Vers un système d'administration de la sécurité pour les réseaux autonomes

Mohamad Aljnidi

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Thèse

présentée pour obtenir le grade de docteur de

TELECOM ParisTech

l'École Nationale Supérieure des Télécommunications

Spécialité : Informatique

Mohamad ALJNIDI

Vers un système d'administration de la sécurité pour les réseaux autonomes

Soutenue le 14 décembre 2009 devant le jury composé de

Isabelle CHRISMENT	**Rapporteurs**
Hatem BETTAHAR	
Dominique GAÏTI	**Examinateurs**
Christophe BIDAN	
Pascal URIEN	
Jean LENEUTRE	**Directeur de thèse**

Résumé étendu

L'administration de la sécurité dans les réseaux sans infrastructures est assez complexe pour être réalisée par des êtres-humains. Notre objectif est de la rendre autonome. Dans ce cadre, cette thèse propose un système autonome de contrôle d'accès.

On fixe la définition d'un réseau autonome, et certaines bases de la sécurité autonome. Ensuite, on définit un type de réseau autonome, organisationnel et sans infrastructure, et on l'appelle IOrg-AutoNet. Les nœuds d'un IOrg-AutoNet sont catégorisés en terme de fiabilité, disponibilité et capacités, ce qui leur permet d'acquérir différents rôles, dont certains donnent droit à coopérer avec d'autres nœuds pour gérer le réseau.

On définit pour les IOrg-AutoNets un modèle de contrôle d'accès basé sur les relations sécurisées, et on l'appelle SRBAC. Ses politiques sont à appliquer quand deux nœuds, déjà reliés par une relation sécurisée leur spécifiant des rôles, se communiquent. Le modèle SRBAC est une version adaptée du modèle RBAC. On propose aussi une extension du profile RBAC du langage XACML v2.0 pour la spécification des politiques SRBAC.

On définit pour SRBAC le modèle administratif associé ASRBAC pour parvenir à notre système autonome de contrôle d'accès. Le modèle ASRBAC est une extension du modèle administratif distribué ARBAC02 qui est associé au modèle RBAC. Cette extension rajoute des aspects contextuels, cognitifs, adaptatifs, coopératifs et autonomes. ASRBAC est basé sur SRBAC lui-même, ce qui représente la base du comportement autonome.

Un exemple d'un système SRBAC/ASRBAC d'un réseau domestique et une conception préliminaire d'un système de réalisation valident et matérialisent nos contributions.

a) Motivation et objectifs

Avec les technologies de nos jours, il est rapide et facile de construire, implémenter et/ou utiliser différents types de réseaux de télécommunications. Cette facilité d'utilisation continuera de s'améliorer, et encore plus de technologies de haut niveau apparaitraient. Ceci amènera à plusieurs types d'applications entrainant des réseaux complexes, hétérogènes, décentralisés et construits ou étendus par des utilisateurs non-experts. De tels réseaux ont besoin d'être capables de se gérer, car l'intervention humaine pour l'administration d'un réseau est prévue d'être de plus en plus indésirable, chère, inefficace et/ou une perte de temps, selon le champ d'application.

Dans le contexte de la sécurité, les réseaux auto-gérables doivent être capables de se protéger, et de protéger leurs ressources, contres des attaques possibles internes ou externes. Dans le cas d'une attaque réussie, un tel réseau doit être capable de résoudre par

elle-même les problèmes résultants. Autrement dit, le système de la sécurité d'un réseau auto-gérable doit être un système auto-gérable. En plus des propriétés de l'autoprotection et l'autoréparation qu'un tel système fournit au réseau dans lequel il est implémenté, il doit être capable de reconfigurer lui-même et de s'optimiser.

Pour réaliser les propriétés de l'autoprotection, l'autoréparation, l'auto-configuration et l'auto-optimisation, un système informatique a besoin d'un ensemble de règles précisant quand et comment réagir, en utilisant quels outils et sur quelles ressources. Autrement dit, un système auto-gérable a besoin de politiques. Selon le champ d'application, il se peut qu'un système auto-gérable ait besoin d'appliquer des politiques spécifiées directement par des utilisateurs finaux en forme d'objectifs de haut niveau. Il doit être capable de réaliser de tels politiques de haut niveau, et d'appliquer des opérations d'autogestion sur elles pour les adapter aux changements dans son contexte.

Notre objectif général est de proposer des solutions de sécurité propres aux réseaux autonomes. La nécessité des solutions d'autogestion avait été constaté par plusieurs spécialistes dans différents domaines pertinents, et plusieurs initiatives correspondantes ont été lancées, comme expliqué dans [55] et [41]. Notre travail est basé sur l'initiative d'IBM [51] qui se porte sur l'informatique autonome (Autonomic Computing). Dans notre recherche, nous étudions la réalisation de la vision de l'informatique autonome [56] pour construire une plateforme d'un système de sécurité autonome pour des réseaux sans infrastructures. Nous croyons qu'une telle plateforme serait une étape **Vers un système d'administration de la sécurité pour les réseaux autonomes**.

Plus précisément, on aimerait par cette thèse établir les bases pour un système autonome de contrôle d'accès dans des réseaux sans infrastructures. *Pourquoi le contrôle d'accès ?* Parce que c'est un domaine où on pourra étudier la plupart des concepts de l'**informatique autonome** en termes de sécurité, et plus particulièrement, la capacité de maintenir des politiques contextuelles cohérentes, ce qui est pointé tout au long de la thèse et élaboré dans le Chapitre 5. *Pourquoi les réseaux sans infrastructures ?* Parce qu'ils présentent plusieurs soucis de complexité, dont le manque d'infrastructure préétablie, la possibilité d'évolution de la topologie et l'hétérogénéité des nœuds, ce qui favorise le besoin de l'autogestion et rend ces réseaux les meilleurs candidats pour être des réseaux autonomes, comme expliqué dans le Chapitre 3.

b) Structure de la thèse

La thèse est divisée en trois parties. Elle parvient à ses objectifs à la fin de la deuxième partie, après que la première partie ne met le lecteur dans le contexte exact de la recherche présentée, et avant que la troisième partie n'essaye de lui convaincre de la faisabilité des solutions proposées. Dans la première partie, une base théorique est fournie avant de donner notre point de vue sur les réseaux autonomes et leur sécurité, et éventuellement la définition de notre modèle réseau. La deuxième partie introduit d'abord un modèle de contrôle d'accès pour les réseaux autonomes, et propose ensuite les bases d'un système autonome associé pour l'administration du contrôle d'accès. Une étude de cas détaillée, prenant un réseau domestique comme exemple, est fournie dans la troisième partie pour

clarifier les concepts de base de nos travaux, et pour les comparer avec des travaux concurrents. En plus, cette dernière partie donne une preuve de concept en forme d'un modèle de réalisation d'un prototype d'un système autonome de contrôle d'accès.

Contexte : L'objectif de la première partie de la thèse est de préciser le contexte dans lequel nous proposons nos solutions. Nous présentons nos visions du réseau autonome et de la sécurité autonome à l'issue d'un état de l'art au début de cette partie. Nous introduisons ensuite des bases d'une plateforme correspondante d'un système autonome de sécurité. A la fin de cette première partie, nous aurons proposé une structure organisationnelle variable pour les réseaux sans infrastructures. Cette structure permet à un réseau et à son système d'administration de la sécurité d'être autonomes.

Solution : Le modèle de réseau autonome et la plateforme de sécurité autonome, qui ont été introduits dans la première partie, présentent des bases pour plusieurs challenges de recherche. Afin de favoriser l'aspect important de gestion de politiques dans les systèmes informatiques autonomes, ainsi que différents autres aspects intéressants, nous avons choisis de travailler sur un système de contrôle d'accès pour les réseaux autonomes. Nous détaillons notre solution dans la deuxième partie. Il s'agit d'un modèle de contrôle d'accès et son modèle administratif. Ils sont proposés comme des fondations pour un système de sécurité autonome pour les réseaux sans infrastructures.

