. The security services are invoked by the security broker only when it receives a command from the coordination service.

In the arrangement shown in figure 4, the security broker is deliberately placed between the application and the coordination service so as to isolate the latter from the former. One of the objectives of the security broker is to isolate the core security architecture from

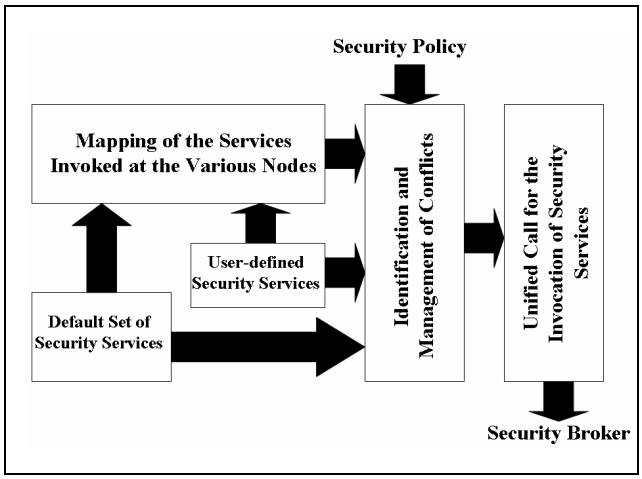

Figure 5.10: Coordination Service Architecture

the applications running over the grid to increase the protection level. The need to protect the coordinated service is evident from the fact that if some malicious user/application succeeds in influencing it then the mismatch of the various security services invoked at the various nodes will cause the self-destruction of the entire security architecture.

5.4.4. Layered Architecture

The layered architecture of the proposed security broker is shown in figure 5.11. The functionalities associated with these layers are:

1. **Application/Client Interface** authenticates the user/application and provides the glue between the user/application and the underlying security broker infrastructure to facilitate communications between them.
2. **Configuration Daemon** is a configuration server. It accepts a machine independent, abstract configuration request and then interacts with the coordination service through a secure channel. It notifies when the coordination service approves the security service configuration. It can keep a log of configurations done or even a complete backup configuration.

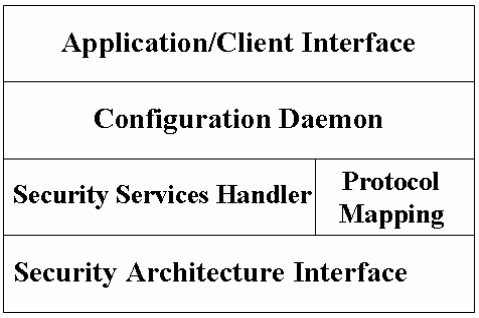

Figure 5.11: Security Broker Architecture

3. **Security Services Handler** absorbs the diversity of the security mechanisms to enable security service calls to be unified according to requirements and not according to the technologies to be supported (cf. figure 5.1).
4. **Protocol Mapping** contains a comprehensive list of the protocols supported by the security architecture through the Security Services Handler.
5. **Security Architecture Interface** consists of *socket* modules to plug various security services. Call for a particular security service is sent to the security services handler through the Configuration Daemon. The security services handler checks the existence of such a security service from the security protocol mapping and if it exists then an instance is invoked to hook the corresponding security service to the security architecture interface.
6. **Real-Time Algorithms**, similar to real-time operating system, are used to address the performance concerns. When building components in a layered architecture, efficiency of interactions among the various layers is of prime importance. These algorithms assure that the entire processing of the security broker takes place in real time and the users/services can invoke these security services at the application layer. These real-time features are implemented at each layer.

5.5. Other Features

5.5.1. Security Policy

We propose a layered security policy in our proposed security architecture. The salient features of this policy include:

9. Flexible policy-based access control mechanisms
10. Inter-domain access control policies
11. Secure group communication
12. Delegation mechanisms to support scalability to large numbers of resources and users

The security policy consists of two distinguished parts: Global Security Policy (P_G) and Local Security Policy (P_L). The Local Security Policy layers are *application* policy, *access*

control policy, *data integrity* policy, *authentication* policy and *encryption* policy. The Global Security Policy defines general security policy and provides the abstraction (virtualization) of all Local Security Policies.

5.5.2. Reconfigurability

We have also explored the reconfigurability/adaptability of the security services to provide security support for the heterogeneous systems. We propose that the security services should be capable of reconfiguring themselves if some new node is introduced or to react to recover any system problem. It is achieved through the employment of a dynamically reconfigurable component-based architecture. This architecture allows nodes to dynamically negotiate the security services, protocols, and cryptographic support needed. Our motivation here is to enable configurability and reconfigurability of core security services functionality without having to make any particular assumptions about the underlying distributed architecture.

This feature has following advantages over the classical security architectures:

7. It makes the security architecture adaptable to such heterogeneous environments where the ultimate composition of the system resources is unknown in the beginning. Hence it supports the dynamic addition and suppression of resources from the overall system at any time instant.
8. It makes the security architecture resilient and hence assures survivability of the overall system. Reconfigurability makes the system to regain its original security configurations after the attack scenario is over and therefore it improves the quality of service of the entire system.
9. It enables the system to cope up with the frequent technology changes so that new devices and resources can be easily integrated into the existing systems without changing the core architecture and without plunging the operation quality of service and performance. For example, if a user submits a request for data analysis to a grid, the grid should perform the task in a timely manner, in a secure environment to avoid tampering, and with all necessary accuracy. (though, this quality of service was not a priority in the initial generations of the grid, where just getting it all to work first was the stated goal).

5.5.3. How the Proposed Architecture Responds the Security Requirements

Figure-3 shows various entities of a user and his target domains including auxiliary pluggable security services. ***User domain*** consists of *user*, *local resources* (both computational and storage), an *authentication server* (that authenticates the user and delivers credentials), and an *attribute server* (that delivers user's privilege attributes and sends the assertions with service requests). ***Target domain*** consists of *target resources* (both computational and storage), an *authorization server* (that validates the certificates), a local CA, and *Access policy* (that makes authorization decisions).

Any interaction between the user domain and his target domain requires some intermediary architecture that can convert the assertions into a form understood by the target domain – for example conversion of authentication credentials (e.g. Kerberos ticket) into a credential form that target domain can work with (e.g. X.509 certificates). This intermediary architecture can also offer a number of pluggable security services to the user. These services are discussed in detail in the second part of this document. Moreover, this intermediary architecture can honor a set of policies when forwarding the request (including the mapping rules and delegation policies. This mapping server, pluggable security services and the various security units of user and his target domains are grouped together in figure-3 as ***Security Services*** that virtualizes the security dialogues between these domains.

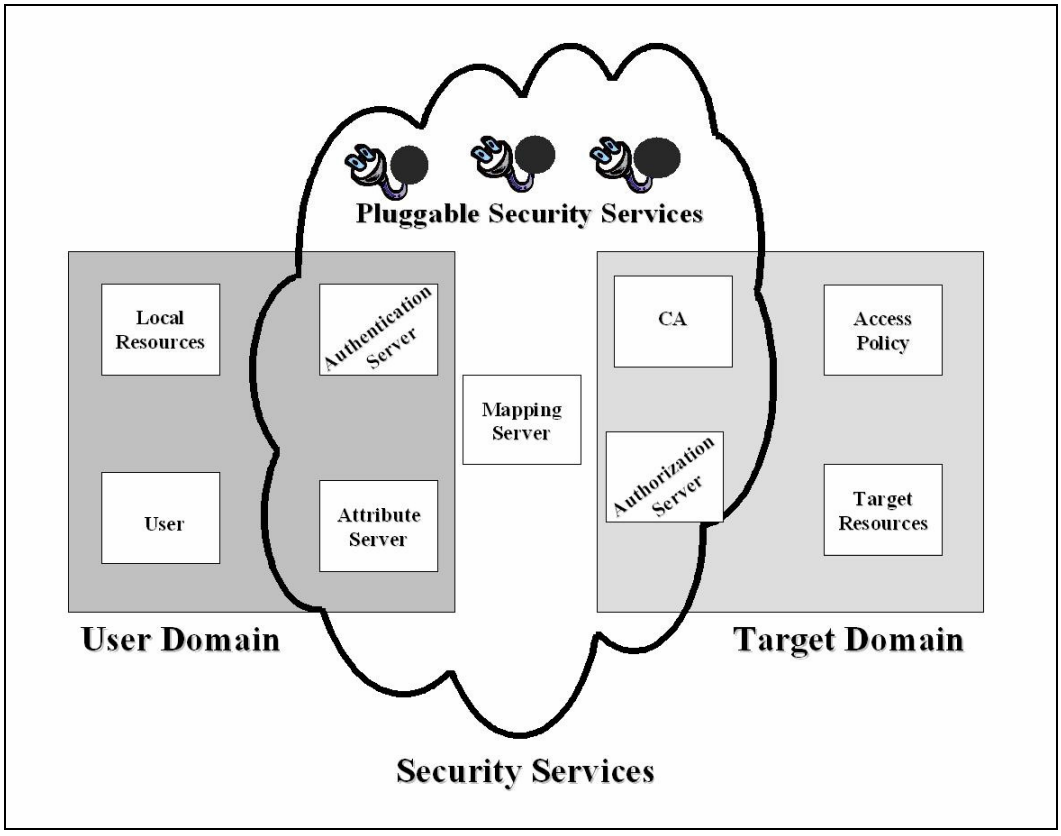

Figure 5.12: Virtual security interaction between a user and his target domains

In such architecture

- New users or groups may be introduced quickly (scalability) as the security services layer harmonizes (virtualizes) the diverse security mechanisms of participating nodes and there is no restriction of specific communication or security requirement.
- The handling of privileges provided to a group or individual can be easily managed as it employs role based access control (RBAC).
- Isolation of applications layer from the core security architecture layer (figure 5.9) enhances the protection of the private data including authentication data.
- Agreed security features could be implemented by making corresponding adjustments in the security broker layer (figure 5.9).
- The intermediary architecture (figure 5.12) could be employed to delegate actions; however, there is a need to shun the cloning of credentials as they could be exploited.
- The attribute server (figure 5.12) could be employed to place limits on the overall amount of resources consumed by particular user or group. These limits are generally defined in the access policy of the target domain (figure 5.12).
- The confidence of the resource providers can be gained by offering them a number of pluggable security services. They can easily incorporate additional security features that assure them that their resources could neither be exploited nor be misused; and in the case of any misuse a chain of accountability could be established.

5.5.4. Performance Evaluations

Validation of the VIPSEC is the focal point of its applicability. We have developed a pervasive grid prototype to validate our propositions. An example scenario of the system is described in this section:

All the teachers and students of our department are supposed to use their PDAs to gain access to the pedagogic resources. Wireless access points are provided in every room of the department. These access points are also used to determine the context of the users. In the library, students can read e-books but can not read their examination paper; whereas in the examination hall, from 9 am to noon, the students can read the examination paper, write the answers file, but can not read books. The teachers can read and write the examination paper from both library and from the exam hall. A PDA is placed in the quarantine zone if its user:

1. tries more than three unsuccessful log-in attempts as student or more than two unsuccessful log-in attempts as teacher, as he/she may be a potential intruder;
2. is using too much bandwidth, as he/she may be trying to cause the Denial of Service (DoS) attack;
3. is seeking unauthorized privileges.

Placement in a quarantine zone implies that:

1. other users are informed of his/her presence, as a troublemaker;
2. he/she is asked to behave normally otherwise he/she will be expelled;
3. after some time Δt it is evaluated whether to clear him/her out the quarantine zone or disconnect him/her from the system. This decision will be based on the close observation of his/her activities during the quarantine period Δt.

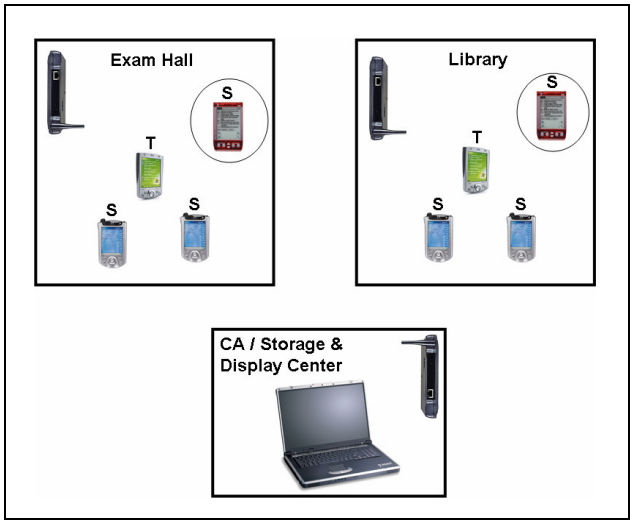

Fig. 5.13: Prototype setup

As shown in figure 5.13, two different Wi-Fi access points at our department building are used to model library and exam hall. PDAs with embedded Wi-Fi card are used to model students (S), teacher (T) and potential attacker (encircled). One PC is used (to be connected from the third Wi-Fi access point) to act as the CA. The overall happening of the system is displayed on its screen including the log of the various actions taken by these PDAs and the time taken by each operation.

We consider a bunch of heterogeneous nodes containing some malicious nodes. These nodes are considered mutually trusted until an attack is detected. A malicious node regularly tries to attack the other nodes. Each attack has a probability p of success. This probability depends on the target node type. A successful attack turns the victim node into a new attacking node for the others. However, in the contrary case the attacker is blocked in its firewall and an alert concerning this node is transmitted in the system.

The results obtained from this grid setup show that there is no considerable overhead on the overall performance of the system due to the consideration of context and state of the mobile nodes. Figure 14 shows few screen shots of this experimental set-up:

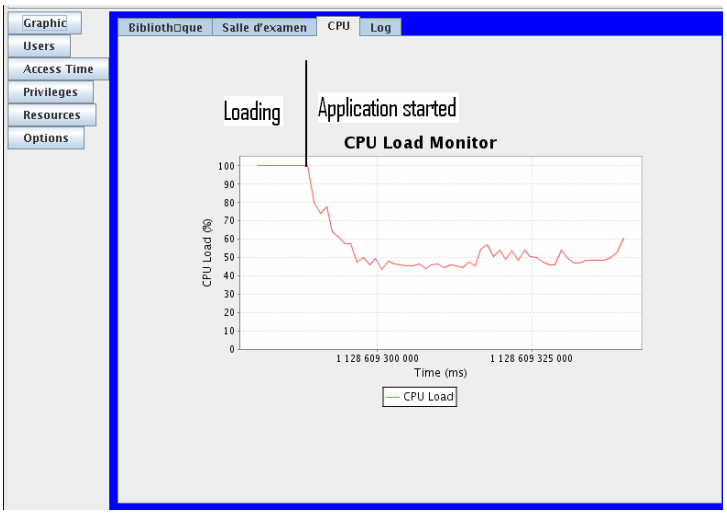

Figure 5.14 (a) : CPU Performance

The performance of the entire system is of prime consideration. We carried out a study to observe the impact of the dynamic consideration of the access privileges. Figure 5.14(a) provides a screen shot of the CPU performance graph.

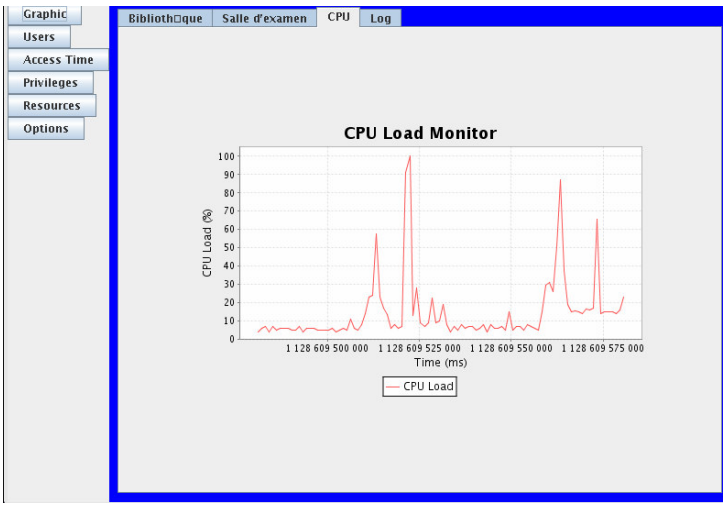

Figure 5.14 (b) : CPU Load Monitor

Figure 5.14(b) shows a screen shot of the CPU load monitoring. It shows the normal behavior of CPU even during the execution of dynamic privileges management.

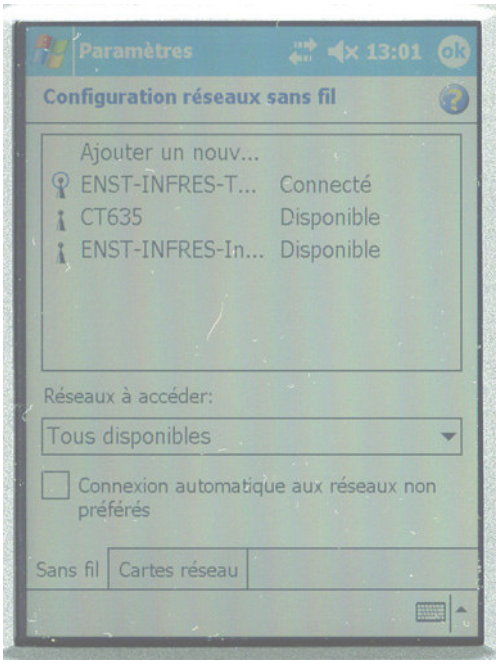

Figure 5.14 (c) : A mobile node

Figure 5.14(c) shows a screen shot of the network configuration settings of a mobile node.

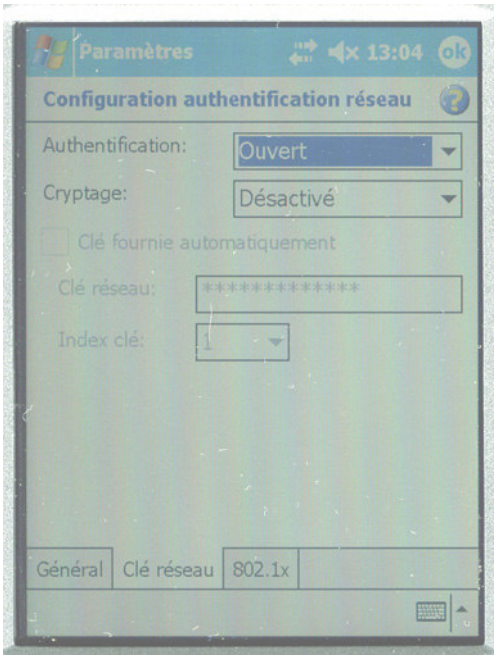

Figure 5.14 (d) : Selection of Encrypted communication

As shown in figure 5.14(d), a mobile user can activate or deactivate the encrypted communications mode. One of the objectives of providing this feature was to study the impact of encryption on the overall performance of the system.

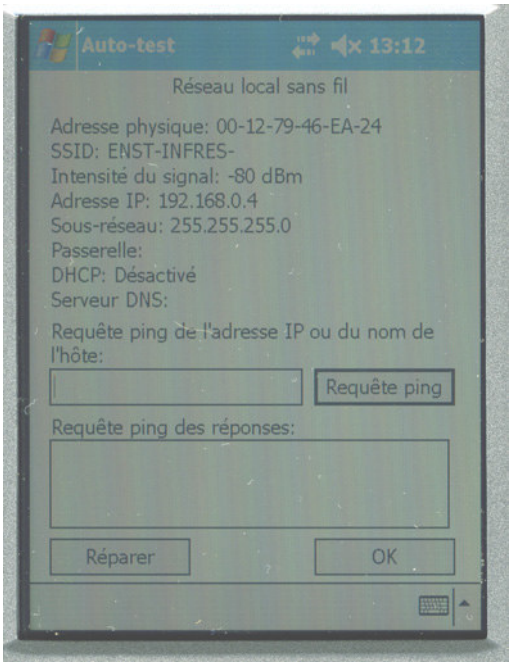

Figure 5.14 (e) : Connection to the Grid

Figure 5.14(e) provides the screen shot of the mobile node which is in the process of establishing connection with the Grid.

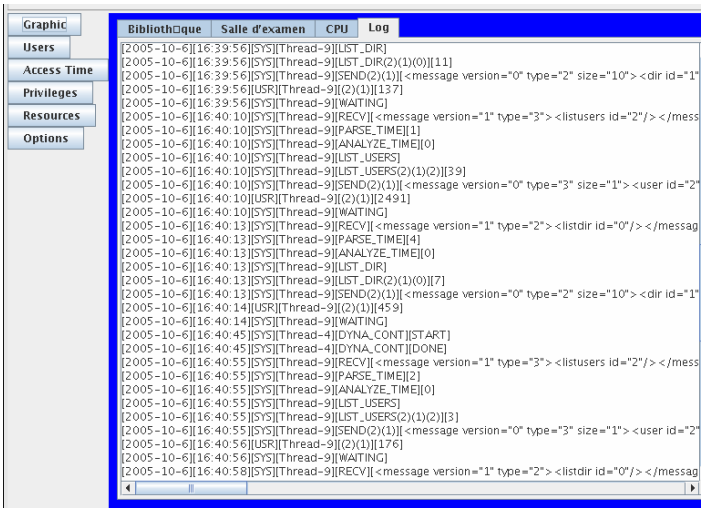

Figure 5.14 (f) : Log file

Figure 5.14 shows a glimpse of the log file that contains all the actions took place with the entire Grid system.

Figure 5.14 (g) : Modifications of the Security parameters in the real time

Figure 5.14(g) shows the screen shot of the options of modifying the security parameters in the real time.

5.6. Trust Management

Establishment of security services and trust relationships are the most desirable features for large scale open heterogeneous systems. They are in need of a consistent security architecture that is sufficiently efficient and scalable. Security of large numbers of users and resources, without relying on a centralized and integrated security infrastructure, should have to move towards the concepts of spontaneous recognition and dynamic trust establishment.

The problem addressed in this section is *how different nodes can trust unknown infrastructure with their private data and vice versa i.e. how a computing infrastructure can trust a node which is seeking access to its resources.* In particular, our trust establishment approach gives greater protection to data and information that may need to be revealed to a third party's computing platform during the process of a transaction. In such computing system, certain number of mutually distrustful participants with varying degree of priori relationships (or perhaps not at all) may want to share resources in order to perform some tasks. In this situation we envisage the trust reflection will provide a means for these participants to evolve the trust relationships quickly and accurately via limited iterations. This will provide an opportunity for the collaborating participants to either achieve full cooperation or to remove untrustworthy participants depending on the result of trust evaluation.

The collaborating members are from different security domains, they may not adhere to the same security policy. The decentralized nature of administration makes it difficult to establish and propagate trust. A distributed trust evaluation scheme is therefore required for these environments. Trust can be based on a history of interactions where credentials can be shown to demonstrate some previous relationships. Such mechanism is called *history-based trust establishment* [99]. However, a challenging situation arises if there is no trust among parties and there is no mechanism to build some trust based on a history of previous interactions. New solutions addressing these issues are required both for the protection of users, including privacy measures, and for controlling the access to valuable resources like commercial services.

We propose dynamic distribution of trust. Our proposed scheme provides a mechanism for delegation of trust and continuous monitoring of the changes to the level of trust of each node. It has the advantage of decentralized hierarchical administration, scalability of certificate issuing capacity and the flexibility of delegation. Since the open services are not limited to a specific range of domains and organizations, we propose a distributed, flexible (adaptable to different security domains) and general-purpose trust management for

establishing a trust relationship among entities, which have no previous interaction, to provide a scalable and decentralized access-control mechanism over the Internet.

To establish trust among the different nodes, we show that instead of having a single value representing the *trustworthiness of a node*, the value should be broken into separate attributes. These attributes are presented to exemplify how to break trust into separate *confidences*. Each attribute represents a *confidence,* and each *confidence* represents a characteristic of a node from which trust can be synthesized. There are varying forms of trust. We can *trust* a node to be accurate (this is important for data integrity). We can *trust* a node to complete tasks reliably. We can *trust* nodes to return data quickly, or always in a guaranteed time, so on and so forth. Like people *trust* physicians for medical advice and stock brokers for financial advice, attributes should be viewed as foundational characteristics used to build particular types of *trust*.

These attributes form a virtual plane to link the resources, users (individuals and services), and the applications. This relationship signifies that there is not a fix form of trust among the various entities. Using a virtual and extensible basis for synthesizing varying types of trust allows for the greatest flexibility from one entity to the other. Flexibility is essential as anything too rigid cannot be easily adopted in a grid environment.

From the functional point of view, these attribute certificates will be used in compliment with identity certificates provided by the existing infrastructure [28, 100]. Where the identity certificates are used to verify the identity of an entity in a highly anonymous environment (e.g. the Internet), the attribute certificates will be used to determine the trustworthiness of an entity in an uncertain environment (such as the pervasive grid).

Our proposed model comprises of:

1. definition of trust relationships between two nodes when there exist:
    a. direct trust relationships within a single domain – although there exist a unique CA or authorization policy, still an invalid proxy certificate generated by a malicious host can run a faked sub-process to accomplish a malicious task. So a node should estimate the *trustworthiness* of the node it is going to interact. Our trust model handles this scenario by using the centralized credentials (X509 or Kerberos) architecture to determine the trust values of the individual nodes by maintaining a trust table of the domain.
    b. indirect trust relationships across multiple domains – crossing the domains further complicate the problem described in part 1a. In this scenario, for a successful interaction, a node has to trust all the intermediate hosts along the path that it has traversed before arriving to the second node (with which it will interact). Our trust model evaluates the trust degree along the whole path keeping in mind that the security policies in different domains as well as in VO may be different. Thus, the trust relationship between a set of nodes is establish.
2. dynamic establishment of trust relationships (using intermediaries in a distributed means) where any node can join and leave anytime and anywhere. As the nodes may belong to different security domains, they may not share the same security policy. The decentralized structure makes it difficult to establish trust in the grid. Our trust model employs a distributed trust evaluation scheme to fit the large scale heterogeneous distributed environments and also supports the basis for satisfying the security requirements to achieve single sign-on and delegation.

5.7. Salient Features of the Proposed Architecture

5.7.1. Security of the *Security Architecture*

The security of the security architecture determines if it can meet the security demands of an active and determined attack. A secure security system can be viewed as a classical security system that possesses supplementary capabilities, such as the capability of intelligently detecting certain faults, the capability of learning from detected faults, the

capability of taking into account the knowledge issued by security experts without compromising the dependability of the whole security system.

We have given special importance to the security of our proposed security architecture. In this section, we mention the various steps taken to assure the security of the security architecture.

Basic Design and Documentation

We have provided adequate documentation that gives sufficient information about the various components of the security architecture. The boundaries of various modules are clearly demarcated. The documentation also includes block diagrams showing all of the major components and their interconnections. One of the most important parts of the documentation though is the identification of any module parts that were excluded from the security requirements and why. Thus, if questioned it can be plainly seen why these parts were not viewed as a security risk.

Module Interfaces

One of the most important requirements in the secure architectures is the restriction placed on the modules interface. The security gaps are introduced in any secure path going through one or more middleboxes that need to perform some processing on passing data packets. These middleboxes include Network Address Translation (NAT) gateways, packet or content filters, proxy firewalls, and Wireless Application Protocol (WAP) gateways. We have separated the interfaces from each other so as to limit the chances that they could cause a breaking point - a place that is more vulnerable to attack.

Authorized Roles and Services

In our proposed model, modules are designed so that they can only perform certain tasks for certain people depending on the privileges that they are allowed. That is, they are designed such that they support authorized roles and the corresponding services that can be performed within those roles. Also, if a module can support multiple simultaneous operators, then the module should internally maintain the separation of the roles and services performed by each operator. This means that operators using the software at the same time should still be restricted to the access that they would have had supposing they were the only one using the module. Furthermore, depending on the security level, all modules are required to make use of access control mechanisms to authenticate an operator accessing the module and to verify that the operator is authorized to perform the desired roles and to perform the desired services within that role.

Physical Security

Physical security of the IT components is an important topic. The overall system should be made attacks resistant. It should be designed to make use of physical security system in order to limit unauthorized physical access to the various system components and to discourage their unauthorized use or unauthorized modification. The idea is that the entire physical setup, including all hardware, firmware, software and data should be protected.

Moreover, the physical security systems are created such that unauthorized attempts at access, use or modify will either have a very good chance of being detected after the attempt has been made by leaving some sort of visible signs, or even better have a very good chance of being detected during the attempt so that appropriate actions can be taken by the module to protect itself.

Software Security

We have used pre and post conditions for each software module, software function and software procedure. The source code listing explains with comments that clearly identify the pre-conditions necessary upon entry into the module, function or procedure in order for it to

execute correctly, and the post-conditions expected to be true when execution of the module, function or procedure is finished.

Operating System Security

The operating system security requirements are necessary when a general-purpose computer running security software as well as some untrusted user-supplied software. We propose that all software be installed only as executable code in order to dissuade analysis and modification by users. The idea is that if the software is only in its executable format, then it will be more difficult for it to be tampered with or analyzed. We also recommend that all security software, encryption keys and other critical security parameters, should be handled by the operating system that provides some sort of controlled access protection. Finally, the operating system needs to provide the means by which to specify a set of users who are authorized to enter encryption keys and other critical security parameters. That way not everybody has access to critical sections and it further limits the chances that there might be an information leak.

Self Testing

A system must be able to perform self-tests in order to ensure that it is functioning properly. We specify that certain self-tests need to be performed whenever the system is powered up. It also states that there are other self-tests that can be performed under various conditions, typically when a particular function or operation is performed. Additionally, the system can optionally perform other self-tests in addition to the tests specified here if they so choose.

Whenever a module fails a self-test, it must enter an error state and output an error indicator via the status interface. That is, the module must notify the user, security officer, or maintenance personnel that some sort of internal error or malfunction has occurred. Essentially then the module goes into a lockdown state and will not perform any operations while in the error state and no data shall be output via the data output interface while the error condition exists.

5.7.2. Security Negotiations Features

Our proposed security infrastructure supports advance negotiation and establishment of a secure session between the endpoints that allows the endpoint to negotiate security requirements. Further, it features support for negotiating and establishing end-to-end and/or hop-to-hop security associations. Such a security infrastructure has broader applicability to general networked environments.

In our proposed security model, the security negotiations are carried out by the Security broker that mediates between applications (more precisely the distributed applications) and the security services. The security broker has a security services handler, which is employed to absorb the heterogeneity of the underlying security services and to provide a homogeneous interface to the upper layer. The security broker is discussed in details in section 5.4.

5.7.3. Security Bootstrapping

It is important to setup security relations in an environment of networked devices without requiring the user to be an expert and without relying on central infrastructure. A security relation between two devices can be a common shared cryptographic key, or an authentic copy of a public key. It allows the two entities to exchange messages in an authentic and/or confidential way. However, security relations need to be bootstrapped, that is, keys have to be generated, distributed, and authenticated. These operations cannot be expected to be done by an average user, in particular not on devices with a limited user interface.

Because it is not realistic and necessary to predefine security relations between all devices, other alternatives are deemed necessary. The common way is to use trust to reduce

the number of initially required security relations. A trusted entity T can, for example, create a bilateral key for two devices A and B if both, A and B have a bilateral key with T and trust T to generate random keys and to forward only authentic information. Unfortunately, it is not practical to find one single entity T that is trusted by all other entities. Using a web of trusted entities would solve at least this concern, but is a delicate process because two devices that want to build a relation must find a path through the web on which they trust every single entity.

Bootstrapping is a process to build security relations between devices that want to interact with one another without relying on predefined relations, central services (e.g., an administrator), or the availability of dedicated entities such as a trusted third party.

The Resurrecting Duckling Policy Model

Ross Anderson and Frank Stajano were the first who recognized the importance for security relations between networked devices. In their Resurrecting Duckling Policy Model [101, 102], they propose two basic elements:

- Secure Transient Association: exchange of a shared secret during physical contact (pairing) representing a master-slave relation between the devices.
- Default policy: the master device can access all services of the slave device; no other device is allowed to use services. One of the services accepts policy updates.

While the relation between devices is a static master-slave relation in the original paper [101], an extension to peer-to-peer relations is presented in [102] by describing relations to other devices in the security policy.

The pair-wise master-slave relations between devices introduce dependencies between devices that limit the usability and are prone to loss of devices. Although peer-to-peer relations partially address this shortcoming, the model does not suggest how the credentials (i.e., keys and certificates) to represent these relations could be described in the policy in an authentic way. Further, the lifecycle of a security association is rather static: Delegation and exception handling (i.e., loss of devices) are not supported.

Our Approach

Our approach to address the shortcomings of the Resurrecting Duckling Policy Model includes the ownership model that builds real peer-to-peer security relations between all devices owned by the same user and thereby strictly defines what devices those are trusted. The security policy defines relations to other devices, assigns rights to security relations, and supports authentic key exchange.

When two devices from the same user get paired, one device creates a certificate for the other device by signing the identifier (i.e., the public device key) of the other device. The resulting certificate chain leads towards one of the user's devices and is used to recognize other devices owned by the same user (i.e., siblings). The default security policy of a device defines the same rights as the Resurrecting Duckling Policy Model.

The advantage of this approach is that a new device has to be paired only with one device to build up authentic relations to all other devices of that user. These relations provides redundancy to cope in situations where devices get lost or delegated, because a device is not dependent on a dedicated other device such as in master-slave models.

The devices are described with credentials such as keys (i.e., the device identifier) or certificates for expressing roles that a device assumes. Instead of configuring these credentials in the security policy, it allows to set a wildcard specifying the conditions to accept the credential of the next device that gets paired with the target device. The involved devices thus exchange their credentials themselves without requiring the user to cope with cryptographic material.

Privacy gets increasingly important in the context of networked devices because personal and private information and resources are exposed to an open and unknown environment.

Besides of the information stored on the devices itself, also meta information such as the user's identity and location needs to be protected. The use of pseudonyms and secure service discovery protocols are emphasized in our work to provide the presence of devices only on a need-to-know basis.

Most of the devices that we intend to connect will be personal and private devices of our daily life. Security aspects are therefore important. However, the most important aspect is the ease-of-use because the user cannot be expected to configure anything. However, we consider the security bootstrapping as a first step towards the ambitious goal of self-configuration in dynamic networks.

5.7.4. Modifications Proposed in the Existing Systems

Extension of Open Grid Services Architecture (OGSA)

OGSA security model casts security functions as OGSA services [103]. This strategy allows well-defined protocols and interfaces to be defined for these services and permits an application to outsource security functionality by using a security service with a particular implementation to fit its current need. We extend the concept of *security as services* to *security as pluggable services*. This extension permits the evolution of security infrastructure with less impact on the resource management functionalities, which are still on the verge of evolution. Moreover, it permits the users and resource providers to configure the security architecture according to their requirements and satisfaction level.

We add some handler modules into OGSA container to extend its functionalities. First the service request comes into the container through service interface, and then passes the security services handlers one after another until it invokes the implementation of that security service. After service executing, the response also may tunnel through some other security services handlers before it gets to client side. A handler may communicate with a corresponding Grid Monitoring and Managing Services (GMMS) during the procedure and send monitoring information to that GMMS according to Web Services Level Agreement (WSLA).

The handler modules are developed as plug-ins for OGSA container, which means those handlers can be inserted into or removed from the container at any moment. That plug-in mechanism brings high flexibility. Firstly, we can customize needed handlers in the container to fit different scenarios. Secondly, we can specify different handlers in service deployment stage in order to do different monitoring functions. The handler may be bound to an existing GMMS in Grid which can do workload analysis.

Extension of Grid Security Infrastructure (GSI)

The Grid Security Infrastructure (GSI) [28] does not attempt to discover middleboxes and negotiate security with them. As a result, security gaps could surface, particularly in cases where some grid resources and nodes exist in a local network behind a firewall. Further, the adaptability of GSI is limited making it hard to port it to lightweight devices with limited capabilities.

We propose that the security architecture deployed must be able to adapt to environments with varying conditions that incorporates greater flexibility, adaptability, and customizability. When it comes to security, one size does not fit all. Hence, the security architecture deployed must be able to adapt to environments with varying conditions. Further, with many different security technologies surfacing and being deployed, the assumption that a particular security mechanism will eventually prevail is flawed. For that reason, it is necessary to support multiple security mechanisms and negotiate security requirements. We also aim to reduce or eliminate security gaps.

Chapter 6

Assessment of Security Functionalities

6.1. Common Criteria (CC) [104]

The Common Criteria for Information Technology security evaluation is a relatively new program, which seeks to establish an internationally agreed-upon language for specifying security functionality, as well as an evaluation methodology to assess the strength of security implementations. We have used Common Criteria (CC) version 2.0 for the evaluation of our proposed security model. CC version 2.0 has following three sections: General Model (section one), Security Functional Requirements (section two), and Security Assurance Requirements (section three).

The CC general audience, groups who would apply CC standards, is comprised of IT system or product consumers, developers, and evaluators. The three CC sections provide guidance on how CC establishes baseline security requirements for buying, developing, or evaluating an IT system or product.

The Common Criteria Evaluation and Validation Scheme (CCEVS) Security Framework [105] is shown in figure 6.1. The first step of evaluating of a system or application using common criteria methodology is to identify a Target of Evaluation (TOE.) The TOE is a system, application, or IT product that is selected to be evaluated according to CC standards. The second step is to develop a set of Security Targets (ST). The ST is the set of criteria to be applied for the evaluation of the TOE. For specific technologies or IT products, previously established protection profiles may be used as the ST criteria. With each step of the security framework, the CC evaluation process requires increasingly detailed information regarding the application or system security profile. The resulting product of progressing through the CC Security Framework steps is an IT product or system that meets a baseline set of security criteria and/or processes that institute fundamental security techniques. Specific security mechanisms or techniques for IT products and technology are addressed through the Common Criteria Protection Profiles.

We have prepared a protection profile of Health Grid. The complete PP is provided in appendix B. However, its concise account and salient features are provided in this section.

6.1.1. Health Grid [106]

Health grids are Grid infrastructures comprising applications, services or middleware components that deal with the specific problems arising in the processing of medical data. Resources in health grids are databases, computing power, medical expertise and even medical devices. The vision of the health grid is to create an environment where information at the five levels (molecule, cell, tissue, individual, population) can be associated to provide individualized healthcare.

6.1.2. Security Architecture for Health Grid

Security is one of the most important features of a heath grid. Personal data (any piece of information in which its owner can be identified, either directly or in combination with information that is available or can be available) is confidential, so access to the information

must be performed only by authorized and authenticated persons, and data must be encrypted to guarantee its confidentiality and integrity. We have used our proposed security architecture for Health Grid [8] to produce protection profile.

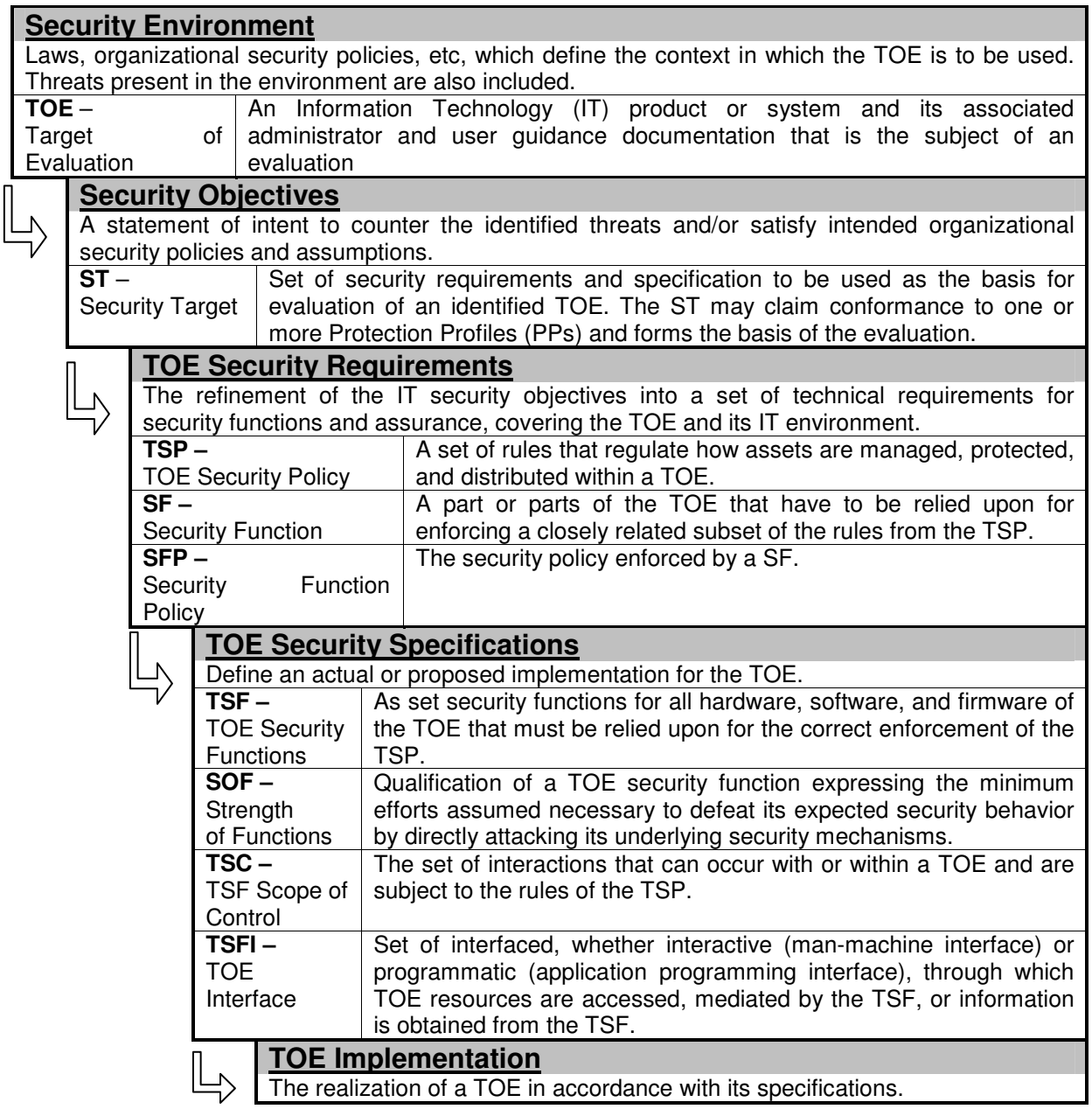

Figure 6.1: Common Criteria Evaluation and Validation Scheme (CCEVS) Security Framework

6.1.3. Protection Profile

Protection Profile (PP): The intent of this Protection Profile is to specify functional and assurance requirements applicable to Health Grid. Security requirements are viewed from the various angles including users, resource providers, and developers' views.