Faisabilité : La solution de contrôle d'accès que nous proposons dans la deuxième partie est décentralisée, contextuelle, dynamique et collaborative. Plusieurs solutions existantes ont ces propriétés aussi. Cependant, notre solution est plus dynamique et plus flexible, et surtout, elle a l'avantage d'être auto-conscient, auto-gérable et auto-adaptable à la fois. Ces caractéristiques nombreuses ont besoin d'être clarifiées et validées en termes de réalisation. Dans la troisième partie, nous présentons un cas d'étude sur un réseau domestique autonome, et ensuite nous introduisons un prototype en forme d'une architecture de réalisation, et des considérations d'implémentation, pour la mise en place du modèle de politique défini dans la deuxième partie.

c) Contributions

Comme indiqué par le titre de la thèse "**Vers un système d'administration de la sécurité pour les réseaux autonomes**", nous visons des contributions dans deux domaines, étant "l'administration de la sécurité" et "les réseaux autonomes". Plus particulièrement, il se trouve que nos contributions soient dans le domaine de "la sécurité autonome". Nous présentons un ensemble de concepts et de définitions dans cette discipline, mais nous essayons essentiellement de contribuer dans le domaine du "contrôle d'accès". Cependant, il était important de commencer le titre de notre thèse par le mot "**Vers**". C'est parce que nous ne prétendons pas que nous proposons une solution complète pour un système de sécurité autonome. Plus précisément, cette thèse propose des solutions de base pour un tel système, que nous croyons indispensable pour les réseaux futurs.

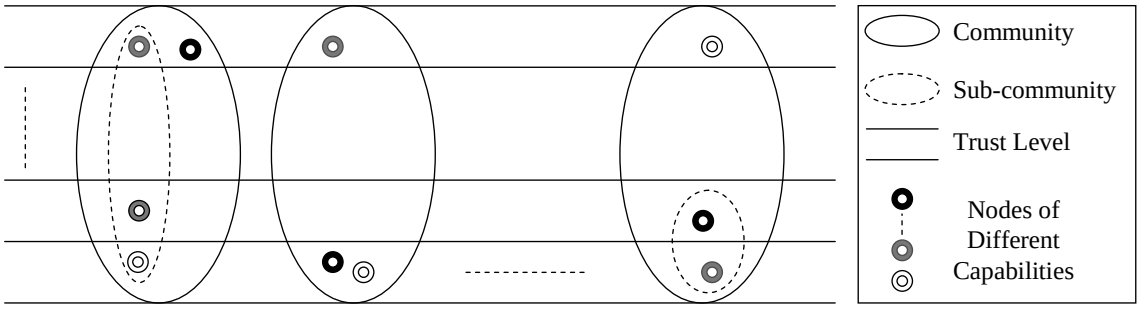


Figure 1: Modèle IOrg-AutoNet (Réseau autonome organisationnel sans infrastructure)

c.1) Réseaux autonomes

Une première contribution, petite mais essentielle, est la définition générique suivante des réseaux autonomes :

Définition 1 *Un réseau autonome est un réseau qui intègre un ensemble de systèmes autonomes, pour lesquels les nœuds du réseau sont des agents qui coopèrent pour reconfigurer, protéger, réparer et optimiser le réseau, et les systèmes autonomes eux-mêmes.*

En nous basant sur la Définition 1, nous introduisons un ensemble de caractéristiques définissant un **Réseau autonome organisationnel sans infrastructure**, que nous appelons **IOrg-AutoNet** (**I**nfrastructureless **Org**anizational **Auto**nomic **Net**work), et dont le modèle est illustré par la Figure 1 et décrit par la définition formelle suivante :

Définition 2 *Le modèle IOrg-AutoNet est défini par un tuple* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ *:*

- $\mathcal{N}$ *est un ensemble de nœuds d'un réseau.*
- $\mathcal{T}$ *est un ensemble de niveaux de confiance totalement ordonné, ayant* L *et* H *comme bornes inférieure et supérieure respectivement.*
- $\mathcal{C}$ *est un ensemble de communautés d'un réseau vérifiant les conditions suivantes :*
  - $\mathcal{C} \subseteq 2^{\mathcal{N}} \setminus \{\emptyset\}$
  - $\bigcup_{c \in \mathcal{C}} c = \mathcal{N}$
  - $\forall c1, c2 \in \mathcal{C}, (c1 \cap c2 \neq \emptyset) \Leftrightarrow ((c1 \subseteq c2) \vee (c2 \subseteq c1))$
- $\mathcal{K}$ *est un ensemble de classes de capacités totalement ordonné, ayant* LD *et* HD *comme bornes inférieure et supérieure respectivement.*
- $\mathcal{F}$ *est l'ensemble des fonctions suivantes :*
  - $tLevel : \mathcal{N} \longrightarrow \mathcal{T}$ *renvoie le niveau de confiance d'un nœud.*
  - $nComm : \mathcal{N} \longrightarrow \mathcal{C}$*, renvoie la communauté d'un nœud.*

$* \ \forall x \in \mathcal{N}, nComm(x) = \bigcap_{\{c \in \mathcal{C} | x \in c\}} c$

– $cClass : \mathcal{N} \longrightarrow \mathcal{K}$ *renvoie la classe de capacité d'un nœud.*

- *NC est un ensemble de catégories de nœuds vérifiant les conditions suivantes :*

  – $NC \subseteq (\mathcal{T} \cup \{nil_{\mathcal{T}}\}) \times (\mathcal{C} \cup \{\mathcal{N}\}) \times (\mathcal{K} \cup \{nil_{\mathcal{K}}\})$

  – $((t, c, k) \in NC) \Leftrightarrow (((t, c, k) = (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}})) \vee$
  $((c = \mathcal{N}) \wedge (k = nil_{\mathcal{K}}) \wedge (\exists x \in \mathcal{N}, tLevel(x) = t)) \vee$
  $((k = nil_{\mathcal{K}}) \wedge (\exists x \in \mathcal{N}, tLevel(x) = t, nComm(x) = c)) \vee$
  $(\exists x \in \mathcal{N}, tLevel(x) = t, nComm(x) = c, cClass(x) = k))$

Notation 1 *On utilise dans la Définition 2, ainsi que dans le reste de ce résumé :*

- *x, y et z, avec des indices éventuels, pour représenter un nœud (un élément de $\mathcal{N}$).*
- *t, avec des indices éventuels, pour représenter un niveau de confiance (un élément de $\mathcal{T}$).*
- *c, avec des indices éventuels, pour représenter une communauté (un élément de $\mathcal{C}$).*
- *k, avec des indices éventuels, pour représenter une classe de capacité (un élément de $\mathcal{K}$).*

Remarque 1 *Dans la Définition 2 :*

- *$\mathcal{N}$, $\mathcal{C}$, $\mathcal{F}$ et NC sont des ensembles variable qui s'adaptent à l'évolution du réseau.*
- *$\mathcal{T}$ et $\mathcal{K}$ sont des ensembles prédéfinis dans un IOrg-AutoNet donné.*
- *Il se peut que $\mathcal{N}$ soit un élément dans $\mathcal{C}$. Dans un tel cas, le réseau est composé d'une seule communauté, et de ses sous-communautés éventuelles.*
- *$\forall x \in \mathcal{N}, nComm(x)$ renvoie la sous-communauté la plus profonde (le sous-ensemble de nœuds le plus petit) à laquelle x appartient.*
- *Les constantes $nil_{\mathcal{T}}$ et $nil_{\mathcal{K}}$ sont utilisées dans l'ensemble de catégories de nœuds NC pour classifier un nœud en négligeant son niveau de confiance et sa classe de capacité respectivement.*
- *L'ensemble NC est définie pour distribuer les nœuds d'un IOrg-AutoNet sur des niveaux de confiances d'abord, ensuite sur les communautés du réseau et leurs sous-communautés à toutes les profondeurs, et finalement sur des classes de capacités.*