Target of Evaluation (TOE): This section describes the TOE as an aid to the understanding of its security requirements and addresses the product type, the intended usage and the

general IT features of the TOE. The TOE is the *Health Grid*, independent of the application(s) being run over it.

TOE Security Environment: This section describes the security aspects of the environment in which the TOE is intended to be used and addresses the description of the assumptions, the threats and the organizational security policies.
Assets are security relevant elements of the TOE that are classified as: data and information across the TOE, applications running over the TOE, computing resources constituting the TOE, storage repositories of the TOE, communication links (wired and/or wireless) within the TOE.
Assumptions include a small community of active users (A.ActiveUsers), a large community of public users (A.PublicUsers), and a provision of periodic revision of the security architecture (A.TechnologyUpdates).
Threats are divided in the two broad categories: Threats to Information (T.I) and Threats to Resources (T.R).

Security Objectives: This section defines the security objectives for the TOE (O.T) and for its environment (O.E) with an emphasis on the use of state of art technologies to achieve these IT security objectives.

TOE Security Requirements: This section defines the functional and assurance security requirements that the TOE and the supporting evidence for its evaluation need to satisfy in order to meet the security objectives for the TOE.
TOE Security Functional Requirements define the functional requirements for the TOE using functional requirements components drawn from the Common Criteria part 2. The minimum strength of function (SOF) level for the TOE security requirements is *high* – SOFhigh. SOF-high is a level of the TOE strength of function where analysis shows that the function provides adequate protection against deliberately planned or organized breach of TOE security by attackers possessing a high attack potential.
TOE Security Assurance Requirements define the assurance requirements for the TOE using functional requirements components drawn from the Common Criteria part 3. The evaluation assurance level (EAL) is 4 – EAL4. EAL4 provides assurance by an analysis of the security functions, using a functional and complete interface specification, guidance documentation, the high-level and low-level design of the TOE, and a subset of the implementation, to understand the security behavior.

Security Rationale: This section presents the evidence used in the PP evaluation. This evidence supports the claims that the PP is a complete and cohesive set of requirements and that TOE would provide an effective set of IT security countermeasures within the security environment. **Security Objectives Rationale** demonstrates that the stated security objectives are traceable to all of the aspects identified in the TOE security environment and are suitable to cover them. **Security Requirements Rationale** demonstrates that the set of security requirements (TOE and environment) is suitable to meet and traceable to the security objectives.

6.2. Case Study: Grid Computing Simulations

The range of available grid simulation tools, such as Bricks [107], SimGrid [108], GridSim [22], GangSim [109], OptorSim [110] etc., does not provide any support for the simulations of grid security functions. The deployment of a grid infrastructure without proper simulations of its various defense capabilities will certainly be an invitation to disaster. One can not remove all the vulnerabilities from a design, no matter how analytically good it is, unless the design has undergone a series of 'real-application-specific' tests. In the absence of a proper validation mechanism, security designers risk wasting time and effort implementing safeguards that do not address any realistic threat to the grid. Or, just as dangerously, they

run the risk of concentrating their security measures on one threat while leaving the grid architecture dangerously exposed to others. We have faced the same problem while working on the virtualization of security services for the grid. This situation obliged us to develop a tool to perform grid security simulations – Grid Security Services Simulator (G3S).

6.2.1. G3S: Grid Security Services Simulator

Motivations The prime motivations behind the design and development of G3S was to lay the foundation of a simulations tool for the grid security services as none of the existing grid simulators provides any support for the security functionalities. It was felt imperative to provide a *Graphics User Interface (GUI)* so that even non-computer professionals (such as health grid users) can benefit from this tool by interactively simulating various grid security features (such as secure exchange of documents, attack patterns, etc).

Principles G3S models security functionalities of a grid. The grid nodes may be static or mobile. For the mobile nodes, it also considers the mobility-related security issues such as security gaps. It is designed to support multiple authentication mechanisms such as X.509 certificates and Kerberos tickets. Role-Based Access Control (RBAC) is used for the authorization purposes – work is underway to support the Community Access System (CAS) [100]. G3S supports Bell-LaPadula Model for the assurance of grid data confidentiality and the Watermarking technique is used to assure the integrity of the data flowing across the grid resources. G3S is designed in user-friendly way, so that even a user with a shallow knowledge of security services may equally use it. For example, a user may intend to simulate confidentiality features without knowing that confidentiality requires access control mechanism. G3S automatically invokes the *prerequisite* security services so that a true scenario can be simulated even if its user does not know all of its parameters.

Simulations of different *attack patterns* is provided so that the designers can see if their design can deter the security threats and can survive after the attack. G3S has a mechanism for threats dissemination. If a node attempts to cross its defined privileges then an alert signal about the presence of a malicious node is sent to all the relevant nodes.

Implementation G3S is written in Java. It is lightweight and can be installed and executed from a single PC. An easy-to-use graphics user interface (GUI) is provided. Detailed log of the various operations is maintained to facilitate the auditability. This log file can be accessed by any querying program for swift access to some particular event as it is very difficult to find the trace of certain activity by general observation of a huge audit trail. Nodes have different geometrical shapes (such as circular, square, triangular, etc.) to graphically exhibit their heterogeneous nature. These shapes correspond to the nature of the participating nodes (e.g. their communication mechanisms, their static or mobile nature, etc.) These nodes can be grouped together to form virtual organizations (VOs) at any instant. A number of VOs may be created simultaneously and their transactions are consequently simulated. A different color is allocated for each VO.

Applications G3S can be used to simulate the working and efficiency of a grid security model. The alpha version of G3S can simulate the security services of a grid of maximum 100 nodes; however, the next release will be able to handle 1000 nodes. These nodes are not necessarily the fixed resources – mobile grid nodes can also be simulated with their corresponding mobility features and constraints.

G3S Structure

G3S is composed of five main components (as shown in figure 6.2): *Core*, *DocumentExchange*, *SecurityPolicy*, *TrustManger* and *Attack*.

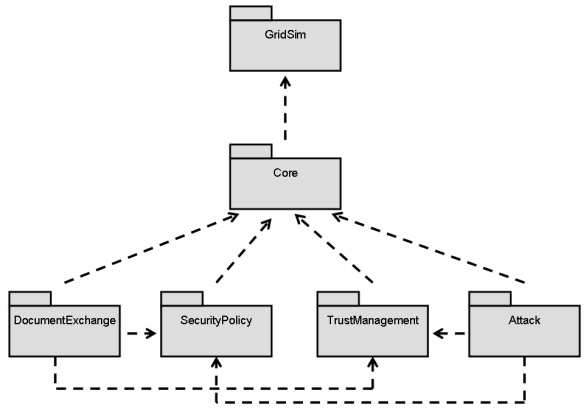

Figure 6.2: G3S main components

Interdependencies of various G3S components are summarized in the table 6.1. The *Core* uses *Security Policy*; the *Document Exchange* uses Core, *Security Policy*, and *Trust Management*; the *Security Policy* is totally independent of other components; the *Trust Management* uses *Core*; and the *Attack* uses *Core*, *Security Policy*, and *Trust Management*.

	Core	**Document Exchange**	**Security Policy**	**Trust Management**	**Attack**
Core		**X**	☑ **R1.3**	**X**	**X**
Document Exchange	☑ **R2.1**		☑ **R2.3**	☑ **R2.4**	**X**
Security Policy	**X**	**X**		**X**	**X**
Trust Management	☑ **R4.1**	**X**	**X**		**X**
Attack	☑ **R5.1**	**X**	☑ **R5.3**	☑ **R5.4**	

☑ **(left) uses (up)**
X (left) doesn't use (up)

Table 6.1: Interdependencies of the G3S components

Various relationships of the table 1 are described below:

R1.3: A VO has one Security Policy (characteristic feature of VOs)

R2.1: A document exchange requires 2 G3SNodes (which exchange the document)

R2.3: A document exchange takes place according to the rules set forth in the security policy of the VO

R2.4: A document exchange needs to check the current trust value of the sending and receiving nodes

R4.1: Trust Management deals with the trust level of each node

R5.1: An attack may result in several victim nodes

R5.3: Success or failure of an attack depends on the strength of the VO security policy

R5.4: If an attack is detected, the trust levels of the attacker and the attacked nodes are changed

These relationships of the various components are not rigid. The existing functions can be easily extended. Likewise, more security functions can also be easily added.

Graphics User Interface (GUI) of G3S

G3S has a graphics window for user interaction. As shown in figure 6.3, buttons for various simulation features (such as adding new users, resources, creation of VOs, security policy configuration, documents exchange, attack pattern, etc.) are provided on the left side of the window beside in the pull-down menus. The central zone is the area where the results of simulations are graphically displayed. A list of different symbols used by the G3S is given on the right side.

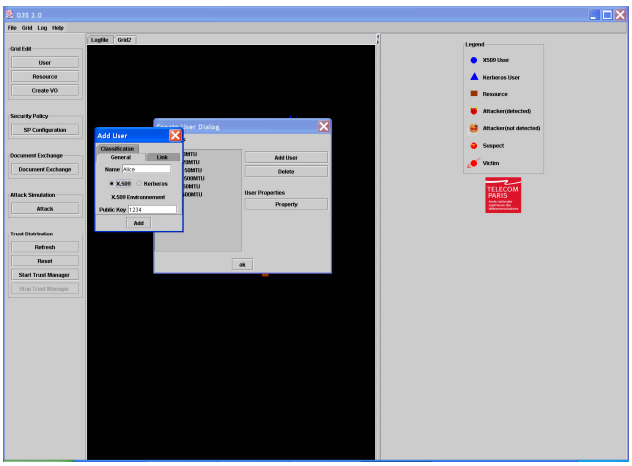

Figure 6.3: G3S Graphics User Interface (GUI)

In the G3S Graphics User Interface (GUI), new users (individuals or groups) can be dynamically introduced (cf. figure 6.3) at any time instant. Apart from the fundamental parameters, such as *name*, *confidentiality level*, etc., specific authentication parameters can be provided after choosing the desired authentication mechanism (Kerberos ticket or X.509 certificate). As soon as a certain authentication mechanism is chosen, G3S GUI automatically asks for the corresponding parameters.

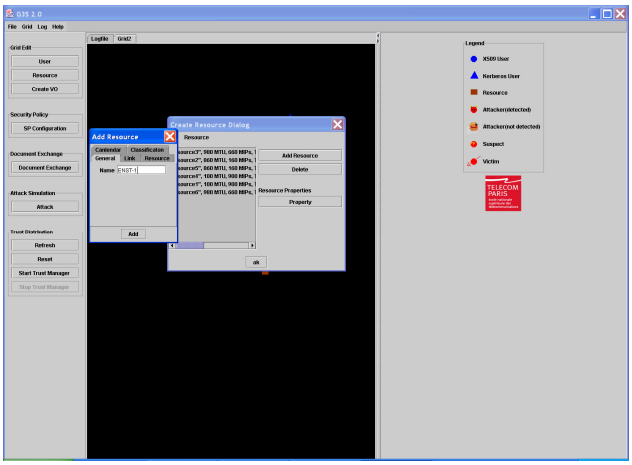

Figure 6.4: Adding new resources

Similarly, new computing resources can be dynamically added (cf. figure 6.4).

New VOs can be created anytime by choosing the participating nodes (users and resources). A unique name is required for each VO (cf. figure 6.5) and the security policy for each VO is configured. A number of VOs may be created simultaneously and their transactions are consequently simulated.

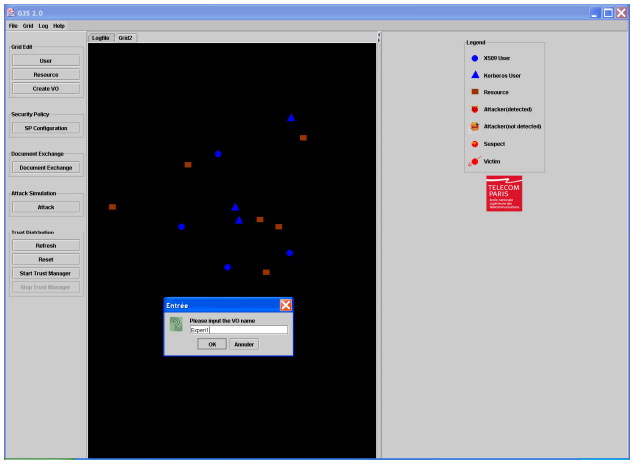

Figure 6.5: Creation of a VO

As shown in figure 6.6, various nodes have different colors and geometrical shapes (such as circular, square, triangular, etc.) to graphically exhibit their heterogeneous nature. These shapes correspond to the nature of the participating nodes (e.g. their communication mechanisms, their mobility mechanism, etc.). A different color is allocated for each VO.

The various nodes of these VOs can collaborate and share resources according to their roles and privileges. All the exchange of data is recorded and the current status of each transaction is graphically displayed. Apart from the collaborations among a VO's nodes, the VOs themselves can collaborate for certain jobs.

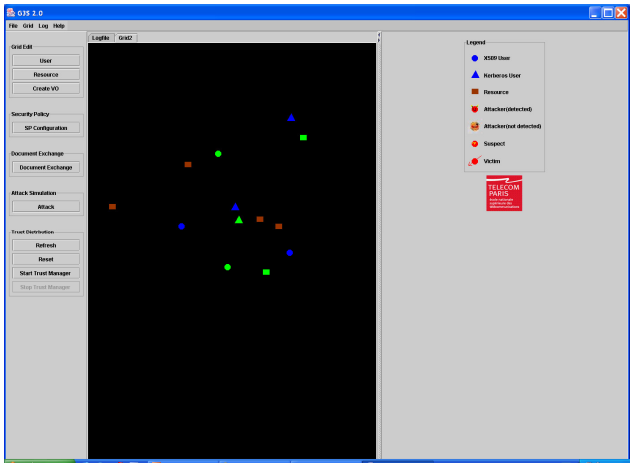

Figure 6.6: G3S entities

Simulations of Attacks Patterns

G3S can also simulate various attack situations such as denial-of-service, man-in-the-middle, relay, and wormhole (cf. figure 6.7). It can also simulate the survivability feature of a security design.

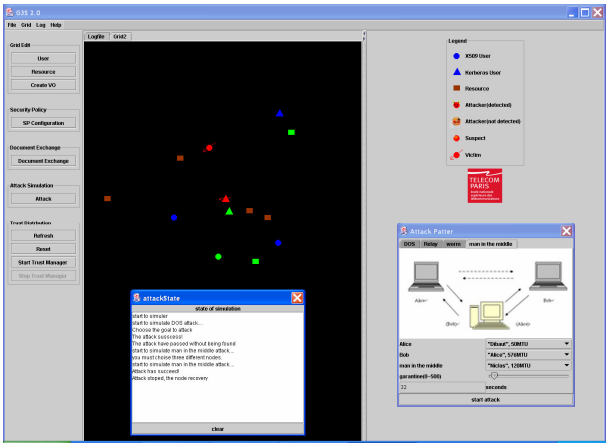

Figure 6.7: Attacks simulations

As an example of attack simulations, we consider a bunch of heterogeneous nodes containing some malicious nodes. These nodes are mutually trusted nodes until an attack is detected. A malicious node regularly tries to attack the other nodes. Each attack has a probability p of success. This probability depends on the target node type. A successful attack turns the victim node into a new attacking node for the others. However, in the contrary case the attacker is blocked in its firewall and an alert concerning this node is transmitted in the system.

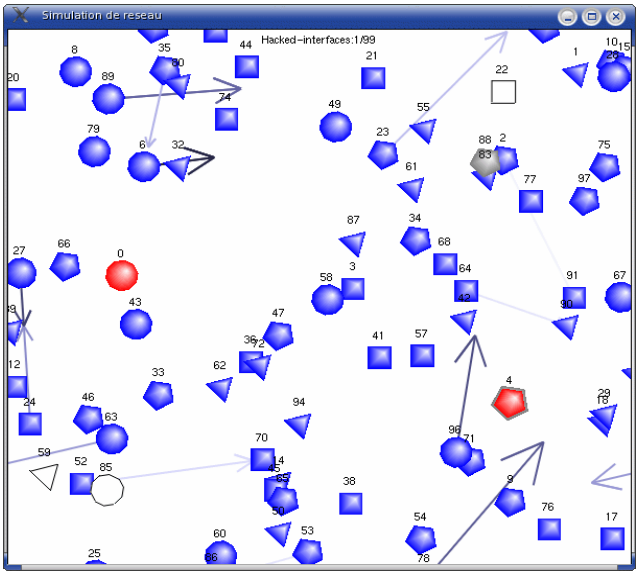

Figure 6.8: Grid nodes

Figure 6.8 shows a graphics display of the Grid nodes. Each node has *xy*-coordinates and its class is determined by its shape (e.g. a triangular shape corresponds to a PDA; a round shape corresponds to a PC, etc.). The color coding used in this scheme is as follows: A node is gray if it does not know about the presence of the malicious node, blue if it is informed of malicious node, and white if it knows all the malicious nodes in the system.

A red halo around a node indicates that it is a victim node (which has become a malicious node itself), blue if the attack was foiled by the security architecture and yellow if the attack failed due to some other reason.

The triangles in the display show the attack propagation whereas the arrows correspond to the distribution of trust among the nodes. The calculation of the distribution of trust is based on a *trust table*. A trust table is shown in figure 6.9. The left entry A is the node that evaluates the entry of the node B from the top side. A color code is employed to quickly determine if there remains a danger of attack in the system: green, if A relies on B, and that A and B are indeed trustworthy; red, if A relies on B, and that B belongs to the attacker or is an attack victim; blue, if A does not rely on B and that B is indeed untrustworthy due to the reasons described in the previous case; white, if A's confidence in B has no importance.

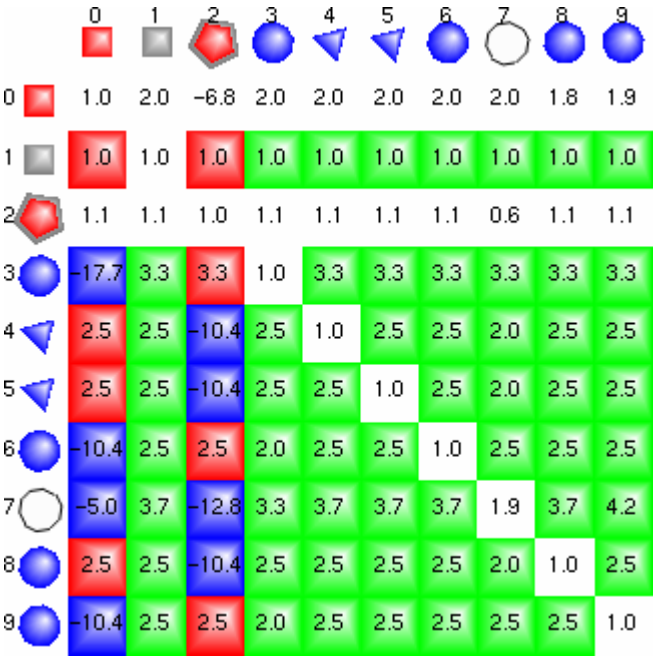

Figure 6.9: Trust table

Figure 6.10 presents the collective defense behavior of the nodes with the described infrastructure of confidence. If the attacker fails in its first attempt, it will be difficult for it to take control of the other nodes. Here node 0 escapes an attack from node 1 and blocks its transmissions. The other nodes are promptly informed of the threat so that they do not remain confident in node 0; and hence the overall system is protected (cf. corresponding values in the trust table).