Le modèle IOrg-AutoNet est essentiellement caractérisé par une structure organisationnelle évolutive et son schéma d'évolution. Les définitions suivantes décrivent les différentes transitions pouvant prendre lieu pendant l'évolution d'un IOrg-AutoNet :

Définition 3 *Un IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ *peut évoluer au niveau des nœuds, et il devient* $(\mathcal{N}', \mathcal{T}, \mathcal{C}', \mathcal{K}, \mathcal{F}', NC)$*, selon les transitions suivantes :*

- *nodeInsertion*(x, c) *: insertion ou réinsertion d'un nœud* x *dans une communauté* $c \in \mathcal{C}$ *:*
  - $\mathcal{N}' = \mathcal{N} \cup \{x\}$
  - $\mathcal{C}' = \{c1 \cup \{x\} \mid c1 \in \mathcal{C}, c \subseteq c1\} \cup \{c2 \in \mathcal{C} \mid \neg(c \subseteq c2)\}$
  - $tLevel(x) = L$
- *nodeRemoval*(x) *: suppression ou bannissement d'un nœud* x *:*
  - $\mathcal{N}' = \mathcal{N} \setminus \{x\}$
  - $\mathcal{C}' = \{c1 \setminus \{x\} \mid c1 \in \mathcal{C}, x \in c1\} \cup \{c2 \in \mathcal{C} \mid x \notin c2\}$

Remarque 2 *Dans la Définition 3 :*

- *Le niveau de confiance initial d'un nouveau nœud est le niveau de confiance le plus bas* L*, ce qui peut être modifié plus tard par un système de réputation que nous supposons intégré dans un IOrg-AutoNet pour gérer la fiabilité des nœuds.*

Définition 4 *Un IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ *peut évoluer au niveau des communautés, et il devient* $(\mathcal{N}', \mathcal{T}, \mathcal{C}', \mathcal{K}, \mathcal{F}, NC')$*, selon les transitions suivantes :*

- *communityIntegration*(c) *: intégration d'une communauté* c *:*
  - $\mathcal{N}' = \mathcal{N} \cup c$
  - $\mathcal{C}' = \mathcal{C} \cup \{c\}$
  - $NC' = (NC \cup \{(t, c, nil_{\mathcal{K}}) \mid \exists x \in c, tLevel(x) = t\})$
    $\cup \{(t, c, k) \mid \exists x \in c, tLevel(x) = t, cClass(x) = k\}$
- *communityRevocation*(c) *: révocation d'une communauté* c *:*
  - $\mathcal{N}' = \bigcup_{\{c_i \in \mathcal{C} \mid \neg(c_i \subseteq c)\}} c_i$
  - $\mathcal{C}' = \mathcal{C} \setminus \{c' \in \mathcal{C} \mid c' \subseteq c\}$
  - $NC' = (NC \setminus \{(t, c1, nil_{\mathcal{K}}) \mid t \in \mathcal{T}, c1 \in \mathcal{C}, c1 \subseteq c\})$
    $\setminus \{(t, c2, k) \mid t \in \mathcal{T}, c2 \in \mathcal{C}, c2 \subseteq c, k \in \mathcal{K}\}$

Remarque 3 *Dans la Définition 4 :*

- *L'intégration d'une communauté implique la création d'un ensemble de catégories de nœuds correspondant aux différents niveaux de confiance déjà utilisés, et pour chacune de ces nouvelles catégories de nœuds, on en crée un ensemble correspondant aux différentes classes de capacité déjà utilisées.*

- *La révocation d'une communauté implique la suppression de tout élément dans NC correspondant à celle-ci où à l'une de ses sous-communautés.*

Définition 5 *En utilisant les transitions de la Définition 4, on est capable de décrire la fusion et la division des communautés dans un IOrg-AutoNet $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$, qui pourrait par conséquent devenir $(\mathcal{N}', \mathcal{T}, \mathcal{C}', \mathcal{K}, \mathcal{F}, NC')$, selon les transitions suivantes :*

- *$communityMerging(c1, c2, c)$: fusion des deux communautés $c1$ et $c2$ en une seule communauté c :*
  - $c = c1 \cup c2$
  - $communityIntegration(c)$
  - $communityRevocation(c1)$
  - $communityRevocation(c2)$
- *$communitySplitting(c, c1, c2)$: division de la communauté c en deux communautés $c1$ et $c2$:*
  - $\forall c' \in \mathcal{C}, c' \subset c$, $communityRevocation(c')$
  - $c1 \subset c, c2 \subset c, c2 = c \setminus c1$
  - $communityIntegration(c1)$
  - $communityIntegration(c2)$
  - $\mathcal{C}' = \mathcal{C} \setminus \{c\}$
  - $NC' = (NC \setminus \{(t, c, nil_{\mathcal{K}}) \mid t \in \mathcal{T}\}) \setminus \{(t, c, k) \mid t \in \mathcal{T}, k \in \mathcal{K}\}$

Remarque 4 *Dans la Définition 5 :*

- *En fusionnant deux communautés $c1$ et $c2$, toute les sous-communautés de $c1$ et $c2$ seraient révoquées éventuellement. Le résultat est une communauté à laquelle appartiennent tous les nœuds de $c1$ et $c2$ sans aucune sous-communauté.*
- *Avant de diviser une communauté, ses sous-communautés devraient être révoquées. Le résultat est deux communautés différentes sans aucune sous-communauté.*

Définition 6 *Un IOrg-AutoNet $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ peut évoluer au niveau du réseau, et il devient $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}', NC')$, selon les transitions suivantes :*

- *$trustChange(x, t)$: affectation du niveau de confiance t au nœud x, ce qui peut être une première utilisation de t :*
  - $NC' = ((NC \cup \{(t, \mathcal{N}, nil_{\mathcal{K}}) \mid \forall y \in \mathcal{N}, tLevel(y) \neq t\})$
    $\cup\{(t, c, nil_{\mathcal{K}}) \mid c \in \mathcal{C}, \forall y \in \mathcal{N}, tLevel(y) \neq t\})$
    $\cup\{(t, c, k) \mid c \in \mathcal{C}, \exists x \in \mathcal{N}, cClass(x) = k, \forall y \in \mathcal{N}, tLevel(y) \neq t\}$
  - $tLevel(x) = t$

- *capabilityChange*(x, k) *: affectation de la classe de capacité* k *au nœud* x*, ce qui peut être une première utilisation de* k *:*
  - $NC' = NC \cup \{(t, c, k) \mid \exists x \in \mathcal{N}, tLevel(x) = t, c \in \mathcal{C}, \forall y \in \mathcal{N}, cClass(y) \neq k\}$
  - $cClass(x) = k$

Définition 7 *On définit les transitions suivantes pour fusionner ou diviser un IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ *:*

- *networkMerging*$(\mathcal{N}', \mathcal{T}', \mathcal{C}', \mathcal{K}', \mathcal{F}', NC')$ *: fusion du réseau avec un autre réseau* $(\mathcal{N}', \mathcal{T}', \mathcal{C}', \mathcal{K}', \mathcal{F}', NC')$ *:*
  - *Le réseau résultant est :* $(\mathcal{N} \cup \mathcal{N}', \mathcal{T} \cup \mathcal{T}', \mathcal{C} \cup \mathcal{C}', \mathcal{K} \cup \mathcal{K}', \mathcal{F} \cup \mathcal{F}', NC \cup NC')$
- *networkSplitting*() *: diviser le réseau en deux :*
  - *Le réseau se divise en* $(\mathcal{N}1, \mathcal{T}, \mathcal{C}1, \mathcal{K}, \mathcal{F}1, NC1)$ *et* $(\mathcal{N}2, \mathcal{T}, \mathcal{C}2, \mathcal{K}, \mathcal{F}2, NC2)$

Remarque 5 *Dans la Définition 7 :*

- *Après une transition de fusion de réseaux, une série de transitions de fusion de communautés pourrait être lancée selon les nouvelles conditions de disponibilité. Cela serait décidé par un système de gestion de ressources que nous supposons intégré dans un IOrg-AutoNet pour gérer la disponibilité des nœuds.*
- *Avant une transition de division d'un réseau, une série de transitions de divisions de communautés pourrait être décidé et lancé par le système de gestion de ressources selon les besoins de la division.*

Nous considérons que Le modèle IOrg-AutoNet est une contribution dans le domaine des réseaux autonomes. Nous avons déjà présenté nos premiers essais pour définir les réseaux autonomes et un modèle organisationnel évolutif correspondant dans des publications introductives [8, 9].

c.2) Sécurité autonome

Nous présentons une vision de la sécurité autonome, selon laquelle un système autonome de sécurité fonctionne comme le système nerveux et le système d'immunité du corps humain, pour des fins de protection contre des attaques et de réparation après des attaques réussies respectivement. Selon cette même analogie, un système autonome de sécurité devient plus protégé et plus immune en optimisant ses politiques avec le temps. Également, tout comme le corps humain, il se peut qu'un expert externe intervienne pour réparer le système autonome de sécurité et optimiser ses politiques s'il n'y parvient pas tout seul parfois.

Dans le contexte d'une plateforme d'un système autonome de sécurité, nous présentons ensuite les bases pour deux contributions. Une première contribution est une proposition d'une architecture intra-nœud de sécurité autonome, illustrée dans la Figure 2. Cette architecture prendrait en compte l'hétérogénéité possible des nœuds dans un IOrg-AutoNet,

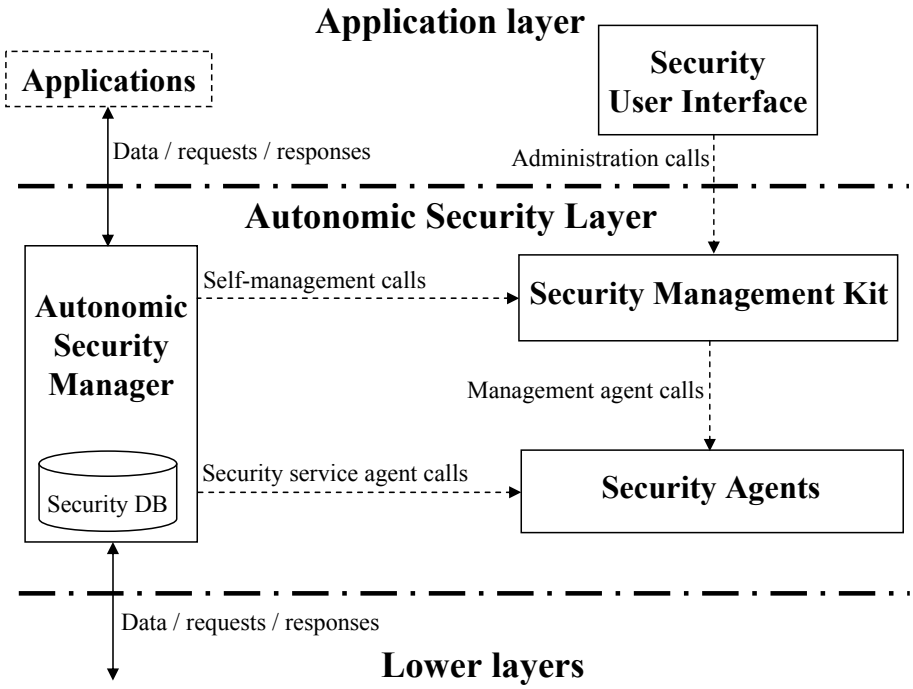
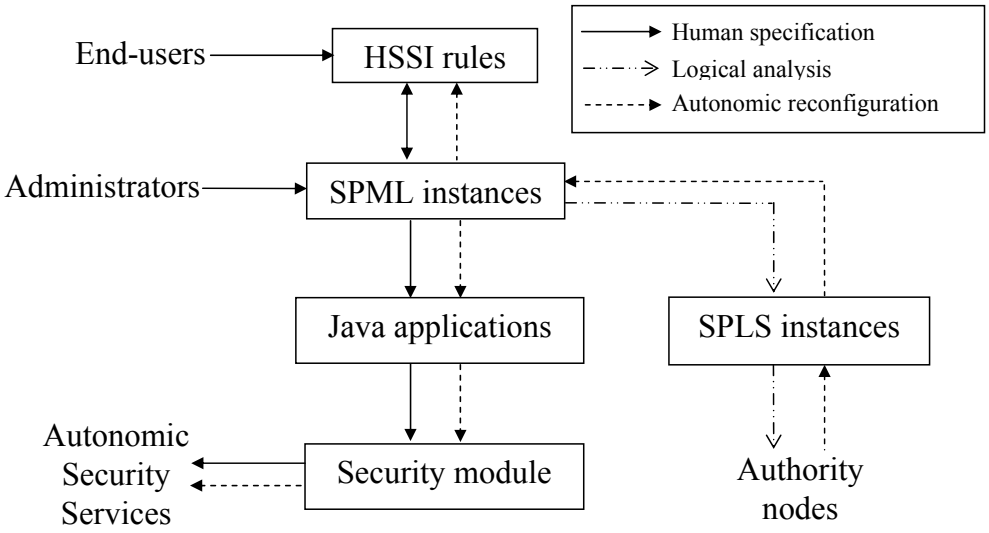


Figure 2: Architecture intra-nœud de sécurité autonome

et elle serait transparente pour l'utilisateur final et indépendante des technologies sous-jacentes de connexion. La deuxième contribution dans ce cadre est plus liée à la solution que nous proposons dans cette thèse, qui est un système autonome d'administration de contrôle d'accès. Il s'agit d'un ensemble de bases d'un système autonome de politiques de sécurité, illustré dans la Figure 3, permettant l'analyse et l'optimisation des règles de bas niveau de sécurité. Nous avons déjà motivé l'architecture intra-nœud de sécurité autonome et le système autonome de politiques de sécurité dans des publications introductives [10, 11].

Comme indiqué dans la Figure 3, nous précisons trois sources de modification pour une politique de sécurité dans un réseau autonome, étant les utilisateurs finaux, les administrateurs et les nœuds d'autorité. Les utilisateurs finaux pourraient changer leurs objectifs de haut niveau de sécurité. Les administrateurs pourraient intervenir pour gérer les politiques dans des cas exceptionnels. Les nœuds d'autorité sont censés analyser, négocier et reconstruire les politiques de sécurité, ou bien des parties de ces politiques, dans le cadre des opérations autonomes. Cette dernière fonctionnalité serait essentielle dans un IOrg-AutoNet, car l'intervention humaine dans la gestion des politiques doit être minimale dans un système autonome.

La Figure 3 montre une piste humaine et une piste autonome pour reconfigurer des politiques. La première pourrait être initiée par un utilisateur final ou un administrateur. Un utilisateur final n'est pas forcément conscient qu'il serait en train de reconfigurer des politiques. En effet, il utilise un langage de haut niveau pour préciser ses objectifs



HSSI : Interface Système de Sécurité / Homme. Exemple : P3P (`w3.org/P3P`).
SPML : Langage de Management de Politiques de Sécurité. Exemple : XACML [4].
SPLS : Spécification Logique de Politiques de Sécurité. Exemple : ASL [52].