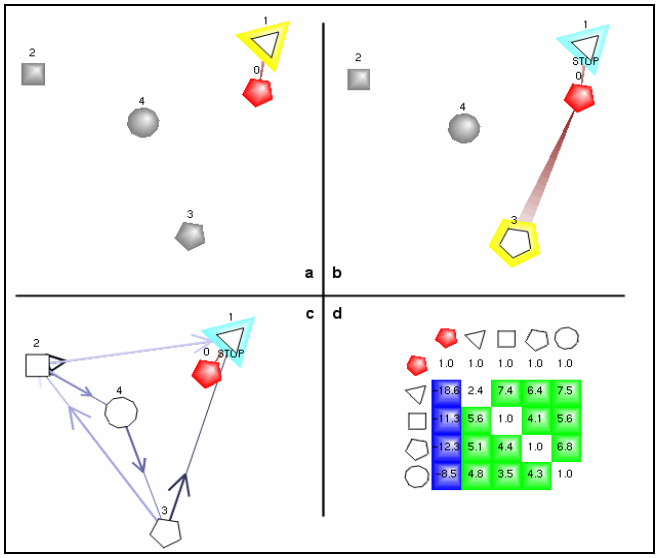

Figure 6.10: Failed attack paradigm

But if the node 0 fell prey to the attack of node 1 (figure 6.11) and then manages to take control of node 3 all the other nodes will soon be affected resulting in the successful endeavor of the attacker.

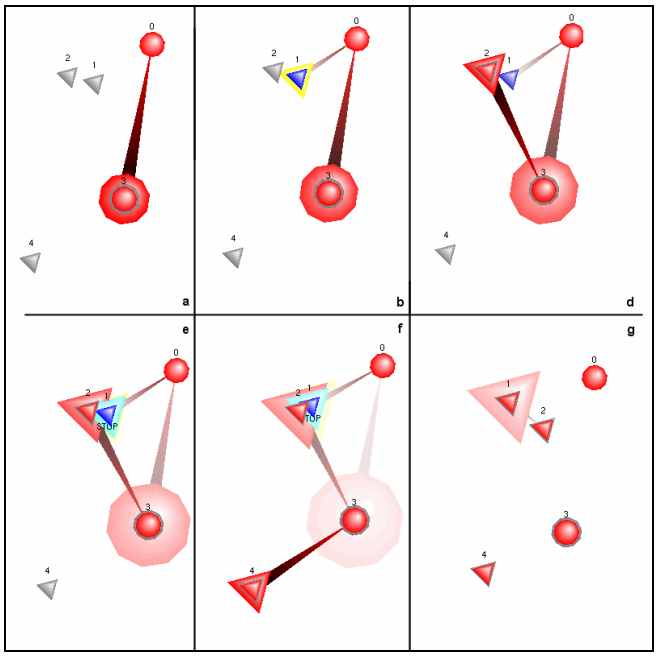

Figure 6.11: Successful attack paradigm

Integration of G3S with GridSim

We interacted with the developers of GridSim during the development phase of the alpha version of G3S so as to give a broader scope to G3S. Moreover, the integration of security services simulations into GridSim will provide a comprehensive simulations tool for the grid community; and hence the users of GridSim can also simulate security functionalities beside scheduling and resource management parameters.

The users and resources defined for the G3S are the *GridUser* and *GridResource* of GridSim, and the actions (such as the exchange of document) are *Gridlets* of GridSim. A *Gridlet* is a package that contains all the information related to the job and its execution management details such as job length expressed in MI (Millions Instruction). For example, the exchange of document is defined as a DocumentGridlet which extends to gridsim.GridSim class.

In the G3S *Core* module, we have defined G3SUser, G3SResource, and G3Slink classes. These classes inherit (*extend*) following GridSim functions:

G3SUser extends gridsim.GridUser
G3SResource extends gridsim.GridResource
G3Slink extends gridsim.net.Link

These classes are harnessed together by a superclass called G3SNode.

There exist some redundancies of code between G3S and GridSim, such as simJava classes. It is in fact required so that G3S can be executed independently without GridSim.

6.2.2. GridSim: A Toolkit for Modeling and Simulation of Grid

The GridSim toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domains distributed computing systems such as clusters and grids. Application schedulers in grid

environment, called resource brokers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. That means, each user has his own private resource broker and hence, it can be targeted to optimize for the requirements and objectives of its owner. Whereas schedulers, managing resources such as clusters in a single administrative domain, have complete control over the policy used for allocation of resources. That means all users need to submit their jobs to the *central* scheduler, which can be targeted to perform global optimization such as higher system utilization and overall user satisfaction depending on resource allocation policy or optimize for high priority users.

Features

Salient features of the GridSim toolkit include the following:

- It allows modeling of heterogeneous types of resources.
- Resources can be modeled operating under space- or time -shared mode.
- Resource capability can be defined (in the form of MIPS as per SPEC benchmark).
- Resources can be located in any time zone.
- Weekends and holidays can be mapped depending on resource's local time to model non-Grid (local) workload.
- Resources can be booked for advance reservation.
- Applications with different parallel application models can be simulated.
- Application tasks can be heterogeneous and they can be CPU or I/O intensive.
- There is no limit on the number of application jobs that can be submitted to a resource.
- Multiple user entities can submit tasks for execution simultaneously in the same resource, which may be time -shared or space-shared. This feature helps in building schedulers that can use different market-driven economic models for selecting services competitively.
- Network speed between resources can be specified.
- It supports simulation of both static and dynamic schedulers.
- Statistics of all or selected operations can be recorded and they can be analyzed using GridSim statistics analysis methods.

System Architecture

A layered and modular architecture is employed for grid simulations to leverage existing technologies and manage them as separate components. A multi-layer architecture and abstraction for the development of GridSim platform and its applications is shown in Figure 9. The first layer is concerned with the scalable Java's interface and the runtime machinery, called JVM (Java Virtual Machine), whose implementation is available for single and multiprocessor systems including clusters. The second layer is concerned with a basic discrete-event infrastructure built using the interfaces provided by the first layer. One of the popular discrete-event infrastructure implementations available in Java is SimJava. Recently a distributed implementation of SimJava is also made available. The third layer is concerned with modeling and simulation of core Grid entities such as resources, information services, and so on; application model, uniform access interface, and primitives application modeling and framework for creating higher level entities. The GridSim toolkit focuses on this layer that simulates system entities using the discrete-event services offered by the lower-level infrastructure. The fourth layer is concerned with the simulation of resource aggregators called grid resource brokers or schedulers. The final layer focuses on application and resource modeling with different scenarios using the services provided by the two lower-level layers for evaluating scheduling and resource management policies, heuristics, and algorithms.

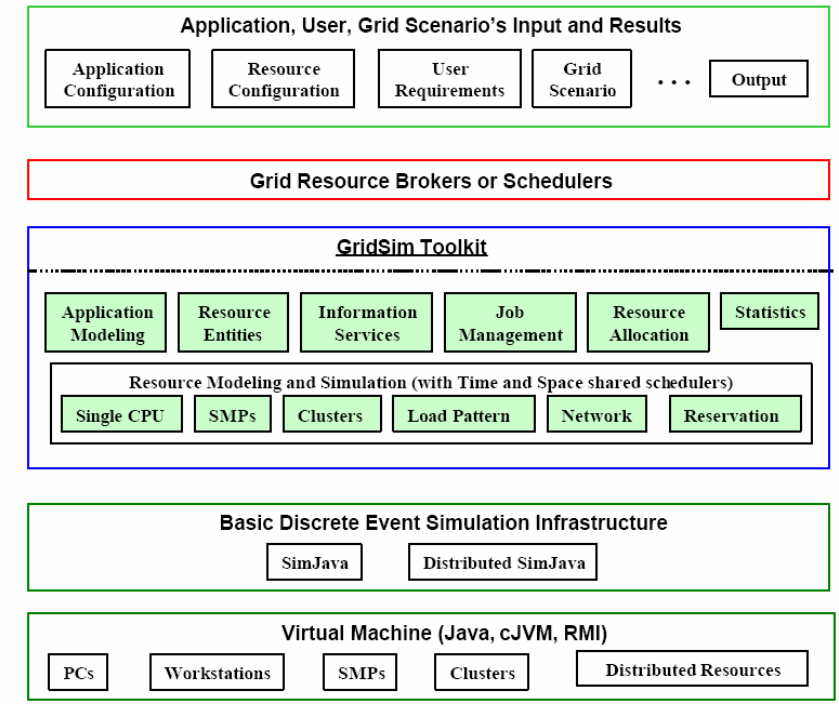

Figure 6.9: A modular architecture for GridSim platform and components

6.2.3. Optimized Network Engineering Tool (OPNET) [111]

In this section, we explore why a commercially available powerful simulation tool OPNET remains inadequate for the simulations of the security services of large scale open heterogeneous distributed systems like Grid. However, before exploring the shortcomings, we present an overview of the architecture and simulation mechanism of OPNET.

OPNET provides a comprehensive development environment for the specification, simulation and performance analysis of communication networks. A large range of communication systems from a single LAN to global satellite networks can be supported. Discrete event simulations are used as the means of analyzing system performance and their behavior. The key features of OPNET are summarized here as:

- **Modeling and Simulation Cycle** OPNET provides powerful tools to assist user to go through three out of the five phases in a design circle (i.e. the building of models, the execution of a simulation and the analysis of the output data).
- **Hierarchical Modeling** OPNET employs a hierarchical structure to modeling. Each level of the hierarchy describes different aspects of the complete model being simulated.
- **Specialized in communication networks** Detailed library models provide support for existing protocols and allow researchers and developers to either modify these existing models or develop new models of their own.
- **Automatic simulation generation** OPNET models can be compiled into executable code. An executable discrete-event simulation can be debugged or simply executed, resulting in output data.

This sophisticated package comes complete with a range of tools which allows developers specify models in great detail, identify the elements of the model of interest, execute the simulation and analyze the generated output data:

- Hierarchical Model Building
  - Network Editor - network topology models
  - Node Editor - data flow models define

- Process Editor - control flow models

- Running Simulations
  - Simulation Tool - define and run simulation
  - Debugging Tool - interact with running simulations

- Analyzing Results
  - Probe Editor –data need to be collected
  - Analysis Tool – statistical results
  - Filter Tool – date processing
  - Animation Viewer – dynamic behavior

Hierarchical Modeling

OPNET provides four tools called editors to develop a representation of a system being modeled. These editors, the Network, Node, Process and Parameter Editors, are organized in a hierarchical fashion, which supports the concept of model level reuse. Models developed at one layer can be used by another model at a higher layer. Figure 10 portrays this hierarchical organization. The following sections introduce each of the modeling domains. The Parameter Editor is always seen as a utility editor, and not considered a modeling domain.

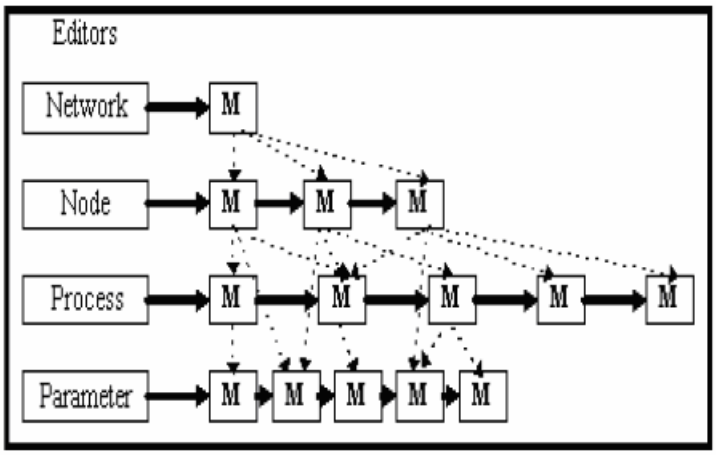

Figure 6.10: Hierarchical Organization of Editors

Network Model

Network Editor is used to specify the physical topology of a communications network, which define the position and interconnection of communicating entities, i.e., *node* and *link*. The specific capabilities of each node are realized in the underlying *model*. A set of parameters or characteristics is attached with each model that can be set to customize the node's behavior. A node can either be fixed, mobile or satellite. Simplex (unidirectional) or duplex (bi-directional) point-to-point links connects pairs of nodes. A *bus link* provides a broadcast medium for an arbitrary number of attached devices. Mobile communication is supported by *radio links*. Links can also be customized to simulate the actual communication channels.

The complexity of a network model would be unmanageable where numerous networks were being modeled as part of a single system. This complexity is eliminated by an abstraction known as a *subnetwork*. A subnetwork may contain many subnetworks, at the lowest level, a subnetwork is composed only of nodes and links Communications links facilitate communication between subnetworks.

Node Model

Communication devices created and interconnected at the network level need to be specified in the node domain using the Node Editor. Node models are expressed as interconnected *modules*. These modules can be grouped into two distinct categories. The first set is modules that have predefined characteristics and a set of built-in parameters. Examples are packet generators, point-to-point transmitters and radio receivers. The second group contains highly programmable modules. These modules referred to as *processors* and *queues*, rely on process model specifications.

Each node is described by a block structured data flow diagram. Each programmable block in a Node Model has its functionality defined by a Process Model. Modules are interconnected by either *packet streams* or *statistic wires*. Packets are transferred between modules using packet streams. Statistic wires could be used to convey numeric signals.

Process Model

Process models, created using the process editor, are used to describe the logic flow and behavior of processor and queue modules. Communication between process is supported by *interrupts*. Process models are expressed in a language called Proto-C, which consists of state transition diagrams (STDs), a library of kernel procedures, and the standard C programming language. The OPNET Process Editor uses a powerful state-transition diagram approach to support specification of any type of protocol, resource, application, algorithm, or queuing policy. States and transitions graphically define the progression of a process in response to events. Within each state, general logic can be specified using a library of predefined functions and even the full flexibility of the C language. Process may create new processes (*child process*) to perform sub-tasks and thus is called the *parent process.*

Running Simulation

Simulation Editor

After defining all the models of the network system, we can exercise them in a dynamic simulation in order to study system performance and behavior. Generally, there are three steps for simulations execution and information collection:

1. Specifying Data Collection: Model developers always need to decide which information should be extracted from the simulation, such as application-specific statistics, behavioral characterizations, and sometimes application-specific visualization. These can take on several different forms including visual animations, time-dependent series of values (vector), and parametric relationships (scalar).
2. Simulation Construction: OPNET simulations are obtained by executing a simulation program, which is an executable file in the host computer's file system.
3. Simulation Execution: Simulation execution is the final step in an "iteration" of a modeling experiment. In general, based on the results observed during this step, changes are made to the model's specification or to the probes, and additional simulations are executed. OPNET provides a number of options for running simulations, including internal and external execution, and the ability to configure attributes that affect the simulation's behavior. This section introduces concepts, techniques, and features that support simulation execution.

OPNET simulations can be run independently from the OPNET graphical tool by using the op_runsim utility program. However, you can also run simulations from the Simulation Tool within OPNET, which offers the convenience of a graphical interface. The Simulation Tool provides the following services: 1) specification of simulation sequences consisting of an ordered list of simulations and associated attribute values 2) execution of simulation sequences 3) storage of simulation sequences in files for later use.

Data Generation

Probe Editor

Most OPNET models that contain objects that are capable of generating vast amounts of output data during simulations. The sources of output data include pre-defined and user defined statistics, automatic animation, and custom programmed animation. Users can use Probe Editor to specify which data to collect. A probe is defined for each source of data that the user wishes to enable. Probes are grouped into a probe list which, allowing them to be collectively applied to a model when a simulation is executed. Several different probe types are provided by OPNET in order to capture different types of output data. These are:

- **Statistic Probe** This type of probe can be applied to predefined, standard statistics monitoring characteristics such as bit error rates or throughput.
- **Automatic Animation Probe** This type of probe is used to generate animation sequences for a simulation.
- **Custom Animation Probe** Process and link models also support the creation of custom animations. The actual specification of the animation's characteristics is defined within the user's code.
- **Coupled Statistic Probe** This type of probes generates output data as the statistic probe does but, in addition, a *primary module* and a *coupled module* need to be defined. Some statistical data is generated at the primary module. This data is only generated when changes to the statistic are due to interactions with the coupled module. This type of probe is only used for radio receiver.

Analysis Tool

Simulations can be used to generate a number of different forms of output, as described above. These forms include several types of numerical data, animation, and detailed traces provided by the OPNET debugger. In addition, because OPNET simulations support open interfaces to the C language, and the host computer's operating system, simulation developers may generate proprietary forms of output ranging from messages printed in the console window, to generation of ASCII or binary files, and even live interactions with other programs. However, the most commonly used forms of output data are those that are directly supported by Simulation Kernel interfaces for collection, and by existing tools for viewing and analysis. Both animation data and numerical statistics fall into this category. Animation data is generated either by using automatic animation probes or by developing custom animations with the KP's of the Simulation Kernel's Anim package; the m3_vuanim utility is then used to view the animations. Similarly, statistic data is generated by setting statistic probes, and/or by the KP's of the Kernel's Stat package; OPNET's Analysis Tool can then be used to view and manipulate the statistical data.

The service provided by the Analysis Tool is to display information in the form of graphs. Graphs are presented within rectangular areas called analysis panels. A number of different operations can be used to create analysis panels, all of which have as their basic purpose to display a new set of data, or to transform an existing one. An analysis panel consists of a plotting area, with two numbered axes, generally referred to as the abscissa axis (horizontal), and the ordinate axis (vertical). The plotting area can contain one or more graphs describing relationships between variables mapped to the two axes. For example, the graph in the panel below shows how the size of a queue varies as a function of time.

Filter Tool

OPNET's Analysis Tool allows the user to extract data from simulation output files and to display this data in various forms, as described in Chapter Datan of the OPNET Modeling Manual. The Analysis Tool also supports several mechanisms for numerically processing the data and generating new data sets that can also be plotted. These include computing probability density functions and cumulative distribution functions, as well as generating histograms. The data presented in the Analysis Tool may also be operated on by numeric filters. These are constructed from a pre-defined set of filter elements in the Filter Editor.

Filter models are represented as block diagrams consisting of interconnected filter elements. Filter elements may be either built-in numeric processing elements, or references to other filter models. Thus, filter models are hierarchical, in that they may be composed of other filter models. However, all filter models must be composed at the lowest level of pre-defined filters discussed in Chapter Datan of the OPNET Modeling Manual.

Filters operate on vectors. Vectors are discrete and ordered sets of numeric data which consist of entries, as discussed in Chapter Datan of the OPNET Modeling Manual. Each entry consists of an abscissa and an ordinate value. These are double-precision floating point numbers. A filter model may operate on one or more vectors and combine them to form its output, which must consist of just one vector. The vectors that are fed into the filter are called input vectors; the result of the filter‘s processing is called the filter‘s output vector.

Problems with the Simulations of Security Services of Distributed Systems

OPNET, as the name implies, is a tool for network simulations. However, it is still being used by the distributed system’s community mainly due to its fame and ease of use. The general practice is to convert a distributed system into a corresponding network model; carry out the simulations; and translate back to the distributed model to evaluate the results obtained. This approach is fairly workable with a limited size of the distributed systems where the scalability factors and the ‘open’ nature are not of paramount importance. In our case, especially with the consideration of the heterogeneity of the large scale systems, it is too risky to have a ‘perfect’ translation of the security architecture of a large scale open heterogeneous distributed system into a corresponding network model and vice versa. Moreover Opnet has some problems when more than one application had to be used on a workstation. For some reason, there was always a workstation that received no traffic of the server. We had several tasks which we called in several applications and put into different profiles. But when we tried to simulate this model, there was always a workstation that did not receive any traffic from the appropriate server, whatever adjustments we introduced to the model.

There also seemed to be quite some bugs in Opnet. For example when starting the simulation, Opnet always gave some 'recoverable errors', though they didn't seem to have any affect on the results. Moreover, it is difficult to simulate a lot of traffic.

We have kept in mind all these problems while designing G3S and that’s why G3S provides better environment than OPNET for the simulations of the security services of large scale open heterogeneous distributed applications and systems. Moreover, it’s successful integration with the GridSim has also resulted in the provision of a tool capable of providing a complete set of simulations for a large system like computational grids.

6.3. Quality of Protection (QoP)

Our approach compliments the weakness in current security evaluation mechanisms that do not provide a discreet quality of protection (QoP) parameter. The Quality of Protection (QoP) is a criterion that includes security bindings supported by the service, confidentiality and integrity requirements It is an essential parameter, as well as security attributes (service identity...), that has to be defined by a service. Service requestors are therefore able to evaluate their invocation policies, discover a service based on their security characteristics. At the other end, service providers can be sure that service requestors are subject to policy checks defined by the access policies attached to the service. For example, some policies may require that the service provider will only allow the invocation of a service after the service requestor has authenticated itself first, and provides an appropriate credential when invoking the service.

A Grid (or any other large scale distributed heterogeneous system) service must be able to define or publish the Quality of Protection (QoP) it requires and the security attributes of the service Aspects of the QoP include security bindings supported by the service, the type of credential expected from the service requestor, integrity and confidentiality requirements, etc. The security attributes of the service can include information such as service identity. This

enables service requestors to discover a service based on the requestor's security characteristics. Additionally, service requestors will be able to evaluate their invocation policies based on the security attributes of the service. Note that that there may be policy restrictions on the visibility of the service's security attributes.

From the service provider's point of view requests to invoke Grid services by service requestors are subject to policy checks defined by the service's access policies. For example, some policies may require that the service provider will only allow the invocation of a service after the service requestor has authenticated itself first, and provides an appropriate credential when invoking the service.

These requirements highlight the need for establishing standard mechanisms for conveying and enforcing the quality of protection, security attributes and access policies associated with services and requesters.

The attack simulations provided by the G3S help its users to evaluate the QoP parameter to examine the trade-offs in terms of system performance and security services.

6.4. Quality of Security Services (QoSS)

To handle changing security requirements in the distributed environments, we strongly recommend the use of Quality of Security Service (QoSS). It governs security technologies and protocols to be used as requested by the application and governed by existing policies, rules, history and trust. Different systems and communications require different levels of security applied to them. This means different security services, policies and processes should be able to dynamically discovered and used. Dynamic composition of these security services to create higher-level composite services is a feature of our proposed security architecture. Based on a predefined request for a certain security level from the application service layer, certain services are negotiated by using the security broker as a security agent between the core security architecture and the applications. It can be imagined as a security stack that is composed dynamically based on certain requirements and requests from the application. We strongly believe that both fine and coarse grain security services and every thing in between should be available to the application layer. In order to pick and choose what components and protocols to be used the application needs to be able to discover what services are available to it at a given time. It also needs to be aware of the network connectivity and network layer security services that are available to it for a certain request. QoSS agreements would be negotiated dynamically. An association control service needs to provide the service elements for establishing, handshaking and agreement on security level and termination of such association. The association could be long lived or short lived depending on the nature of the request.

For a Quality of Service (QoS) dimension to be supported means that users can request or specify a level of service for one or more attributes of this dimension, and the underlying QoS control mechanism is capable of entering into an agreement to deliver those services at the requested levels. Therefore, the control mechanism must be able to modulate the level of the service to individual subscribers (e.g., users).

Users may have expectations (i.e., functional and assurance requirements) with respect to the security services they are provided. Quality of Security Service (QoSS) has the meaning that security and security requests can be managed as a responsive *service* for which quantitative measurement of service *efficiency* is possible.

QoS mechanisms can be more effective with security appearing as a QoS dimension: when variable levels of security services and requirements are presented to users or network tasks, the underlying system can adapt more gracefully to changes in resource availability during the execution of a task, and thereby do a better job at maintaining requested or required levels of service in all of its dimensions.

The enabling technology for both QoSS and a security adaptable infrastructure is variant security, or the ability of security mechanisms and services to allow the amount, kind or degree of security to vary, within predefined ranges. This notion of network Quality of

Security Service has the potential to provide administrators and users with more flexibility and potentially better service, without compromise of network and system security policies.

To be general, we will define that all security requirements have a range of permissible behavior. In some cases, a range may be unitary, or degenerate, in which case it represents no choice. Where a range represents a choice, the requirement is termed *security variant*. In the same sense all system security services can be considered as having a range: since they are invoked at the discretion of the user or application, the range is at least binary (i.e., invoked or not invoked).

This notion of variant requirements and security ranges may, at first, seem strange. For many, either you have security or you don't. This is true on a gross scale, since without some minimum level of security, a system will be considered inadequate for user requirements. But if a user's minimum requirements are met, there can be some choice with respect to what is adequate.

Some examples in which security ranges and choices could be available:

- collaborative applications, such as video teleconferencing with shared electronic boards and application suites: if a group member is participating in the collaboration from a hotel room in a foreign country known for government support of corporate espionage, his security requirements and choices will be quite different than if he were in "friendly" territory. These security choices may form a range from which the user or application can select, and can include different levels of authentication, confidentiality and integrity.
- a variable packet authentication scheme [112]: the recipient might be satisfied if a certain percentage of each packet in an image stream was authenticated (e.g., 80% to 100%); this might have applicability for image display, especially considering that the low order bits of each byte are not very significant visually, in some display protocols.
- an Intrusion Detection System (IDS): an administrator may choose to run the IDS within a range rather than a fixed level. There would be a minimal level of IDS processing below which the system would not be permitted to fall, but the IDS would be balanced against performance requirements of the organization's tasks. Thus the IDS might perform more thoroughly (with deeper histories) when the system is lightly loaded than during peak hours. The administrator might also choose to set an upper limit to IDS performance.

The following are some example security variables, with characterizations of how they could be specified or measured:

- Strength of cryptographic algorithm, e.g., RSA, DES: measured in terms of the work factor associated with a brute force attack
- Length of cryptographic key: characterized by bit length
- Percentage of packets authenticated: characterized by percentage of total (e.g., a multimedia environment might tolerate a percentage of data modification or loss)
- Security functions present in destination job execution environment: characterized by operating system or boundary control security policy enforcement mechanisms.
- Confidence of policy-enforcement in remote login environment: characterized by third party evaluation
- Robustness of authentication mechanism: here the range might span weak password, strong password, biometric, and smart cards with on-board display and input interfaces.

Chapter 7

Applications

7.1. Overview

In this chapter, a list of potential systems that can benefit from the security model proposed in this thesis. They are the contemporary, large scale, open, distributed, and heterogeneous systems. The need of a suitable security architecture for these systems is keenly felt in the recent literature [122-125]. In the following sections, a description of these systems is given with special emphasis on their specific security requirements. As our proposed security model is specifically designed for such kind of systems and we have shown in chapter 5 that how its various units tackle the overall security requirements of these systems and its evaluation is presented in chapter 6. In this chapter, we elaborate a set of real life applications of our proposed security model that justifies its significance in the contemporary large scale open distributed heterogeneous systems.

7.2. Life Sciences

Information technology has dramatically reduced the costs, increased the speed, and improved the productivity of life sciences research and development (R&D). Life sciences R&D, in turn, have opened up new challenges and opportunities for IT applications. This virtuous cycle has contributed to a whole new frontier for knowledge generation. For example, the confluence of IT and biological advances made possible the mapping of the entire human genome and genomes of many other organisms in just over a decade. These discoveries, along with current efforts to determine gene and protein functions, have improved our ability to understand the root causes of human, animal and plant diseases and find new cures. Furthermore, many future IT innovations will likely be spurred by the data and analysis demands of the life sciences.

Health Grid

Based on the grid technologies, the vision of Health Grid is to create an environment where information at the 5 levels (molecule, cell, tissue, individual, population) can be associated to provide individualized healthcare.

Already today, the availability of large amount of data (clinical, genomic, proteomic, etc) in heterogeneous sources and formats, and the rapid progress in fields such as computer based drug design, medical imaging and medical simulations have lead to a growing demand for large computational power and easy accessibility to heterogeneous data sources in the Health domains.

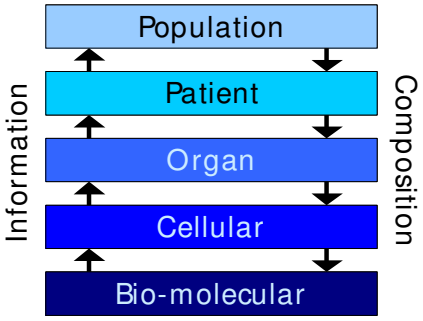

Figure 7.1 : Health Grid

Health Grid Security Requirements

Health Grid does not intend to restrict to the use of Grid technology for distributed computing only. Eventually, Health Grid should offer a generic platform for all e-Health actors. Sharing of large amounts of distributed heterogeneous (on various levels) data is therefore an important point of attention.

It is clear that the linkage of several distributed data sources bound to a single individual on a data Grid opens up a range of security concerns. The (virtual) federation of a large amount of personal medical data is not the only risk at hand. Grid technology will undoubtedly further stimulate the use of genomic data in research. However, this particular type of data has a number of specific characteristics related to security which are not found in any other type of (medical) information:

- Genetic data not only concerns individuals, but also their relatives. A person's consent to release his or her genetic information constitutes a de facto release of information about other individuals, i.e. his or her relatives. In the case of genomic medicine, there is a complex interaction between individual rights and collective requirements;
- Medical data deal with past and current health statuses of persons, whereas genetic information can also give indications about future health or disease conditions;
- An individual person's genotype is almost unique and stable, hence it can become the source of an increasing amount of information;
- The full extend of the information included in the genomic data is not known yet, hence it is difficult to assess the full extent of disclosure;
- Genomic data is easily wrongly interpreted by non-professionals; *susceptibility* to diseases can easily be mistaken with certainty of illness.

The above clearly indicates that the reconciliation of two seemingly conflicting objectives: on the one hand, the maximization of medical research productivity and efficiency in data handling; on the other, the protection of the human privacy rights; is the challenge at hand.

A couple of basic approaches to safeguarding confidentiality have been identified in the past in healthcare practice. The first approach focuses on the creators and maintainers of the information, prohibiting them from disclosing the information to inappropriate parties. Basically, this comes down to the deployment of classical security measures (**access control, authorization**).

With the introduction of a Health Grid, the need for confidentiality and data protection is more real than ever. The Grid promises access to heterogeneous resources, which means that in a Health Grid environment remote resources will be storing and processing sensitive personal data. These resources should thus be trusted by the end-user. But how can one know? Who can be the judge of *trustworthiness* of a Grid resource? A simple and straightforward solution is to use *closed* systems, which means that any resource in the Grid is well known and specified in advance. This however conflict with the vision of the dynamic nature of Grid technology.

Solutions should rather be searched in the area of policy advertising and negotiation. Resources should be able to inform a candidate user on how the data dealt with will be treated, which policies are applied, what PETs are used, who can have access to the data, etc. These methods are sometimes referred to as not being genuine Privacy Enhancing Techniques, as they do not actually limit collection of personal identifiable data and do not give any guarantees about the actual processing. A resource can claim to adhere to strict rules, but in practice this can not be verified.

The first steps in the direction of policy management have already been taken by Grid developers. The development of standards such as WS-Privacy, WS-Policy and EPAL (Enterprise Privacy Authorization Language) is an effort in the good direction. However, implementation till this day is rather limited, and the full possibilities of the technology will not be researched unless effort is spent here from the healthcare area (the main application

domain). A Health Grid would be the ideal environment where such PETs can be tested and further developed.

The above directly impacts typical Grid mechanisms such as data replication. Replication mechanisms automatically copy data on a resource in order to increase efficiency (e.g. to avoid transfer delays). With medical data, this might however not be allowed. The site on which the data will be replicated should at least be as trustworthy as the data source and should adhere to the same strict policies. A Health Grid should be able to handle such cases autonomously in order not to loose its dynamic nature and hence its efficiency. Another example is delegation. Delegation of rights is fundamental in a Grid environment, however in the medical world, this is far from obvious. If one passes on rights to others (resources), one becomes liable for actions performed on one's behalf. In a healthcare environment this has serious implications on liability. Auditability and accountability features offer a path to a solution suitable for medical applications.

Policy management will be an important topic in Health Grid, both for security (e.g. authorization policies) as for data protection (privacy policies). A difficult problem in this context is the one of policy enforcing and assuring that a certain policy is followed.

Equally important and closely related to this subject, is the implementation of **auditing mechanisms**. All actions in a medical context should be logged in a trustworthy way. Non-repudiation combined with a legal framework could help solve liability issues in healthcare.

Next to the areas of interest mentioned in this text, there exist several other healthcare needs for Grid applications that could be developed at e.g. upper middleware level for the benefit of a large community within a Health Grid context. Among those: encrypted storage for medical data (a far from obvious problem) and trustworthy federation of research databases – virtual federation of small "cells" of de-identified data (e.g. geographical area, hospital, …) can decrease the re-identification risk (by increasing the anonymity set). Finally a range of PETs which are well suited for distributed environments is emerging (Private Information Retrieval and Storage, privacy preserving data mining, processing of encrypted data, ...). However the road to an advanced generic privacy preserving framework for e-Health is still long and littered with technical difficulties which should be tackled one at a time.

It is however a fact that Grid technology can only be successful in a biomedical environment if the ethical guidelines and legal requirements are adequately met by technological solutions which are continuously evaluated and updated as new needs arise.

7.3. Critical Infrastructures

Critical Infrastructures are large scale distributed systems that are highly interdependent, both physically and in their greater reliance on the information and communication technologies (ICT) infrastructures, which logically introduce vulnerabilities that make them increasingly complex and fragile. Failures, accidents, physical or cyber attacks can provoke major damages which can proliferate by cascading effects and then can severely affect a part or the whole society. Because of their interdependencies and their increasing reliance on open systems, critical infrastructures constitute an unbounded system where faults may occur and proliferate in a severe way and where security represents a real challenge and requires new methodologies and tools. Securing the communications is an essential task. However, it is necessary to protect the infrastructures themselves (especially critical infrastructures) so that they become self-healing, fault tolerant, fault resistant, and resilient architectures.

New Paradigms in ICTs

In the ICT domain, new paradigms are emerging that comply with the complex demands of proximity and use, and that encourage the IT and telecom industries to prefer specific solutions that reconcile technology and markets with geography and users. Among these ideas, the concept of *ambient intelligence* points to the filling of geographical space with dynamic digital content (either information or computer programs). The concept of *grids* for intensive computation and the birth of pervasive computing means global and local

computation are becoming omnipresent. The planet will be covered with enormous middleware systems which will communicate two by two with variable granularity. Grids are dynamic virtual organizations for performing huge computations, with networks of clustered computers, scalable to enable massive distributed computation. Outsourced computing infrastructures will become semi-public resources and will be separated from their owner-users. This freedom to share computational infrastructures raises several questions such as ethical issues that are not easily solved. Finally, the *urbanization of heterogeneous interconnected networks* proclaims the ubiquity of communications and universal access to telecommunication infrastructures. The planet will soon be covered by these enormous fixed or movable structures, enabling local access to a digital infrastructure that can interoperate with all the other digital structures. Here again, the granularity and size of autonomous networks are very different depending if we consider a PAN, WLAN, WAN, or the Internet.

At the same time, the widespread of *wireless infrastructures* makes it almost impossible to delineate the contours of an information system. Not only has radio enabled building wireless networks, but also using the resurrected distributed computing, we are able, based on a standard resource available in proximity, to weave and configure in space a real and enormously powerful machine performing computation for its own sake. With the standardization of interconnections, it has become impossible to trace connecting wires or interoperating lines between several computers (running sometimes into the millions worldwide). This capability will become a permanent threat, as it will enhance the strength of the individual in relation to the State.

Because of their interdependencies and their increasing reliance on open systems, critical infrastructures constitute an unbounded system where faults may occur and proliferate in a severe way and where security represents a real challenge and requires new methodologies and tools [113]. Modern enterprises adapt quickly with short-decision cycles, fast-reaction loops and just-in-time procurement cycles. This results in chain reactions and/or hazardous automatic decisions when gaps appear in the behavior of systems and organizations, following inventory shortages, insufficient time, or shortages in logistics with unexpected consequences.

Potential threats to the normal functioning of infrastructures are both natural (“Murphy’s Law and Mother Nature”) and man-made. Individual outages can be serious enough, but this growing degree of interconnectedness can make possible a whole new scale of synergistic, nonlinear consequences.

System Requirements

The goals of a security management model are to be able to foresee the development flaws, detect anomalous behaviors to proactively manage the system in order to prevent serious problems, install prevention measures, and reactively control the system by making adjustments in response to changes (that may be sudden as when following an attack) within the system or its environment.

Even if it is almost impossible to prevent attacks, it is really important to be able to act quickly within the system to stop a potential proliferation of the problem. Consequently, two correlated works of modeling can be distinguished: one concerning CIs and one concerning security management.

Therefore, the basic requirements of the system are motivated by the following security functional requirements: prediction and scenario simulation (development, proactive management, etc.), prevention, monitoring (global view, reactive and proactive management, real-time), distributed intelligence and autonomy.

Security Requirements

When regarding the protection of the essential information infrastructures (and especially critical infrastructures), most of the time one concentrates on the availability subject. However, we put emphasis on not forgetting to protect integrity of provided services as well. Moreover, Service availability may conflict with other security goals that can be more

fundamental in some infrastructure cases; when integrity and confidentiality are the main goals, the most secure system is often one that does nothing. Therefore, protection against DoS often requires architectural changes to the system, which may prove expensive.

Another challenge for securing infrastructures is to make a trade-off between security and privacy. Technological developments and the needs of law enforcement provide increased opportunities for surveillance in cyberspace. Better managing and strengthening the infrastructure would make it more efficient and resilient without the need for unnecessary surveillance. A typical aspect of this issue is the problem of attack trace-back in Internet between the security (detecting the attacker) and the privacy (protecting the anonymity of Internet users).

7.4. Environmental/Meteorological Systems

While distributed sensor networks have great potential for advancing science, distributed collections of environmental data carry significant security implications. Sensor network architects and users must address security issues from the initial system design, and continue to do so with the data collected well after the network is dismantled. In a general sense, most security problems found in distributed sensor networks are also found in other distributed computer systems. However, the embedded nature and scale of distributed sensor networks pose novel security threats and exacerbate others.

Examples from the Internet motivate the need for investment in privacy and security. Consider the large amount of data generated and posted publicly on the Internet in the 1990s, without concern for security or privacy. At the time, lack of explicit control was of limited risk because data were transient, difficult to search, and seen by relatively few people. However, the data were archived, and are now indexed and easily searchable by today's search engines. Similarly, in the 1980s and early 1990s, systems attached to the Internet were rife with security vulnerabilities, but exploitation of these holes was rare and piecemeal. Today, in contrast, even a single vulnerability can cause widespread economic disruption.

Analogues to these and other problems exist in sensor networks. Data collected from a sensor network today may be difficult to exploit and seemingly innocuous. However, future improvements in programmability and data mining may result in unintended consequences. It is also clear that sensor networks can be attacked, which will result in erroneous data being saved. Future networks comprised of millions of embedded sensors might even provide a platform for a network or physical attack.

Users of sensor networks have security needs that are similar to users of traditional systems. They need data integrity and authentication: they want to know that the data they receive are uncorrupted, and know where they came from and when. Networks must maintain availability and be resilient to disruption; sensor networks that do not produce data are not useful. Privacy is needed, both for the scientists and the objects being observed. For reasons of correct attribution of work, scientists must be able to perform experiments confidentially, prohibiting others from viewing experiments in progress. There is also an issue of privacy regarding certain data that may inadvertently contain information beyond what the experimenters sought to gather. And while these needs fit into well-understood security categories, their threats and the means to neutralize those threats do not.

Key sensor network vulnerabilities include denial of service attacks, passive listening, and data insertion or corruption. Denial of service [114] can occur in many ways (e.g., by physically inserting a device that jams the wireless communications). Since a distributed sensor network may be deployed in remote regions, an adversary may physically destroy some subset of the devices. The wireless communication also permits passive listening by unauthorized individuals. Even worse, the insertion of corrupt sensor or control data could cause the system to stop operating, operate dangerously, make the collected data meaningless, or cause incorrect data to retard or wrongly direct scientific investigation.

Data collection on a large scale can have unintended consequences that can cause security risks. For example, a large system deployed in the ocean, such as NEPTUNE

(http://www.neptune.washington.edu/), can use microphones and sonar to monitor fish migrations. However, these raw data may unintentionally record faint traces of the U.S. submarine fleet; an adversary may be able to mine the raw data to learn valuable military intelligence.

The issue of data mining also poses threats to people's privacy. For example, once many sensor networks exist, data from different systems might be merged and assessed to acquire unexpected information about individuals, corporations, or governments. People need some degree of understanding and control over how they are observed by such networks, allowing them to make informed decisions about their privacy.