Figure 3: Système autonome de politiques de sécurité

de sécurité. Nous utilisons le nom générique “Interface Système de Sécurité / Homme” pour ce langage haut-niveau, ou encore “HSSI (Human / Security System Interface)”. Quant aux administrateurs, ils utilisent un langage de management de politiques, qui serait à la fois de haut niveau et interprétable par la machine. Le nom générique que nous utilisons pour un tel langage est “Langage de Management de Politiques de Sécurité”, ou encore “SPML (Security Policy Management Language)”. Dans notre proposition pour l’administration autonome de contrôle d’accès, nous détaillons l’utilisation de la norme OASIS XACML (Extensible Access Control Markup Language) [4] comme un SPML.

La piste autonome du système autonome de politiques de sécurité commence par une opération d’analyse lancée par un nœud d’autorité pour négocier la modification de certaines parties des politiques avec d’autres nœuds d’autorité. Les parties à négocier seraient reformulées dans un langage logique facilitant l’analyse des règles de sécurité. L’utilisation d’un tel langage est nécessaire pour représenter et analyser la logique des règles. En revanche, on ne peut pas l’utiliser comme SPML, car cette dernière doit être, d’une part, d’un niveau plus haut et, d’une autre part, plus facile à traduire en code Java. Le nom générique que nous utilisons pour le langage logique employé dans les opérations autonomes est “Spécification Logique de Politiques de Sécurité”, ou encore “SPLS (Security Policy Logic-based Specification)”. Après la négociation et la modification des parties SPLS des politiques, la piste autonome de reconfiguration est lancée par les nœuds d’autorité. Il s’agit de reformater ces parties en SPML et de les réintégrer dans les politiques SPML d’origine. Ensuite, d’une part, un interpréteur SPML applique les politiques modifiées. D’une autre part, un traducteur SPML-à-HSSI met à jour la configuration du système de

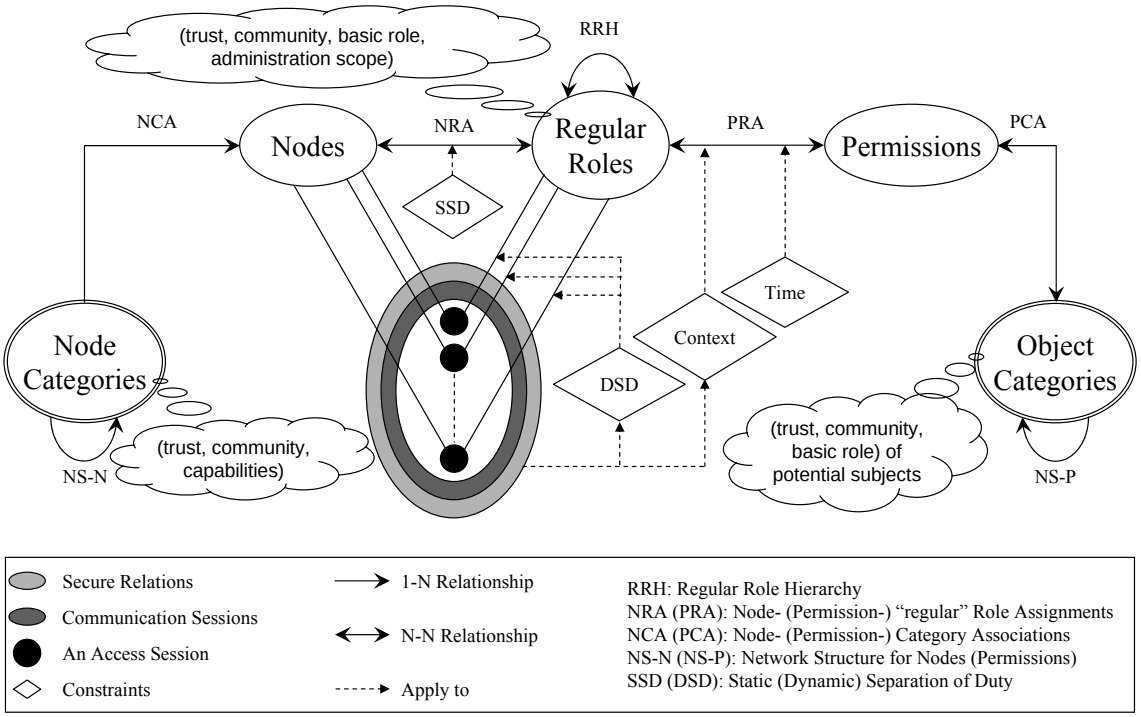


Figure 4: Modèle SRBAC : Version adaptée du modèle RBAC

sécurité au niveau de l'utilisateur final.

c.3) Modèle de contrôle d'accès

En effet, les contributions essentielles de cette thèse sont précisément dans le domaine de contrôle d'accès. Nous définissons un modèle de contrôle d'accès pour les IOrg-AutoNets. Notre modèle est une variante du modèle basé sur les rôles RBAC (Role-Based Access Control) [89]. Il est le résultat d'une adaptation de RBAC aux aspects organisationnels et autonomes des IOrg-AutoNets. Notre modèle est basé sur des relations sécurisées déjà établies entre les nœuds d'un IOrg-AutoNet, et il s'appelle SRBAC (Secure Relation Based Access Control). La Figure 4 montre les différents composants du modèle SRBAC.

Nous avons déjà présenté nos premiers essais pour définir le modèle SRBAC dans une publication introductive [12]. Le modèle SRBAC satisfait les exigences suivantes de contrôle d'accès dans un IOrg-AutoNet :

1. Les privilèges d'accès se déterminent selon des rôles affectés aux nœuds.

2. Il est possible de définir des relations hiérarchiques entre les rôles des nœuds pour organiser l'affectation des privilèges d'accès.

3. La structure organisationnelle du réseau est prise en compte.

4. Le modèle est contextuel vis-à-vis de trois attributs de nœuds, qui sont l'autorité, la disponibilité et la fiabilité.
5. Il est possible d'appliquer des politiques de sécurité d'une façon décentralisée.
6. Le modèle permet de définir des politiques distribuées d'administration.
7. Le modèle permet à son système d'administration d'être collaboratif.
8. Le modèle même peut être utilisé pour spécifier ses politiques d'administration.
9. Le modèle permet à son système d'administration d'être autonome.
10. Une correspondance entre les rôles administratifs et certains rôles réguliers.

Le modèle de contrôle d'usage UCON (Usage CONtrol) [105] pourrait être un modèle de base intéressant pour SRBAC. On pourrait définir les rôles des nœuds et les catégories des sujets et des objets en utilisant des attributs contextuels et mutables. D'une autre part, l'application et la gestion des politiques ne sont pas forcément centralisées. En plus, la possibilité de spécifier des obligations et des conditions, et l'aspect de continuité de décision, rendent UCON un modèle générique, extensible, flexible et dynamique, ce qui est exigé dans un système autonome. Néanmoins, les relations entre les attributs ne sont pas prises en compte, ce qui rend difficile la définition des hiérarchies de rôles. D'une autre part, il n'a pas un modèle administratif associé, contrairement au modèle RBAC.

Le modèle orienté organisation Or-BAC (Organization-Based Access Control) [5] est basé sur les rôles et permet d'en définir des hiérarchies. Des catégorisations de sujets et d'objets sont possibles. Il permet l'utilisation des informations contextuelles. L'application des politiques peut être décentralisée, et le modèle administratif associé AdOr-BAC [32] est distribué et collaboratif. Autrement dit, il suffirait de bien configurer les composants du modèle Or-BAC, en prenant en compte la nature autonome des IOrg-AutoNets, pour définir le modèle SRBAC. Néanmoins, il serait plus simple et efficace d'utiliser pour base de SRBAC un modèle moins sophistiqué comme RBAC.