7.4.1. Challenges and Solutions

Three key factors pose significant security issues and challenges distinct from those found in traditional Internet-based systems: scale, embedment, and privacy [115]. As scientists and researchers deploy greater numbers of large-scale sensor networks, the security requirements of these systems and their impact on these three factors will become clearer. Identifying and characterizing these new security models is a significant task.

Sensor networks exist at many scales, from the 50-node NEPTUNE network to mote-based networks with thousands of nodes. Even larger systems and systems-of-systems will exist in the future. This wide range of scale imposes a correspondingly wide range of security challenges and required solutions.

Challenges

Modern computing systems such as laptops and desktops are typically rich in computational resources: they use billions of CPU cycles and hundreds of megabytes of memory to edit text or view images. This growth in power has allowed what were once computationally taxing operations to become commonplace. For example, when Adelman, Shamir and Rivest first proposed RSA encryption in 1978, encryption with a cutting-edge VAX computer took on the order of 30 seconds. Today, RSA encryption is used every time a secure website is accessed, taking a few milliseconds. These techniques may be applied to wired, resource rich nodes such as NEPTUNE.

In contrast, mote-based sensor networks are resource limited. With processors only marginally faster than those of a 1978 VAX and a few kilobytes of memory, they cannot afford to use the same algorithms and mechanisms that have become commonplace on personal computers. Since 1978, however, the importance of security in computing systems has increased greatly. For example, the first Internet worm was ten years later, in 1988. Mote based sensor networks must meet modern security needs but have available only limited resources, e.g., current motes must solve security problems with resource capacities similar to those available in general purpose processors twenty years ago.

In addition, mote-based networks are composed of large numbers of devices. A mote network administrator may be responsible for thousands of devices, and keeping track of each individual node is not feasible. As the scale of the network increases, this decreases the mean-time-to-failure of a node from the network. In networks with a large number of nodes that can readily fail, the administrator focuses on maintaining operation of the network as a whole even with these problems. The security model of a mote-based network must be similarly resilient to failure. This broad range of scales for networks results in a spectrum of security approaches, and heterogeneous networks must deal with many points on that spectrum simultaneously.

Unlike traditional computing systems, sensor networks are embedded in uncontrolled environments. For example, in Internet-based systems such as Web severs, physical compromise is rarely an issue, as the computers are in dedicated and locked server rooms. In sensor networks, however, the opposite conditions generally prevail, and nodes are not similarly protected. Instead, the network is often deployed in remote locations, far from easy visual observation. Under such conditions, an adversary can physically compromise nodes

even if the network communication is secure, and systems must be able to continue to operate in the presence of compromised nodes.

Not only does embedment pose security risks to a sensor network, it also raises questions on security implications for the collected data. Monitoring the environment can lead to gathering data on unsuspecting (or unwilling) subjects. For example, as mentioned above, the U.S. military has recently been concerned with NEPTUNE's deployment of seismographic and acoustic sensors in the deep ocean. Although the sensors are intended for geological, chemical, and biological research, the same data could be used to monitor ship and submarine movements. Protection against unintended uses of data is a very challenging problem.

As a result of the special needs of sensor networks, new security models must be developed. New metrics for assessing the security and safety of these systems are required. Fundamental questions that relate the lower bound on resource requirements necessary to meet various types and degrees of security need to be answered. Means to assess the impact of compromised nodes on the final accuracy of the collected data must be developed.

Solutions

The following proposed solutions are not meant to be exhaustive, but rather to illustrate directions that can provide some immediate solutions.

Many nodes used in sensor networks provide limited resources for computation and communication. These limitations severely hinder the use of widely available implementations of cryptographic algorithms that have driven security solutions in the broader community [116]. Research aimed at developing *light-weight implementations of cryptographic algorithms* [117] could enable for sensor networks a large collection of techniques that have been tested and evaluated in a broader community.

Given a sensor network consisting of thousands of nodes operating in a harsh environment, node failures due to factors such as hardware errors, software bugs, or attack are inevitable. In addition to securing individual nodes, it is necessary to design systems that are *resilient to attacks* and other forms of node failure. The concept of graceful degradation has been a cornerstone of distributed and fault tolerant systems, and the applicability of this approach to sensor networks and security should be explored. In particular, systems should be able to continue to operate in the presence of compromised nodes. The broader community has developed a number of approaches for detecting intrusions and network anomalies. These approaches may be fruitfully adapted to the environment presented by a sensor network. Such approaches should make it possible to *identify compromised nodes* and *revoke any rights* they may have within the network. As an example, work in wireless ad hoc networks that enable each node to actively overhear the wireless channel, identifying anomalies of its neighbors' transmissions, has demonstrated the capability of such active defense to be an effective counter to attacks [118, 119].

Physical compromise of a sensor node could reveal critical information (e.g., encryption keys) that could be used to impersonate the compromised node. Special, *tamper resistant* nodes that destroy their storage upon physical tampering would defeat such an attack.

Characteristics of the deployed network and the subjects being sensed can be used to validate the authenticity of collected data. As an example, identifying the presence of an automobile in one location at one instance followed immediately by an indication that the automobile had moved a great distance or that the automobile was following a physically impossible path could be an indication that the network is being spoofed. Also, given the high density of sensors in networks, the inherent redundancy can be exploited to solve some of these security problems.

The correct operation of middleware services such as the localization of nodes, time synchronization, data routing [117], and self-calibration are essential to the functioning of many sensor networks. When necessary, these *middleware services* should be secured against attack. A number of proposals [117, 120, 121] have begun to address these issues, but the broader space of such problems remains largely unexplored.

Attacks can be launched against different levels of a system. A malicious “black-hole” node might try to attract data from nodes throughout the network, interfering with the data-collecting ability of a real base station. A “jammer” might transmit noise to disable the communication in its vicinity. *Multiple layers of defense* not only protect the network from a diverse spectrum of attacks, but also ensure that a breach of one line of defense does not compromise the entire system.

Sensor network users are likely to perceive security as an absolute, i.e., they are likely to believe that the system is either secure or not secure. As with other systems, the reality is not so well-defined. A sensor network may be protected from some security violations while being vulnerable to others. Specific issues include the degree of trust and the potential for social impact (e.g., invasion of privacy) of the sensing and data collection activities. Scientists and the public need to be informed about the complex consequences associated with deployment of sensor networks. This aspect of security is best addressed through education.

In practice, sensor networks are likely to be deployed by scientists who are not security experts. A *composable security infrastructure* which supports the construction of sensor networks from smaller parts that are secure and trusted will be invaluable to the future deployment of sensor networks. As an example that works for the Internet, SSL (Secure Sockets Layer) provides an infrastructure that allows individual machines to be added to the Internet while retaining the desired security properties.

Future sensor networks may require large numbers of heterogeneous nodes. Authentication schemes will need to be able to scale to the magnitude required to support such large-scale systems. The building blocks of *authentication* should have sufficient *modularity* to easily enable interoperation among heterogeneous software and hardware components for a coherent system.

Basic Research in Cyber Security

While it is clear that the security challenges introduced by sensor networks will benefit from general research in cyber security, sensor networks present four research opportunities that are unlikely to arise in other contexts. First, the security of sensor networks should take advantage of properties of the physical environment in which they are deployed. This exploitation of physical properties to enhance network security is a fertile ground for novel techniques and mechanisms. Second, security mechanisms of sensor networks should self-organize to minimize human intervention. Because of the potentially large scale of sensor networks, autonomic approaches such as self-diagnosis and self-healing are necessary to relieve the user from the burden of attending large numbers of nodes individually. Third, research should identify the extent to which not just individual nodes but overall system architectures can be secured.

Because many sensor networks will be constructed from sensors with severely limited resources, traditional approaches that emphasize the security of individual nodes may not be appropriate. System level approaches, including resilience techniques that ensure operation of the network in the presence of a certain percentage of compromised nodes, should be investigated. Finally, because sensor networks rely on the correct operation of specific services such as routing, localization, etc., research should investigate the degree to which the security of these “middleware services” can be enhanced, in light of the limited resources available on a sensor node.

Testbed Sensor Network Systems

While many of the issues related to security in sensor networks can be studied in isolation, design and implementation will need to be examined in a more complete context. To ensure the validity of approaches to network security, funds are needed to support the development of fairly large testbed/prototype sensor network systems that involve multidisciplinary teams from both science and technology. These systems should be driven by scientific exploration

of a specific phenomenon where security is an explicit requirement. Security must also be an integrated part of the design from the beginning.

7.5. Collaborative Distance Learning

In order to support ubiquitous, collaborative, experiential and contextualized learning in dynamic virtual communities a learning environment should provide the following features for learners:

- Collaboration; Socio-constructivist: group working should be routinely supported as well as the more traditional model of the solitary learner – this includes support for self-organizing online communities who share common educational goals
- Experiential; Active Learning: learning resources should be interactive, engaging, and responsive – active learning and knowledge formation should be emphasized above simple information transfer
- Realism: real-world input should be easy to incorporate, as should simulations, ranging from simple interactive animations to immersive VR
- Personalized: students should find themselves at the centre of their online environment, with their individual needs addressed - the quality of the learning experience should be continually validated and evaluated
- Ubiquity and accessibility:
    - wider, more flexible access to educational resources should be provided, often referred to as “anytime/anywhere” learning.
    - multiple different types of devices, interfaces, and network connection types should be supported where possible
- Contextualized; Adaptive: appropriate learning contexts may be naturally be short-lived, as well as the more traditional static situations such as the classroom and the library – this calls for dynamicity in the creation of contexts

The pedagogical goals outlined above have highly demanding technical requirements, many of which are also the concerns of distributed systems research. Group working implies shared interactive resources, necessitating both concurrency control and awareness of others activities. Active learning requires interactive resources, many of which will only be engaging if they are suitably responsive – a quality of service (QoS) issue that depends on many components of a distributed system – the low-level infrastructure (hardware, OS, network), the middleware and the interface software. Concurrency control and interactive responsiveness can make conflicting demands on a system. Real world input, such as live stock market prices, or remote sensing data, makes a network connection mandatory, and this again raises QoS issues such as fault detection, masking and tolerance for the learning environment. Accessibility, as in anytime/anywhere, requires availability, which may be supported through replication of resources, but this creates further tensions with responsiveness and concurrency control due to the need to maintain state across replicas. Accessibility also means adapting to available capabilities. For example: can the same learning environment be delivered through low-bandwidth mobile devices and high-bandwidth multimedia workstations? Accessibility also means supporting special needs of the individual, such as disabilities. More generally, the individual user should be recognized and catered for, and this personalization requires semantic tagging and profiling that can be difficult to formulate, both conceptually and in terms of machine representation. Standards efforts have been particularly slow in addressing this problem. Finally, contextualization requires a move from the traditional view of an online learning environment as a stable long-lived entity (e.g. during the lifetime of a teaching module) – to one where the environment may evolve and change much more frequently, perhaps even several times a day – a dynamicity that is alien to current e-Learning products.

Chapter 8

Conclusions

Managing security in large scale heterogeneous distributed computing systems is a non-trivial problem. In such systems, the relationships are dynamic in nature which requires dynamic and adaptable security modules. Due to these reasons, currently available solutions usually lead to heavy administrative burden or weak security.

In this thesis, a new approach is proposed to deal with a number of security challenges presented by large scale, open, distributed heterogeneous systems. The most salient feature of our approach is the flexible and adaptive nature of security services. We have used virtualization to provide standardized ways of enabling the federation of multiple heterogeneous security mechanisms. To have minimal reliance on the emerging resource management functionalities, and to make our model more adaptive, we have extended the concept of *security as services* to *security as pluggable services*. The other features are the self-security of the security architecture; use of security broker that negotiates for security services; description of security ontology to enable standard protocol interactions of core security bootstrapping services; and user-centered security services where usability is the prime motivation.

Our research has been a first step to come towards a systematic approach in the design process of security architecture for large scale, open, distributed heterogeneous systems. Although a wide variety of complex systems are considered but more consideration is given to the computational grid based systems. This work can be continued to explore more specific security solutions for other complex systems such as ubiquitous systems, P2P systems, etc. Moreover, the concept of virtualization could be extended to adapt country-specific legal requirements, population-based ethical issues, and the business-oriented interests. Furthermore, virtualization could be used to achieve the best trade-off between security guarantees and processing capabilities.

8.1. Recommendations for the Future Research

This work has laid a foundation for the comprehensive security services concept for the large scale open heterogeneous distributed computing systems and it opens up several avenues for future work. Even though our work covers security services required for a large scale heterogeneous distributed computing systems, there are other areas of research that can provide additional features into this work. Such as fault tolerant mechanisms can be developed and deployed, use of more advanced methodologies to provide automated updation of trust values, features supported for other information services (such as Globus Toolkit) can be exploited to support policy publishing and parameters retrieval.

Another important and interesting area of research as an extension to this work is designing an automated security services selection system based on the history, context and state of the system. Several challenges related to these factors needed to be addressed in such systems. If selection is merely based on reputation, then the services with high reputations will always suffer from heavy load. Robust algorithms can be developed to handle such issues in developing automated security services selection systems. Such work would greatly enhance the current state of this model.

Within the Grid community, there is a great interest in building an accounting model and infrastructure. Integration of some accounting mechanism similar to the concept of Grid Market Directory, Grid Bank (GB) [126] will be of great interest.
An addition to any comprehensive security policy is the inclusion of an intrusion detection system, either signature- or anomaly-based. The benefits of a good anomaly-based system are obviously great and are preferred to those of a signature. Unfortunately, anomaly-based IDSs are still in their infancy and, therefore, beginning with a signature-based IDS to provide known intrusion detection would be a lower risk approach.

Along with an intrusion detection capability, the system should possess an effective and efficient response capability in order to effectively protect the system and minimize effects of an attack. Responses could take the form of denial of future connectivity to a malicious application, dynamic key changes in response to a discovered compromise of the symmetric keys, or even the use of software decoys in order to learn more information about the attacker and the nature of the intrusion.

The G3S toolkit is rapidly evolving. The security model needs to be enhanced by supporting various types of other functionalities such as delegation of rights, networks with different static and dynamic configurations and cost-based QoS services.

8.2. Final Comments

It is important to remember that security is a process, the threat picture is always changing, and threat analysis needs to be continuously updated. In other words, grid infrastructure should be subject to constant review and upgrade, so that any security loophole can be plugged as soon as it is discovered. The growth in the users community should lead to improvements as larger number of users will find the loopholes faster, and more developers will be available to fix them and release patches.

References

1. Naqvi S., Riguidel M., *Security Architecture for Heterogeneous Distributed Computing Systems*, IEEE International Carnahan Conference on Security Technology 2004 (IEEE ICCST2004), Albuquerque, New Mexico - USA, October 11-14, 2004. pp 34-41 (ISBN 0780385063)
2. Naqvi S., Riguidel M., *Problems in the Implementation of Grid Security Services*, Cracow Grid Workshop 2004 (CGW'04), Krakow – Poland, December 12-15, 2004. pp 338-346 (ISBN 8391514145)
3. Naqvi S., Riguidel M., *Security Challenges for Highly Available Systems*, IEEE International Carnahan Conference on Security Technology 2005 (IEEE ICCST2005), Las Palmas, Spain, October 11-14, 2005
4. Naqvi S., Riguidel M., *Security Risk Analysis for Grid Computing*, Proceedings of Cracow Grid Workshop 2003 (CGW'03), Krakow – Poland, October 27-29, 2003. pp 174-189 (ISBN 8391514137)
5. Naqvi S., Riguidel M., *Threat Model for Grid Security Services*, European Grid Computing Conference 2005 (EGC2005), Amsterdam, Netherlands, February 14-16, 2005. pp 1048-1055 (ISBN 8391514145)
6. Naqvi S., Riguidel M., *Addressing Secure Access Challenges for Nomadic Grid: A Hospital Case Study*, Grid Asia Conference 2005, Biopolis, Singapore, May 2-6, 2005
7. Naqvi S., Riguidel M., *Secure Data Exchange Between Intelligent Devices and Computing Centers*, SPIE Defense and Security Symposium 2005 (SPIE-DSS2005), Orlando, Florida - USA, March 28-April 01, 2005. pp 157-166 (ISBN 0819457884
8. Naqvi S., Riguidel M., Demeure I., *Security Architecture for Health Grid using Ambient Intelligence*, Health Grid Conference 2004 (HG2004), Clermont-Ferrand – France, January 29-30, 2004.- Published in the Special Grid Issue of Methods of Information in Medicine (MIM), vol. 44, May 2005, pp 202-206 (ISSN 0026-1270)
9. Kreuwels C., *Electronic data interchange*, IEEE Information Technology Conference 'Next Decade in Information Technology' (Cat. No. 90TH0326-9) 1990, Jerusalem, October 22-25, 1990, pp 214-224
10. Foster I., Kesselman C., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufman Publishers, ISBN 1-55860-475-8, August 1998
11. Lorch M., Kafura D., *Grid Community Characteristics and their Relation to Grid Security*, Technical Report TR-03-20, Computer Science, Virginia Tech., June 2003
12. Connor D., *Grid Computing Hits Security Gridlock*, Network World Fusion online magazine, 06 October 2002
13. European Union Information Society Technologies, *A thematic priority for Research and Development under the Specific Program "Integrating and Strengthening the European Research Area" in the Community sixth Framework Program*, http://www.cordis.lu/ist
14. National Science Foundation, http://www.nsf.gov

15. Naqvi S., Riguidel M., *Designing Security Architecture for Large Scale, Open, Distributed Heterogeneous Systems*, IEEE Symposium on Security and Privacy 2005 (IEEE-SP2005), Berkeley/Oakland, California - USA May 8-11, 2005

16. Naqvi S., Riguidel M., *VIPSEC: Virtualized and Pluggable Security Services Infrastructure for Adaptive Grid Computing*, Proceedings of IEEE International Symposium on Network Computing and Applications (IEEE NCA04), Cambridge, Massachusetts - USA, August 30–September 01, 2004 (ISBN 0769522424)

17. Naqvi S., Riguidel M., *Security and Trust Assurances for Smart Environments*, IEEE International Workshop on Resource Provisioning and Management in Sensors Network 2005 (RPMSN05), Washington DC, USA, November 7-10, 2005

18. Naqvi S., Riguidel M., *Dynamic Distribution of Trust in the Grid Environments*, eChallenges Conference 2005, Ljubljana, Slovenia, October 19-21, 2005

19. Naqvi S., Riguidel M., *Trust Establishment in Pervasive Grid Environments*, Cracow Grid Workshop 2005 (CGW'05), Krakow – Poland, November 20-23, 2005

20. Naqvi S., Riguidel M., *G3S: Grid Security Services Simulator*, Health Grid Conference 2005 (HG2005), Oxford, UK, April 7-9, 2005

21. Naqvi S., Riguidel M., *Grid Security Services Simulator (G3S) – A Simulation Tool for the Design and Analysis of Grid Security Solutions*, IEEE International Conference on e-Science and Grid Computing 2005 (e-Science 2005), Melbourne, Australia, December 5-8, 2005

22. Buyya R. and Murshed M., *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, The Journal of Concurrency and Computation: Practice and Experience, Wiley Press, May 2002. pp 1-32

23. Naqvi S., Riguidel M., *Performance Measurements of the VIPSEC Model*, High Performance Computing Symposium (HPC 2005), San Diego, California - USA, April 3-7, 2005. pp 182-187 (ISBN 1565552938)

24. Naqvi S., Riguidel M., *Impact of Comprehensive Security Services on Grid Computing Performance*, IEEE International Conference on Dependable Systems and Networks 2005 (IEEE-DSN2005), Yokohama, Japan, June 28 - July 1, 2005 (ISBN 0769522823)

25. Naqvi S., Riguidel M., *Dynamic Access Control for Pervasive Grid Applications*, IEEE International Conference on Computational Intelligence and Security 2005 (IEEE-CIS05), Xi'an, China, December 15-19, 2005

26. Naqvi S., Riguidel M., *Evaluation of Grid Security Solutions using Common Criteria*, Computing in High Energy Physics 2004 (CHEP'04), Interlaken - Switzerland, September 27 - October 01, 2004. pp 854-857 (ISBN 9290832452)

27. Naqvi S., Riguidel M., *Securing Grid-Based Critical Infrastructures*, IEEE International Conference on Intelligence and Security Informatics (IEEE ISI-2005), Atlanta, Georgia - USA May 19-20, 2005, pp 654-655 (ISBN 3540259996)

28. Foster I., Kesselman C., Tsudik G., Tuecke S., *A Security Architecture for Computational Grids*, Proceedings of the 5th ACM conference on Computer and communications security, Sann Francisco, California, United States, 1998, pp 83-92, ISBN:1581130074

29. Foster I., Kesselman C., Nick J., Tuecke S., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, January 2002.

30. Lai C., Medvinsky G. and Neuman, B., Endorsements, Licensing, and Insurance for Distributed System Services, Proceedings of the 2nd ACM Conference on Computer and Communication Security, 1994.