Une relation sécurisée dans un IOrg-AutoNet relie deux nœuds. Un nœud peut avoir un différent rôle de base dans chacune de ses relations sécurisées. Voir la Définition 8 des rôles de base, et la Définition 9 des relations sécurisées. Le rôle de base ayant le maximum de permissions administratives est le rôle d'autorité A. Un nœud d'autorité peut utiliser les privilèges d'administrateur quand il coopère avec d'autres nœuds d'autorité pour accomplir des opérations autonomes. Il peut aussi utiliser certains autres privilèges administratifs quand il gère l'ensemble des nœuds qui sont sous son contrôle. En revanche, il utilise les privilèges du rôle de base non-administratif NA, qui est le rôle de base sans aucune permission administrative, quand il communique tout simplement avec d'autres nœuds. Il y'aurait éventuellement d'autres rôles de bases selon l'évolution du réseau.

Définition 8 *On définit l'ensemble totalement ordonné des rôles de base $\mathcal{B}$, avec NA et A pour bornes inférieure et supérieure respectivement. On définit la fonction bAbility qui renvoie la classe de capacité représentant le minimum des conditions pour acquérir un rôle de base.*

- $bAbility : \mathcal{B} \longrightarrow \mathcal{K}$
  - $\forall b1, b2 \in \mathcal{B}, (b1 \geq b2) \Leftrightarrow (bAbility(b1) \geq bAbility(b2))$
  - $bAbility(NA) = LD$
  - $bAbility(A) = HD$

Notation 2 *On utilise dans la Définition 8, ainsi que dans le reste de ce résumé :*

- *b, avec des indices éventuels, pour représenter un rôle de base (un élément de $\mathcal{B}$).*

Définition 9 *On définit l'ensemble de toutes les relations sécurisées SR, et la fonction sRels qui renvoie le sous-ensemble des relations sécurisées d'un nœud.*

- $SR \subseteq \mathcal{N}^2 \times \mathcal{B}^2$
  - $\forall \rho \in SR, \rho = (x, y, b_x, b_y), b_x$ *étant le rôle de base de* x *dans* ρ *et* b_y *étant le rôle de base de* y *dans* ρ
  - $\forall \rho 1, \rho 2 \in SR, ((\rho 1 = (x, y, b_x, b_y)) \wedge (\rho 2 = (x, y, b'_x, b'_y))) \Rightarrow ((b_x = b'_x) \wedge ((b_y = b'_y))$
- $sRels : \mathcal{N} \longrightarrow 2^{SR}$
  - $\forall x \in \mathcal{N}, sRels(x) = \{\rho \in SR \mid (\rho = (x1, x2, b1, b2)) \wedge ((x = x1) \vee (x = x2))\}$

Notation 3 *On utilise dans la Définition 9, ainsi que dans le reste de ce résumé :*

- *ρ, avec des indices éventuels, pour représenter une relation sécurisée (un élément de SR).*

Etant basé sur le modèle RBAC, le modèle SRBAC définit d'ores et déjà les mécanismes de contrôle d'accès suivants pour un IOrg-AutoNet :

1. Un nœud acquière des permissions en fonction de son rôle régulier. Voir la Définition 10 d'un rôle régulier, dans laquelle on définit également un champ d'administration.

   Définition 10 *On définit l'ensemble des rôles réguliers RR, la fonction bRole qui renvoie le rôle de base d'un nœud dans une relation sécurisée, la fonction aComm qui renvoie le sous-ensemble de communautés géré par un nœud en fonction d'un rôle de base donné (champ d'administration), et la fonction rRole qui renvoie le rôle régulier d'un nœud dans une relation sécurisée.*

   - $RR \subseteq \mathcal{T} \times \mathcal{C} \times \mathcal{B} \times 2^{\mathcal{C}}$
   - $bRole : \mathcal{N} \times SR \longrightarrow \mathcal{B}$
   - $aComm : \mathcal{N} \times \mathcal{B} \longrightarrow 2^{\mathcal{C}}$
   - $rRole : \mathcal{N} \times SR \longrightarrow RR$

- $\forall x \in \mathcal{N}, \forall \rho \in sRels(x), \exists b \in \mathcal{B}, bRole(x, \rho) = b,$
  $rRole(x, \rho) = (tLevel(x), nComm(x), b, aComm(x, b))$

2. Une hiérarchie sert à structurer les rôles réguliers. Cette hiérarchie, qu'on appelle RRH (Regular Role Hierarchy) dans SRBAC, définit une relation de domination qui organise un héritage des privilèges entre les rôles réguliers; Un rôle régulier hérite les privilèges des rôles réguliers qu'il domine. Voir d'abord la Définition 11 qui décrit une relation de domination entre les champs d'administration, et ensuite la Définition 12 qui décrit la relation de domination caractérisant l'héritage entre les rôles réguliers.

   Définition 11 *Etant donné as_1 et as_2 deux champs d'administration représentés par deux ensembles de communautés, on écrit $as_1 \succeq as_2$ et on dit que as_1 domine as_2 si et seulement si les nœuds de n'importe quelle communauté dans as_2 appartiennent à une communauté dans as_1.*

   - $\forall as_1, as_2 \in 2^{\mathcal{C}},$
     $(as_1 \succeq as_2) \Leftrightarrow (\forall c_2 \in as_2, \exists c_1 \in as_1, c_2 \subseteq c_1)$

   Notation 4 *On utilise dans la Définition 11, ainsi que dans le reste de ce résumé :*

   - *as, avec des indices éventuels, pour représenter un champ d'administration (un élément de $2^{\mathcal{C}}$).*

   Définition 12 *Etant donné $r1$ et $r2$ deux rôles réguliers, on écrit $r1 \succeq_{RR} r2$, et on dit que $r1$ domine $r2$ dans RR, si et seulement si :*

   - $\forall r1, r2 \in (RR \cup \{(L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})\}),$
     $r1 = (t1, c1, b1, as1), r2 = (t2, c2, b2, as2),$
     $(r1 \succeq_{RR} r2) \Leftrightarrow ((t1 \geq t2) \wedge (c1 \subseteq c2) \wedge (b1 \geq b2) \wedge (as1 \succeq as2))$

   Notation 5 *On utilise dans la Définition 12, ainsi que dans le reste de ce résumé :*

   - *$(L, \mathcal{N}, NA, \emptyset)$ pour borne inférieure de RRH. C'est un rôle abstrait qui ne donne droit à aucune permission.*
   - *$(H, \emptyset, A, \mathcal{C})$ pour borne supérieure de RRH. C'est un rôle abstrait qui donne droit à toutes les permissions spécifiées dans le système de contrôle d'accès.*
   - *r, avec des indices éventuels, pour représenter un rôle régulier (un élément de RR).*

3. Un nœud peut avoir plusieurs rôles réguliers, et un rôle régulier peut être affecté à plusieurs nœuds. Autrement dit, la relation "affectation nœud-rôle", qu'on appelle NRA (Node-Role Assignment), est une relation de type plusieurs-à-plusieurs. Voir les Définitions 13, 14 et 15 qui décrivent l'ensemble NRA et les fonctions utilisées pour parcourir ses éléments.