31. Nagaratnam, N, Janson P., Dayka J., Nadalin A., Siebenlist F., Welch V, Foster I. and Tuecke S., *The Security Architecture for Open Grid Services*, Version 1, 17 July 2002

32. Gordon L., Loeb M., Lucyshyn W., and Richardson R., *2004 CSI/FBI Computer Crime and Security Survey*, Computer Security Institute, 2004

33. LinuxWorld Report, *Linux Attacks On the Rise, Says Report - But It's Not As Simple As That*, February 22, 2004, http://www.linuxworld.com/story/43760.htm

34. ComputerWorld Report, *Security Statistics*, July 09, 2001, http://www.computerworld.com/securitytopics/security/story/0,10801,62002,00.html

35. San Diego Supercomputer Center (SDSC) Security Experiment – worm.sdsc.edu http://security.sdsc.edu/incidents/worm.2000.01.18.shtml

36. CERT Vulnerability Notes, http://www.cert.org

37. Nicole D., Scalability of Network Simulators Revisited, Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference Orlando, FL , February 2003

38. Bernstein D., Infosecurity News - Industry Survey, Infosecurity News, May 1997

39. Owens M., A Discussion of Covert Channels and Steganography, SANS Report, March 2002

40. IBM, Introduction to Business Security Patterns, IBM White Paper

41. Information Processing Systems, Open System Interconnection, Basic Reference Model, Part 2: Security Architecture (ISO 7498-2)

42. Massachusetts Medical Society House of Delegates, Massachusetts Medical Society Policy: Patient Privacy and Confidentiality, 1996

43. Internet Engineering Task Force (IETF) RFC 3280, http://www.ietf.org/rfc/rfc3280.txt

44. MyProxy Online Credential Repository, http://grid.ncsa.uiuc.edu/myproxy

45. Foster I., Kesselman C., Tuecke S., *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal of Supercomputer Applications, volume 15, issue 3, 2001.

46. http://www.globus.org

47. http://www.unicore.org

48. Fu Y., Chase J., Chun., Schwab S., and Vahdat A., *SHARP: An Architecture for Secure Resource Peering*, Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, August 2003

49. Zeiger A., *Grid Security: State of the Art*, IBM developerWorks online magazine, August 2003

50. http://www.cs.wisc.edu/condor

51. Frey J., Tannenbaum T., Foster I., Livny M., and Tuecke S., *Condor-G: A Computation Management Agent for Multi-Institutional Grids*, Journal of Cluster Computing volume 5, pages 237-246, 2002

52. http://legion.virginia.edu

53. Kunszt P., Guy L., *The Open Grid Services Architecture and Data Grids*, Grid Computing: Making The Global Infrastructure a Reality (Edited by Fran Berman), John Wiley & Sons 2003.

54. World Wide Web Consortium, XQuery 1.0: An XML Query Language, W3C Working Draft, December 2001.

55. Sandholm T., Tuecke S., Gawor J., Seed R., Maguire T., Rofrano J., Sylvester S., Williams M., Java OGSI Hosting Environment Design - A Portable Grid Service Container Framework, Global Grid Forum Drafts, GGF7 Meetings, March 2003.

56. Wasson G., Beekwilder N., Morgan M., Humphrey M., OGSI.NET: OGSI-compliance on the .NET Framework, Proceedings of 2004 IEEE International Symposium on Cluster Computing and the Grid, Chicago, Illinois, April 19-22, 2004

57. Gonzalez-Castano F., Vales-Alonso J., Livny M., Condor Grid Computing from Mobile Handheld Devices, Mobile Computing and Communications Review. Vol. 6, No. 2. ACM SIGMOBILE Mobile Computing and Communications Review. Volume 6, Issue 2, April 2002.

58. Phan T., Huang L., Dulan C., Challenge: Integrating Mobile Wireless Devices into the Computational Grid, Proceedings of MOBICOM'02, Atlanta, Georgia, USA, ISBN 1-58113-486-X, September 23-26, 2002, pp 271-278

59. Clarke B., Humphrey M., Beyond the 'Device as Portal': Meeting the Requirements of Wireless and Mobile Devices in the Legion Grid Computing System, 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (associated with IPDPS 2002), Ft. Lauderdale, April 19, 2002.

60. Rowstron, A., Druschel, P., *Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer-to-Peer Systems*, Proceedings of the IFIP/ACM Middleware 2001, Heidelberg, Germany, 2001

61. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S., *A Scalable Content Addressable Network*, Proceedings of the ACM SIGCOMM'01, San Diego, California, US, 2001

62. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H., *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*, Proceedings of the ACM SIGCOMM'01, San Diego, California, USA 2001

63. Zhao, B., Kubiatowicz, J., Joseph, A., *Tapestry: An infrastructure for fault-resilient wide-area location and routing*, Technical Report UCB//CSD-01-1141, University of California Berkeley, 2001

64. Douceur J., *The Sybil Attack*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts, USA, 2002

65. Merkle R., *Secure Communications Over Insecure Channels*, Communications of the ACM 21**,** 1978, pp 294-299

66. Schwartz J., Tedeschi B., *New Software Quietly Diverts Sales Commissions*, New York Times, 2002

67. Spring T., *KaZaA Sneakware Stirs Inside PCs*, PC World 2002
http://www.cnn.com/2002/TECH/internet/05/07/kazaa.software.idg/index.html

68. Weatherall D., *Active Network Vision and Reality: Lessons from a Capsule-based System*, Proceedings of the 17th ACM Symposium on Operating System Principles, Kiawah Island, SC, 1999, pp 64-79

69. Hicks M., Kakkar P., Moore J., Gunter C., Nettles S., *PLAN: A Packet Language for Active Networks*, Proceedings of the 3rd ACM SIGPLAN International Conference on Functional Programming Languages, ACM, 1998, pp 86-93

70. Wallach D., Balfanz D., Dean D., Felten E., *Extensible Security Architectures for Java*, Proceedings of the 16th ACM Symposium on Operating System Principles, Saint-Malo, France, 1997, pp 116-128

71. Reed M., Syverson P., Goldschlag D., *Anonymous Connections and Onion Routing*, IEEE Journal on Selected Areas in Communication: Special Issue on Copyright and Privacy Protection 16, 1998

72. Reiter M., Rubin A., *Anonymous Web Transactions with Crowds*, Communications of the ACM 42, 1999, pp 32-48

73. Waldman M., Rubin A., Cranor L., *Publius: A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System*. Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, USA, 2000, pp 59-72

74. Waldman M., Mazires D., *Tangler: A Censorship Resistant Publishing System based on Document Entanglements*, Proceedings of the 8th ACM Conference on Computer and Communication Security (CCS-8), Philadelphia, Pennsylvania, USA, 2001

75. Hazel S., Wiley B., *Achord: A Variant of the Chord Lookup Service for Use in Censorship Resistant Peer-to-Peer*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts, USA, 2002

76. Serjantov A., *Anonymizing Censorship Resistant Systems*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts, USA, 2002

77. Freedman M., Sit E., Cates J., Morris R., *Tarzan: A Peer-to-Peer Anonymizing Network Layer*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, Massachusetts, USA, 2002

78. Sit E., Morris R., *Security Considerations for Peer-to-Peer Distributed Hash Tables*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts, USA, 2002

79. Dingledine R., Freedman M., Molnar D., *Accountability Measures for Peer-to-Peer Systems*, Peer-to-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly and Associates, 2000

80. Bellovin S., *Security Aspects of Napster and Gnutella*, Invited talk in Usenix Annual Technical Conference, Boston, Massachusetts, USA, 2001

81. Yurcik W., Koenig G., Meng X., Greenseid J., *Cluster Security as a Unique Problem with Emergent Properties: Issues and Techniques*, *The 5th LCI International Conference on Linux Clusters: The HPC Revolution 2004*, May 2004.

82. Amoroso E., *Fundamentals of Computer Security Technology*, Prentice Hall International, 1994, ISBN 0-13305-541-8

83. Sheyner O., Haines J., Jha S., Lippmann R., Wing J., *Automated Generation and Analysis of Attack Graphs*, IEEE Symposium on Security and Privacy, 2002.

84. Burgess M., *Cluster Management with GNU cfengine*. Newsletter of the IEEE Computer Society's Task Force on Cluster Computing, 2002.

85. Kim G. and Spafford E., *The Design and Implementation of Tripwire: A File System Integrity Checker*, Proceedings of the 2nd ACM Conference on Computer and Communications Security, 1994, pp 18-29

86. Gorsuch N., *Linux Cluster Security*, Linux Revolution Conference, Urbana, Illinois, USA, June 26-27, 2001.

87. Distributed Security Infrastructure Open Source Project, http://disec.sourceforge.net

88. C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartmann, *Linux Security Modules: General Security Support for the Linux Kernel*, Usenix Security Symposium, 2002. http://lsm.immunix.org

89. Weiser M., *The Computer for the Twenty-First Century*, Scientific American, September, 1991, pp 94-10

90. MIT Project Media Lab – www.media.mit.edu/~nmarmas/comMotion.html

91. Carnegie Melon University Aura Project – www-2.cs.cmu.edu/~aura

92. University of California at Berkeley's Endeavour project – endeavour.cs.berkeley.edu

93. MIT Oxygen Project – www.oxygen.lcs.mit.edu

94. University of Washington Portolano project – portolano.cs.washington.edu

95. The Sentient Computing Project – www.uk.research.att.com/spirit

96. The CoolTown Project – www.cooltown.com

97. Microsoft Easy Living Project – research.microsoft.com/easyliving

98. Kishimoto H., Savva A., Snelling D., *OGSA Fundamental Services: Requirements for Commercial GRID Systems*, Technical Report, Open Grid Services Architecture Working Group (OGSA WG), April 2003.

99. Bussard L., *Trust Establishment Protocols for Communicating Devices*, PhD Thesis, October 2004

100. Pearlman L., Welch V., Foster I., Kesselman C., Tuecke S., *A Community Authorization Service for Group Collaboration,* Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY '02), Monteray, California, U.S.A. June 2002

101. F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks. In M. Roe B. Christianson, B. Crispo, editor, Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science. Springer-Verlag, 1999.

102. F. Stajano. The Resurrecting Duckling – what next? In M. Roe B. Christianson, B. Crispo, editor, Security Protocols, 8th International Workshop Proceedings, Lecture Notes in Computer Science. Springer-Verlag, 2000.

103. Welch V., Siebenlist F., Foster I., Bresnahan J., Czajkowski K., Gawor J., Kesselman C., Meder S., Pearlman L., Tuecke S., *Security for Grid Services*, Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), 2003

104. International Standards Organization, ISO/IEC 1508: Common Criteria, 1999

105. Wallace K., *Common Criteria and Protection Profiles: How to Evaluate Information Technology Security*, SANS Institute GIAC practical repository – version 1.4b, 2003

106. The Health Grid Organization, *Whitepaper on Health Grid*, 2004 – www.healthgrid.org

107. Takefusa A., Matsuoka S., Aida K., Nakada H., and Nagashima U., *Overview of a performance evaluation system for global computing scheduling algorithms*, Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC'99), Washington, DC, USA, 3-6 August 1999, pp 97-104

108. Legrand A., Marchal L., Casanova H., *Scheduling Distributed Applications: The SimGrid Simulation Framework*, Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid 2003 (CCGrid2003), May 12-15, 2003, pp 138-145

109. Dumitrescu C. and Foster I., *Gangsim: A Simulator for Grid Scheduling Studies*, Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05), Cardiff, UK, may 2005

110. Cameron D., Carvajal-Schiaffino R., Millar P., Nicholson C., Stockinger K., and Zini F., *OptorSim: A Grid Simulator for Replica Optimisation*, UK e-Science All Hands Conference 31 August - 3 September 2004.

111. www.opnet.com

112. Schneck, P.A. and Schwan, K, "Dynamic Authentication for High-Performance Networked Applications", Technical Report GIT-CC-98-08, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1998

113. Ellison R., Fisher D., Linger R., Lipson H., Longstaff T., Mead N., *Survivability: Protecting Your Critical Systems*. IEEE Internet Computing, Volume 3, No. 6, November/December 1999

114. A. Wood and J. Stankovic. Denial of service in sensor networks. *IEEE Computer*. 15(4), 48-56, 2002.

115. L. Zhou and Z. Hass. Securing ad hoc networks. *IEEE Network*. 13(6), 24-30, 1999.

116. D. Carman, P. Kruus and B. Matt. Constraints and approaches for distributed sensor network security. *NAI Labs: Technical Report # 00-010*, 2000.

117. V. Wen, A. Perrig and R. Szewczyk. SPINS: Security protocols for sensor networks. *Proceedings of the seventh annual international conference on mobile computing and networking*. Rome, Italy, July 16-21, 2001. pp 189-199.

118. S. Marti, T. Giuli, K. Lai and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Proceedings of the sixth annual international conference on mobile computing and networking.* Boston, MA, August 6-11, 2000. pp 255-265.

119. H. Yang, X. Meng and S. Lu. Self-organized network layer security in mobile ad hoc networks. *Proceedings of the first ACM Workshop on Wireless Security (WiSe)*. Atlanta, GA, September 28, 2002. pp 11-20.

120. Y. Hu, A. Perrig and D. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Proceedings of the eighth annual international conference on mobile computing and networking (Mobicom)*. Atlanta, GA, September 23-26, 2002.

121. J. Deng, R. Han and S. Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks, *Proceedings of the second international workshop of information processing in sensor networks*. Palo Alto, CA, April 22-23, 2003. pp 349-363.

122. Montagnat J, Bellet F., Benoit H., Breton V., Brunie L., Duque H., Legre Y., Magnin I., Maigne L., Miguet S., Pierson J., Seitz L., Tweed T., *Medical images simulation, storage and processing on the European DataGrid testbed,* Journal of Grid Computing 2(4):387-400, December 2004, Springer Verlag, ISSN 1570-7873

123. Ribeyrol C., *Support Policy for Future Projects on Critical Infrastructure Security*, Conference on Critical Infrastructures, Grenoble, France, June 2003

124. Chivers H., McDermid J., *Refactoring Service-Based Systems: How to Avoid Trusting a Workflow Service*, Grid Workflow 2004 Special Issue of Concurrency and Computation: Practice and Experience.

125. United States General Accounting Office, Progress and Challenges for DOD's Advanced Distributed Learning Programs, Report to Congressional Committees, February 2003

126. The GRIDBUS Project – www.gridbus.org

Glossary

ACL	Access Control List
AD	Actuator Dispatcher
AMS	Archive Management System
API	Application Programming Interface
BIOS	Basic Input/Output System
BOOTP	Bootstrap Protocol
BS	Bootstrap Service
CA	Certificate Authority
CAS	Community Authorization Service
CCM	Configuration Cache Manager
CDB	Configuration Database
CDP	Configuration Distribution Protocol
CDR	Central Data Recording
CE	Computing Element; a Grid-enabled computing resource.
CERT	Computer Emergency Response Team
CLI	Command Line Interface
CMP	Cache Manager Protocol
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CVS	Concurrent Versioning System
DAQ	Data Acquisition
DBMS	Data Base Management System
DHCP	Dynamic Host Configuration Protocol
FabNAT	Fabric Network Address Translation service
FLIDS	Fabric-Local Identity Service
FMFT	Fabric Monitoring and Fault Tolerance
FR	Federation Representative
FSM	Finite State Machine
FTA	Fault Tolerance Actuator
FTDU	Fault Tolerance Correlation Engine
FTP	File Transfer Protocol
GGF	Global Grid Forum
GIF	Graphics Interchange Format
GIS	Grid Information Service
GMA	Grid Monitoring Architecture; monitoring architecture defined by GGF
GRAM	Grid Resource Allocation Management
GriFIS	Grid Fabric Information Service
GSI	Grid Security Infrastructure
GUI	Graphical User Interface
HDF	Hierarchical Data Format
HEP	High Energy Physics
HLD	High-Level Description
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HW	Hardware
I/O	Input/Output
IDL	Interactive Data Language
ISDN	Integrated Services Digital Network
JDL	Job Description Language
LB	Logging and Bookkeeping

LCAS	Local Centre Authorization Service
LCMAPS	Local Credential MAPping Service
LDAP	Lightweight Directory Access Protocol
LLD	Low-Level Description
LON	Logical Object Name
LSF	Load Sharing Facility
MDS	Globus Meta-computing Directory Service
MLD	Machine Level Description
MR	Monitoring Repository
MS	Monitoring Sensor
MSA	Monitoring Sensor Agent
MSS	Mass Storage System
MUI	Monitoring User Interface
MySQL	Widely distributed SQL database open source implementation
NFS	Network File System
NIS	Network Information System
NMA	Node Management Agent
OS	Operating System
PDS	Payload Data Segment
PFN	Physical File Name
PKI	Public Key Infrastructure
PXE	Preboot eXecution Environment
QoS	Quality of Service
QoSS	Quality of Security Service
RB	Resource Broker
RC	Replica Catalog
RDBMS	Relational Database Management System
Replica	A copy of a file that is managed by the Grid middleware
RM	Replica Manager
RMS	Resource Management Subsystem
RPC	Remote Procedure Call
RSL	Resource Specification Language
SAN	Storage Area Network
SE	Storage Element
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SP	Software Package
SSL	Secure Sockets Layer
SW	Software
TFN	Transport File Name
TFTP	Trivial File Transfer Protocol
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
VO	Virtual Organization
VPN	Virtual Private Network
VRML	Virtual Reality Modeling language
WM	Workload Management
WWW	World Wide Web
XML	eXtensible Markup Language

Appendix

Selected Publications

1. **Naqvi S., Riguidel M., 'Dynamic Access Control for Pervasive Grid Applications', IEEE International Conference on Computational Intelligence and Security 2005 (IEEE-CIS05), Xi'an, China, December 15-19, 2005**

   The current grid security research efforts focus on static scenarios where access depends on the identity of the subject. They do not address access control issues for pervasive grid applications where the access privileges of a subject not only depend on their identity but also on their current context (i.e. current time, location, system resources, network state, etc). Our approach complements current authorization mechanisms by dynamically granting permission to users based on their current context. The underlying dynamic and context aware access control model extends the classic role based access control, while retaining its advantages (i.e. ability to define and manage complex security policies). The major strength of our proposed model is its ability to make access control decision dynamically according to the context information. Its dynamic property is particularly useful for pervasive grid applications.

2. **Naqvi S., Riguidel M., 'Grid Security Services Simulator (G3S) – A Simulation Tool for the Design and Analysis of Grid Security Solutions', IEEE International Conference on e-Science and Grid Computing 2005 (e-Science 2005), Melbourne, Australia, December 5-8, 2005**

   Security services are one of the most desirable characteristics of the computational grids. Nowadays the swelling number of applications and consequent increase in the amount of critical data over the grids has considerably raised the stakes for efficient security architecture. Establishing security solutions for computational grid remains in its initial stages, as there are a number of impediments in the way of successful implementation of these security designs on a real grid. Absence of suitable mechanism to simulate the various functionalities of grid security models is a major concern for security designers. A reliable simulator for the grid security services is indispensable so that the grid security solutions can be adequately tested before their implementation on a real grid. The available range of grid simulators does not provide any support for the security functions. This vacuity has overwhelmingly motivated us to develop the Grid Security Services Simulator (G3S).

3. **Naqvi S., Riguidel M., 'Trust Establishment in Pervasive Grid Environments', Cracow Grid Workshop 2005 (CGW'05), Krakow – Poland, November 20-23, 2005.**

   Pervasive grids are characterized by their global mobility feature that enable them to hook anywhere at anytime to the computing resources. Therefore, providing a dynamic and adaptive security model that overlays a secure framework over an untrustworthy network is one of the biggest challenges in Pervasive Grids. Current grid security enforcement and policy maintenance models are generally based on the assumption of a stable, static, and long-term grid establishment with a small set of seldom-changing users. Hence, these frameworks cannot be directly applied to the pervasive grid framework. Ad-hoc and federated grids require an adaptive security model that incrementally builds a secure grid community based on the notion of trust. In this paper, we have outlined our approach to handle the challenging problem of establishing trust in the pervasive grid environments where there is no a priori trust among its entities and no mechanism to build some trust based on a history of previous interactions.