Définition 13 *On définit la fonction nRoles qui renvoie l'ensemble de tous les rôles réguliers affectés à un nœud donné, et la fonction rNodes qui renvoie l'ensemble de tous les nœuds ayant un rôle régulier donné :*

- $nRoles : \mathcal{N} \longrightarrow 2^{RR}$,
  $\forall x \in \mathcal{N}, \forall \rho \in sRels(x), nRoles(x) = \{r \in RR \mid rRole(x, \rho) = r\}$
- $rNodes : RR \longrightarrow 2^{\mathcal{N}}$,
  $\forall r \in RR, rNodes(r) = \{x \in \mathcal{N} \mid \exists \rho \in sRels(x), rRole(x, \rho) = r\}$

Définition 14 *On définit l'ensemble des affectations directes nœuds-rôles NRA, et la fonction dRoles qui renvoie l'ensemble des rôles réguliers affectés directement à un nœud donné :*

- $NRA \subseteq \mathcal{N} \times RR$
  - $\forall x \in \mathcal{N}, \forall r \in RR, ((x, r) \in NRA) \Leftrightarrow$
    $((r \in nRoles(x)) \wedge (\forall r' \in RR, r' \neq r, r' \succeq_{RR} r, r' \notin nRoles(x)))$
- $dRoles : \mathcal{N} \longrightarrow 2^{RR}, \forall x \in \mathcal{N}, dRoles(x) = \{r \in RR \mid (x, r) \in NRA\}$

Définition 15 *On définit la fonction allnRoles qui renvoie les rôles réguliers directs d'un nœud et tous leurs rôles réguliers hérités (dominés) :*

- $allnRoles : \mathcal{N} \longrightarrow 2^{RR}, \forall x \in \mathcal{N},$
  $allnRoles(x) = \{r \in RR \mid \exists r' \in dRoles(x), r' \succeq_{RR} r\}$

4. On peut appliquer des contraintes de séparation statique de tâches (SSD: Static Separation of Duty) sur l'ensemble des affectations nœuds-rôles. Selon une contrainte de type SSD, l'affectation d'un rôle régulier à un nœud pourrait empêcher l'affectation d'autres rôles réguliers à ce même nœud.

5. Une permission peut être affectée à plusieurs rôles réguliers, et un rôle régulier peut avoir plusieurs permissions. Autrement dit, la relation "affectation permission-rôle", qu'on appelle PRA (Permission-Role Assignment), est une relation de type plusieurs-à-plusieurs. Voir les Définitions 16 et 17 qui décrivent l'ensemble PRA et les fonctions utilisées pour parcourir ses éléments.

   Définition 16 *On définit l'ensemble des catégories d'objets OC, et la fonction rObjects qui renvoie l'ensemble des catégories d'objets associées à un rôle régulier par correspondance :*

   - $OC \subseteq (\mathcal{T} \cup \{nil_{\mathcal{T}}\}) \times (\mathcal{C} \cup \mathcal{N}) \times (\mathcal{B} \cup \{nil_{\mathcal{B}}\})$
   - $rObjects : RR \longrightarrow 2^{OC}$
     - $\forall r \in RR, (r = (t, c, NA, \emptyset)) \Leftrightarrow (rObjects(r) = \{(t, c, NA)\})$
     - $\forall r \in RR, ((r = (t, c, b, as)) \wedge (b \neq NA))$
       $\Leftrightarrow (rObjects(r) = \{(t, c', b) \in OC \mid c' \in as\})$

Remarque 6 *Dans la Définition 16:*

- $nil_{\mathcal{B}}$ *est utilisé à la place du rôle de base pour l'ignorer si nécessaire.*
- *L'ensemble des catégories d'objets OC est défini pour créer une hiérarchie qui distribue les permissions sur des ensembles d'objets selon les attributs des nœuds demandeurs d'accès potentiels. On prend en considération les niveaux de confiance tout d'abord, ensuite les communautés puis leurs sous-communautés à toutes les profondeurs, et finalement les rôles de base.*

Définition 17 *On définit l'ensemble des permissions $\mathcal{P}$, la fonction pPool qui renvoie le sous-ensemble de $\mathcal{P}$ associées à une catégorie d'objets, l'ensemble PRA des affectations directes permissions-rôles, la fonction $rPerms$ qui renvoie l'ensemble des permissions affectées à un rôle régulier, et la fonction pRoles qui renvoie l'ensemble des rôles réguliers ayant une permission donnée :*

- $pPool : OC \longrightarrow 2^{\mathcal{P}}$
- $PRA \subseteq RR \times \mathcal{P}$
  - $\forall r \in RR, \forall p \in \mathcal{P},$
    $((r,p) \in PRA) \Leftrightarrow (\exists oc \in rObjects(r), p \in pPool(oc))$
- $rPerms : RR \longrightarrow 2^{\mathcal{P}}, \forall r \in RR,$
  $rPerms(r) = \{p \in \mathcal{P} \mid \exists r' \in RR, ((r',p) \in PRA) \wedge (r \succeq_{RR} r')\}$
- $pRoles : \mathcal{P} \rightarrow 2^{RR}, \forall p \in \mathcal{P},$
  $pRoles(p) = \{r \in RR \mid \exists r' \in RR, ((r',p) \in PRA) \wedge (r \succeq_{RR} r')\}$

6. On peut appliquer des contraintes dynamiques sur les affectations permissions-rôles. Une contrainte dynamique est souvent contextuelle basée sur la relation sécurisée qui encapsule l'opération d'accès, ou bien une contrainte de temps.

7. Une session d'accès est lancée par un seul nœud.

En outre, le modèle SRBAC définit particulièrement les mécanismes de contrôle d'accès suivants en termes d'adaptation du modèle RBAC aux aspects organisationnels et autonomes des IOrg-AutoNets :

1. L'hétérogénéité des nœuds est prise en compte : la classe de capacité définit le rôle de base (Définition 8).

2. L'évolution et les caractéristiques du contexte sont prises en compte : les attributs définissant les rôles réguliers sont contextuels, et un rôle régulier dépend de la relation sécurisée dans le cadre de laquelle se passe l'opération d'accès (Définition 10).

3. Autogestion des affectations des rôles réguliers aux nœuds : la spécification d'un rôle régulier identifie les catégories des nœuds où le système d'administration du contrôle d'accès devrait chercher des candidats pour l'acquérir.

4. Aspect organisationnel : les catégories de nœuds définissent une structure organisationnelle (NS-N: Network Structure for Nodes), facilitant davantage l'autogestion des affectations des rôles réguliers aux nœuds.

5. Auto-configuration des affectations des permissions aux rôles réguliers : la spécification d'un rôle régulier identifie un ensemble de catégories d'objets associées, et par conséquent les permissions associées à ces dernières.

6. Dérivation des politiques : les catégories d'objets définissent une structure organisationnelle (NS-P: Network Structure for Permissions), qui permet au système d'administration de contrôle d'accès de catégoriser les règles définies par les utilisateurs finaux, et de les traduire en politiques de sécurité à bas-niveau.

7. Distribution : un seul rôle régulier peut être activé par un nœud dans une session d'accès, et cela serait le rôle régulier caractérisé par la relation sécurisée qui encadre l'opération d'accès, mais un nœud pourrait toujours activer plusieurs rôles réguliers dans des sessions d'accès parallèles selon plusieurs relations sécurisées.

8. On peut appliquer des contraintes de séparation dynamique de tâches (DSD: Dynamic Separation of Duty) sur l'activation des rôles réguliers. Selon une contrainte de type DSD, l'activation d'un rôle régulier par un nœud pourrait empêcher l'activation d'autres rôles réguliers par le même nœud dans des sessions d'accès parallèles.