4. **Naqvi S., Riguidel M., 'Security and Trust Assurances for Smart Environments', IEEE International Workshop on Resource Provisioning and Management in Sensors Network 2005 (RPMSN05), Washington DC, USA, November 7-10, 2005**

   Smart sensor networks increasingly become viable solutions to many challenging problems and will successively be employed in many areas in the near future. However, deploying a new technology without security and trust issues in mind has often proved to be unreasonably dangerous. We propose a security and trust vision for these smart environments. This paper provides the details of the concepts of Infospheres and Security Domains which lead to the phenomenon of virtualization of security services. We propose Virtual-to-Virtual (V2V) paradigm to solve the security and trust problems of the smart environments.

5. **Naqvi S., Riguidel M., 'Dynamic Distribution of Trust in the Grid Environments', The e- 2005 eChallenges Conference, Ljubljana, Slovenia, October 19-21 2005**

   The Grid vision is to allow computing resources to be shared and utilised globally, with these distributed resources belonging to the same Virtual Organisation (VO). These resources execute jobs submitted by users, who are not in the resources' local domain and hence have no control over these resources. Conversely these users are not controlled by the resource owners. Certificates provide a common, useful security mechanism to overcome these barriers and set out access rights, but they do not guarantee that the resources, or users, can be trusted. Resources and users may be unreliable; this situation may not be reflected in the users' perception of the reliability of the resource owner as a whole or vice versa.
   This paper describes a trust framework model for Grid computing, which enables users to execute their jobs on reliable and efficient resources, thereby satisfying clients' quality-of-service (QoS) requirements. We propose an optimistic trust model that provides probabilistic guarantees based on the status of the nodes. Nodes have the ability to revoke their relationships with malicious nodes and thus cause the trust values of wrong-doers to be reduced. The accuracy of the guarantee depends on how thoroughly each node can discover and validate the trust values of other nodes.

6. **Naqvi S., Riguidel M., 'Security Challenges for Highly Available Systems', IEEE International Carnahan Conference on Security Technology 2005 (IEEE ICCST2005), Las Palmas, Spain, October 11-14, 2005.**

   Nowadays Highly Available (HA) systems are a must for almost any business process. More recently, the need for HA systems has increased as electronic commerce and other internet-based applications have become widely used with the growing web usage. Security is a major concern for these systems. Companies want to make sure that their security systems are working flawlessly and efficiently. Making sure that these systems are available to allow the right people access to the right areas of the company is imperative. Traditionally, HA systems consist of proprietary hardware and software components. However, the price/performance advantages of commercial-off-the-shelf (COTS) based clusters have had a compelling affect on HA vendors and their marketplace. The emergence of Computational Grids makes it feasible to develop cost-effective, large-scale geographically distributed HA systems. Making sure that critical

applications on this new generation of HA systems are secured is a challenging proposition.
In this article, we have identified a list of challenges for the next generation of Grid-based HA systems. We have explored the virtualization of security services with their pluggable implementation to address the security needs of these Grid-based HA systems. The main advantages of this solution include independence with respect to the underlying security mechanisms; best trade-off between security guarantees and processing capabilities; configurability of security architecture; better portability across heterogeneous platforms; and a smaller application development cycle for the HA functionality in the system.

7. **Naqvi S., Riguidel M., 'Impact of Comprehensive Security Services on Grid Computing Performance', IEEE International Conference on Dependable Systems and Networks 2005 (IEEE-DSN2005), Yokohama, Japan, June 28 - July 1, 2005 (ISBN 0769522823)**

   The grid is no longer just a synonym for networked high performance computing. It is emerging as a bigger vision of flexible, secure, coordinated resource-sharing among dynamic collections of individuals, institutions, and resources. In the evolution of computational grids, security features were overlooked in the desire to implement a high performance distributed computational system. Thus there was no need to investigate the impact of in-depth security on the grid performance. But now the growing size and profile of the grid require comprehensive security solutions as they are critical to the success of the endeavor. Currently the real meaning of grid security performance is being explored as different research communities introduce novel approaches to the security performance monitoring and evaluation. With the emerging grid security solution comes the question how to measure the quality. This information is essential for the entire grid community. Yet, there is no widely accepted and deployed technique that can solve this problem. In this paper we have presented a study of the effects of in-depth security services on the performance of computational grids.

8. **Naqvi S., Riguidel M., 'Securing Grid-Based Critical Infrastructures', The IEEE Symposium on Intelligence and Security Informatics (IEEE ISI-2005), Atlanta, Georgia, USA, May 19-20, 2005 (ISBN 3540259996)**

   As the computing world has grown more dependent on the communications networks, the Grid computing is increasing the visibility of computer systems in the running of businesses, boosting the cost of system downtime; even short interruptions in the functioning of the Internet and other networks have become unacceptable. Consequently, Denial of Service (DOS) attacks that prevent access to online services are one of the greatest threats to the information infrastructure.
   When regarding the protection of the essential information infrastructures (and especially critical infrastructures), most of the time one concentrates on the availability subject. However, we put emphasis on not forgetting to protect integrity of provided services as well. Moreover, Service availability may conflict with other security goals that can be more fundamental in some infrastructure cases; when integrity and confidentiality are the main goals, the most secure system is often one that does nothing. Therefore, protection against DoS often requires architectural changes to the system, which may prove expensive.
   Another challenge for securing infrastructures is to make a trade-off between security and privacy. Technological developments and the needs of law enforcement provide increased opportunities for surveillance in cyberspace. Better managing and strengthening the infrastructure would make it more efficient and resilient without the

need for unnecessary surveillance. A typical aspect of this issue is the problem of attack trace-back in Internet between the security (detecting the attacker) and the privacy (protecting the anonymity of Internet users).

9. Naqvi S., Riguidel M., 'Designing Security Architecture for Large Scale, Open, Distributed Heterogeneous Systems', IEEE Symposium on Security and Privacy 2005 (IEEE-SP2005), Berkeley/Oakland, California - USA May 8-11, 2005.

In this work, we have proposed a security architecture to address the comprehensive security needs of today's large scale, open, distributed heterogeneous systems. Extensive groundwork was carried out to establish the real needs of the security architecture in order to reduce unnecessary overheads and to create robustness. These include requirements analysis, risk analysis, threat modeling, and implementation feasibility.

The concept of virtualization is introduced for the security services. This concept of virtualization of security services is needed to have the absolute freedom to choose the underlying security mechanisms. From the security point of view, the virtualization of a service definition encompasses the security requirements for accessing that service. The need arises in the virtualization of security semantics to use standardized ways of segmenting security components (e.g., authentication, access control, etc.) and to provide standardized ways of enabling the federation of multiple security mechanisms. Virtualization permits each participating end-point to express the policy it wishes to see applied when engaging in a secure conversation with another end-point. Policies can specify supported authentication mechanisms, required integrity and confidentiality, trust policies, privacy policies, and other security constraints. This concept of virtualization of security services can be realized through distributed virtual engines that will enable security service calls to be unified according to requirements and not according to the technologies to be supported.

A configurable mechanism for the invocation of security services is proposed to address security needs of the different kinds of users. This approach permits the evolution of security infrastructure with less impact on the resource management functionalities, which are still on the verge of evolution. Moreover, it permits the users and resource providers to configure the security architecture according to their requirements and satisfaction level. The set of these security services include core security services (such as authentication, authorization, identity mapping, audit, etc.) as well as contemporary security services (such as mobile access control, dynamic digital signature, etc.).

The concept of virtualization could be extended to adapt country-specific legal requirements, population-based ethical issues, and the business-oriented interests. Moreover, virtualization could be used to achieve the best trade-off between security guarantees and processing capabilities.

10. Naqvi S., Riguidel M., 'Addressing Secure Access Challenges for Nomadic Grid: A Hospital Case Study', Grid Asia Conference 2005, Biopolis, Singapore, May 2-6, 2005.

Despite the sensitivity of the health data and the growing threat, relatively little attention has been paid to the complexities of grid access constraints in middleware development. The scope of access control lists (ACLs) is limited and is inflexible in the mobile arena. Much attention has been given to encryption techniques but, while encryption is certainly important, it protects only the communication and authentication in the system. It provides only the basis for a secure access control mechanism.

We present a detailed examination of the dynamic access control challenges for a nomadic health grid with the aim of achieving optimal access rights for each of the involved principals. We show that the designing challenges are very complex and cannot be expressed easily or clearly using the static per-method access control lists generally supported by component-based software. We derive general requirements for the expressiveness of access challenges and propose criteria for a more suitable access control mechanism in the context of nomadic health grid applications. The results are based on a hospital case study. It first provides an overview of requirements for access control in a hospital environment. We proceed by superposing the requirements resulting from each paradigm. Conflicts are solved by giving priority to the functional requirements. The paper further presents our initial results and briefly discusses how it meets the requirements.

11. Naqvi S., Riguidel M., 'G3S: Grid Security Services Simulator', Health Grid Conference 2005 (HG2005), Oxford, UK, April 7-9, 2005

Security services are one of the most desirable characteristics of health grid. Nowadays the swelling number of applications and consequent increase in the amount of critical data over health grid has considerably raised the stakes for efficient security architecture. Establishing in-depth security solutions for health grid remains in its initial stages, as there are a number of impediments in the way of successful implementation of these security designs on a real grid. Absence of some suitable mechanism to simulate the various functionalities of grid security models is a major concern for security designers. The available range of grid simulators does not provide any support for the security functions. This vacuity has overwhelmingly motivated us to develop the grid security services simulator – G3S. Traditionally, system developers periodically release patches to overcome the shortcomings of their previous release. These patches are generally released when some vulnerability present in their product is successfully exploited. The same practice is, however, not feasible for health grid due to the scope of the applications and the nature of the stored data. Hence a reliable simulator for the security services is indispensable so that the grid security solutions be adequately tested before their implementation on a real health grid. G3S is the first milestone in this direction.

12. Naqvi S., Riguidel M., 'Performance Measurements of the VIPSEC Model', High Performance Computing Symposium (HPC 2005), San Diego, California - USA, April 3-7, 2005.

The grid computing paradigm offers both the availability of abundant computing resources, and the storage of increased amounts of valuable data. Such information systems heavily rely on the provision of adequate security. It is imperative that techniques be developed to assure trustworthiness of these environments. While security performance evaluation of parallel and distributed systems is well investigated and there exist practical solutions, in most cases these techniques cannot be transferred directly to the grids. Currently the real meaning of grid security performance is being explored as different research communities introduce novel approaches to the security performance monitoring and evaluation. With the emerging grid security solution comes the question how to measure the quality. This information is essential for the entire grid community. Yet, there is no widely accepted and deployed technique that can solve this problem. The authors have developed a basic simulator for their proposed grid security model VIPSEC: Virtualized and Pluggable Security Services. This article presents the preliminary simulation results of this model.

13. **Naqvi S., Riguidel M., 'Secure Data Exchange Between Intelligent Devices and Computing Centers', SPIE Defense and Security Symposium 2005 (SPIE-DSS2005), Orlando, Florida - USA, March 28-April 01, 2005.**

The advent of reliable spontaneous networking technologies (commonly known as wireless ad-hoc networks) has ostensibly raised stakes for the conception of computing intensive environments using intelligent devices as their interface with the external world. These smart devices are used as data gateways for the computing units. These devices are employed in highly volatile environments where the secure exchange of data between these devices and their computing centers is of paramount importance. Moreover, their mission critical applications require dependable measures against the attacks like denial of service (DoS), eavesdropping, masquerading, etc.
In this paper, we propose a mechanism to assure reliable data exchange between an intelligent environment composed of smart devices and distributed computing units collectively called 'computational grid'. The notion of infosphere is used to define a digital space made up of a persistent and a volatile asset in an often indefinite geographical space. We study different infospheres and present general evolutions and issues in the security of such technology-rich and intelligent environments. It is beyond any doubt that these environments will likely face a proliferation of users, applications, networked devices, and their interactions on a scale never experienced before. It would be better to build in the ability to uniformly deal with these systems. As a solution, we propose a concept of virtualization of security services. We try to solve the difficult problems of implementation and maintenance of trust on the one hand, and those of security management in heterogeneous infrastructure on the other hand.

14. **Naqvi S., Riguidel M., 'Threat Model for Grid Security Services', European Grid Computing Conference 2005 (EGC2005), Amsterdam, Netherlands, February 14-16, 2005.**

The grid computing paradigm involves both the availability of abundant computing resources, and the storage of increased amounts of valuable data. Such information systems heavily rely upon the provision of adequate security. It is imperative that techniques be developed to assure the trustworthiness of these environments.
Formal verification provides the tools and techniques to assess whether systems are indeed trustworthy, and is an established approach for security assurance. When using formal verification for security assessment one of the most important concerns should be to be precise about the threat model. A comprehensive threat model is indispensable for the simulations of a grid security model. This article presents a survey of the various threat models and discusses how and when these threat models may be inappropriate for use in the grid computing environments. Then a fine-grained threat model for grid computing is presented.

15. **Naqvi S., Riguidel M., 'Problems in the Implementation of Grid Security Services', Cracow Grid Workshop 2004 (CGW'04), Krakow – Poland, December 12-15, 2004.**

Security services were overlooked in the early stages of the grid evolution when the grid community was composed of dedicated computing researchers and the data was non-critical. Nowadays the swelling number of grid applications and consequent increase in the amount of critical data over grid has considerably raised the stakes for an efficient security architecture. Establishing in-depth security solutions for grid remains in its initial

stages, as there are a number of impediments in the way of successful implementation of these security designs on a real grid. These problems have to be overcome in order to make the grid endeavor successful.
System developers periodically release patches to overcome the shortcomings of their previous release. These patches are generally released when some vulnerability present in their product is successfully exploited. The same practice is, however, not feasible for the grids due to the scope of the applications and the nature of the data stored over it. Certain grid applications like healthcare, where a patient's data has to be protected throughout its lifecycle, require a truly dependable security mechanism. In such applications the loss of information is irreversible and hence a well-designed security mechanism is required to persuade the already sceptical potential users to participate in the global computing environment.
This article presents a thorough analysis of the various problems faced by the designers and developers of grid security solutions. These problems range from the non-availability of an adequate mechanism to simulate the grid security services to the grid specific constraints for the implementation of rigorous security solutions. The impact of these problems on the pace of the development of the grid security technologies is outlined and subsequently some remedial solutions are presented. Grid community's lack of experience in the exercise of the Common Criteria (CC), which was adopted in 1999 as an international standard for security product evaluation is also discussed, as the evaluation of grid security solutions requires excellent criteria to assure sufficient security to meet the needs of its users and resource providers.

16. Naqvi S., Riguidel M., 'Security Architecture for Heterogeneous Distributed Computing Systems', IEEE International Carnahan Conference on Security Technology 2004 (IEEE ICCST2004), Albuquerque, New Mexico - USA, October 11-14, 2004.

Distributed systems face a proliferation of users, applications, networked devices, and their interactions on a scale never experienced before. The advent of reliable spontaneous networking technologies has ostensibly raised the stakes for the design of computing intensive environments using intelligent devices. As environmental intelligence grows so will the number of heterogeneous devices connected to the environment. The creation of security and trust paradigms for such technology rich environments is today's great challenge. If the intelligent devices present in a smart environment act as gateways to some huge distributed computing system, then it is indispensable to sweep the threats out from these smart environments, so as to protect not only the local environment, but also the entire distributed system.
This article proposes a design of consistent but fine-grained levels of trust and security in distributed systems, open to pervasive, mobile, heterogeneous networks featuring ambient intelligence by gradually virtualizing their security functions. These systems interact in various ways, with floating semantic interoperability between applications, interoperability of communications depending on shared links between those systems, and versatile interconnections. Threats and vulnerability vary according to different systems, objects, applications, and communication links. The salient features of this design include: consideration of duration and time factors in cryptographic protocols by introducing a trusted clock in the network; space for the security of distributed environments by context awareness in the system; mobility (security of mobile code, mobile agents and speed of movement); virtualization of security services.

17. **Naqvi S., Riguidel M., 'Evaluation of Grid Security Solutions using Common Criteria', Computing in High Energy Physics 2004 (CHEP'04), Interlaken - Switzerland, September 27- October 01, 2004.**

In the evolution of computational grids, security threats were overlooked in the desire to implement a high performance distributed computational system. But now the growing size and profile of the grid require comprehensive security solutions as they are critical to the success of the endeavor. A comprehensive security system, capable of responding to any attack on grid resources, is indispensable to guarantee its anticipated adoption by both the users and the resource providers. Some security teams have started working on establishing in-depth security solutions. The evaluation of their grid security solutions requires excellent criteria to assure sufficient security to meet the needs of its users and resource providers. Grid community's lack of experience in the exercise of the Common Criteria (CC), which was adopted in 1999 as an international standard for security product evaluation, makes it imperative that efforts be exerted to investigate the prospective influence of the CC in advancing the state of Grid security. This article highlights the contribution of the CC to establishing confidence in grid security, which is still in need of considerable attention from its designers. The process of security evaluation is outlined and the roles each part of the evaluation may play in obtaining confidence are examined.

18. **Naqvi S., Riguidel M., 'VIPSEC: Virtualized and Pluggable Security Services Infrastructure for Adaptive Grid Computing', Proceedings of IEEE International Symposium on Network Computing and Applications (IEEE NCA04), Cambridge, Massachusetts - USA, August 30 - September 01, 2004 (ISBN 0769522424)**

Large scale distributed systems like the computational Grid combine network access with multiple computing and storage units. The need for efficient and secure data transportation over potentially insecure channels creates new security and privacy issues, which are exacerbated by the heterogeneous nature of the collaborating resources. Traditional security approaches require adequate overhauling to address these paradigms. In this paper, we propose a new two-pronged approach to address Grid security issues. First, the virtualization of security services provides an abstraction layer on the top of the security infrastructure, which harmonizes the heterogeneity of underlying security mechanisms. Second, the pluggable nature of the various security services permits the users and resource providers to configure the security architecture according to their requirements and satisfaction level. This approach allows the security infrastructure to develop with minimal impact on the Grid resource management functionalities, which are still being developed.

19. **Naqvi S., Riguidel M., Demeure I., 'Security Architecture for Health Grid using Ambient Intelligence', Health Grid Conference 2004 (HG2004), Clermont-Ferrand – France, January 29-30, 2004.**
    → published in the Special Grid Issue of Methods of Information in Medicine (MIM) vol. 44, May 2005, pp 202-206 (ISSN 0026-1270)

Security concerns are severely impeding the grid community effort to spread its wings in health applications. In this paper, we have proposed a high level approach to incorporate ambient intelligence for health grid security architecture and have argued that this will significantly improve the current state of the grid security paradigm with an enhanced user-friendly environment. We believe that the time is ripe to shift the onus of traditional

security mechanisms onto the new technologies. The incorporation of ambient intelligence in the security architecture of a grid will not only render a security paradigm robust but also provide an attractive vision for the future of computing by bringing the two worlds together.

20. Naqvi S., Riguidel M., 'Security Risk Analysis for Grid Computing', Proceedings of Cracow Grid Workshop 2003 (CGW'03), Krakow – Poland, pp 174-189, October 27-29, 2003. (ISBN 8391514137)

The security and privacy issues are coming to the fore with the growing size and profile of the grid community. The forthcoming generations of the computational grid will make available a huge number of computing resources to a large and wide variety of users. The diversity of applications and mass of data being exchanged across the grid resources will attract the attention of hackers to a much higher extent. A comprehensive security system, capable of responding to any attack on its resources, is indispensable to guarantee the anticipated adoption of grid by both the grid users and the resource providers. In this article, we argue that the first brick of an effective plan of countermeasures against these threats is an analysis of the potential risks associated with grid computing.

This article presents a pragmatic analysis of the vulnerability of existing grid systems and the potential threats posed to their resources once their spectrum of users is broadened. Various existing grid projects and their security mechanisms are reviewed. The experience of using common grid software and an examination of grid literature served as the basis for this analysis. Legal loopholes in the implementation of grid applications across the geopolitical frontiers, and the ethical issues that could obstruct the wide acceptance and trustworthiness of grids are also discussed. The weaknesses revealed are classified with respect to their sources and possible remedies are discussed. The results show that the main reason for the vulnerability is the fact that grid technology has been little used except by a certain kind of public (mainly academics and government researchers). This public benefit greatly from being able to share resources on the grid, and have no intention of harming the resource owners or fellow users. Thus there was no need to address security in depth. This is all about to change. The number of people who know about the grid is growing fast, as are the worthwhile targets for the potential attackers. The security nightmare can not be avoided unless the problem is addressed urgently. This detailed taxonomy of potential threats and the sources of vulnerability in the existing grid architectures is the first milestone on the road to a robust grid security system. It provides a comprehensive overview which shall enable us to effectively plan the countermeasures against the existing risk. Our future direction includes the definition of a Protection Profile (Common Criteria) followed by the formulation of a comprehensive security policy and finally its implementation.