La définition formelle du modèle SRBAC : Après avoir défini ses différents composants ci-haut, on peut décrire le modèle SRBAC formellement et en totalité par la définition suivante (Définition 18) :

Définition 18 *Le modèle SRBAC de contrôle d'accès (Secure Relation Based Access Control), qui est défini pour les réseaux IOrg-AutoNets (Figure 1 et Définition 2) et illustré par la Figure 4, est composé des éléments suivants :*

- $\mathcal{N}$*: l'ensemble des nœuds du réseau (cf. Définition 2).*
- $\mathcal{S}$*: un ensemble de sessions d'accès.*
- CS*: un ensemble de sessions de communication.*
- SR*: l'ensemble des relations sécurisée (cf. Définition 9).*
- RR*: l'ensemble des rôles réguliers (cf. Définition 10).*
- RRH*: la relation d'ordre partiel définie sur* RR *qu'on appelle l'hiérarchie des rôles réguliers (cf. Définition 12).*
- NC*: l'ensemble des catégories des nœuds (cf. Définition 2).*
- $NS-N$*: une relation d'ordre partiel définie sur* NC*, et on l'appelle la structure réseau pour les nœuds (Network Structure for Nodes).*

- *OC: l'ensemble des catégories des objets (cf. Définition 16).*
- *$\mathcal{P}$: l'ensemble des permissions (cf. Définition 17).*
- *$NS-P$: une relation d'ordre partiel définie sur OC, et on l'appelle la structure réseau pour les permissions (Network Structure for Permissions).*
- *NRA: l'ensemble des affectations directes nœuds-rôles réguliers (cf. Définition 14).*
- *PRA: l'ensemble des affectations directes permissions-rôles réguliers (cf. Définition 17).*
- *NCA: un ensemble d'associations entre les nœuds et les catégories des nœuds.*
- *PCA: un ensemble d'associations entre les permissions et les catégories des objets.*
- *$node : \mathcal{S} \longrightarrow \mathcal{N}$: une fonction qui relie une session d'accès s_i au nœud qui l'a lancée $node(s_i)$.*
- *$activeRoles : \mathcal{S} \longrightarrow 2^{RR}$: une fonction qui relie une session d'accès s_i à un ensemble de rôles réguliers déjà activé, tel que $activeRoles(s_i) \subseteq allnRoles(node(s_i))$. Voir la Définition 15 pour la spécification de la fonction $allnRoles$.*
- *$canAccess : \mathcal{S} \times \mathcal{P} \longrightarrow \{true, false\}$: un prédicat spécifiant que $(canAccess(s, p) = true) \Leftrightarrow (\exists r \in activeRoles(s), p \in rPerms(r))$. Voir la Définition 17 pour la spécification de la fonction $rPerms$.*

Langage de spécification des politiques SRBAC : Nous proposons le langage XACML (eXtensible Access Control Markup Language) [4] pour décrire les politiques SRBAC. Il est un langage interprétable au niveau système, où des politiques bas-niveau peuvent être dérivées, ou bien des spécifications logiques équivalentes peuvent être extraites dans le cadre d'un processus autonome. Il peut également servir comme une interface homme-machine à utiliser par des administrateurs. Autrement dit, XACML est un bon candidat pour le langage de gestion du type SPML (Security Policy Management Language) proposé dans notre modèle de système autonome de politiques de sécurité (Figure 3). Une autre bonne raison pour choisir XACML est sa conformité avec la nature hétérogène de l'environnement d'un IOrg-AutoNet, étant une norme ouverte et basée sur le langage XML universellement utilisé.

Plus précisément, nous profitons du profil défini pour RBAC dans la version 2.0 du langage XACML [15]. Ce profil est basé sur la norme ANSI INCITS 359-2004 [2] qui décrit le modèle RBAC de référence [85] adopté par l'institut NIST [3]. En effet, nous étendons le profil RBAC de XACML v2.0 pour pouvoir exprimer des composants spécifiques à SRBAC. Comme illustré dans la Figure 5, nous définissons les nouvelles entités NCPS (Node Category Policy Set) et OCPS (Object Category Policy Set) en XACML v2.0 pour représenter les composants organisationnels de SRBAC, étant les catégories des nœuds, les catégories des objets et les structures NS-N et NS-P associées (voir la définition formelle du modèle SRBAC : Définition 18).

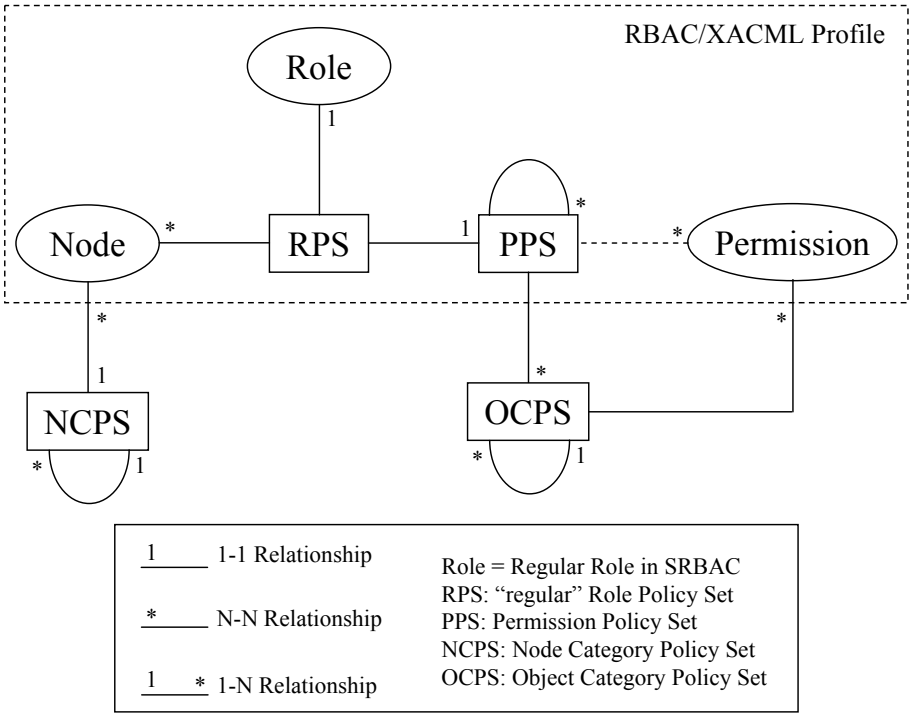


Figure 5: Le profile RBAC de XACML v2.0 étendu par des entités SRBAC

La différence principale entre le profil étendu SRBAC et le profil original RBAC se manifeste dans la spécification des permissions. C'est l'entité PPS (Permission Policy Set) qui décrit les permissions dans le profil RBAC. Quant au profil SRBAC, c'est l'entité OCPS (Object Category Policy Set) qui joue ce rôle. Néanmoins, on utilise les entités PPS pour exprimer les permissions d'administration de SRBAC. Les liaisons entre les entités PPS et les entités OCPS permettent aux politiques d'administration du système SRBAC de s'adapter aux changements du contexte de ce dernier.

c.4) Modèle d'administration

L'objectif principal de la thèse est de proposer une solution de sécurité autonome pour les réseaux de type IOrg-AutoNet. Cela serait une solution d'administration de la sécurité. Plus précisément, nous proposons une solution d'administration de contrôle d'accès. Notre contribution dans ce cadre est une modèle d'administration associé à notre modèle de contrôle d'accès SRBAC. Notre modèle d'administration s'appelle ASRBAC (Administrative SRBAC), et il est illustré dans la Figure 6. Ce dernier montre également que le modèle ASRBAC est basé sur le modèle SRBAC lui-même, ce qui lui permet d'avoir des propriétés d'un système autonome. Nous avons déjà détaillé une première version du modèle ASRBAC dans une publication introductive [13].

En effet, ASRBAC est une extension du modèle d'administration ARBAC02 [73] associé au modèle de contrôle d'accès RBAC [89] qu'on utilise comme base pour notre

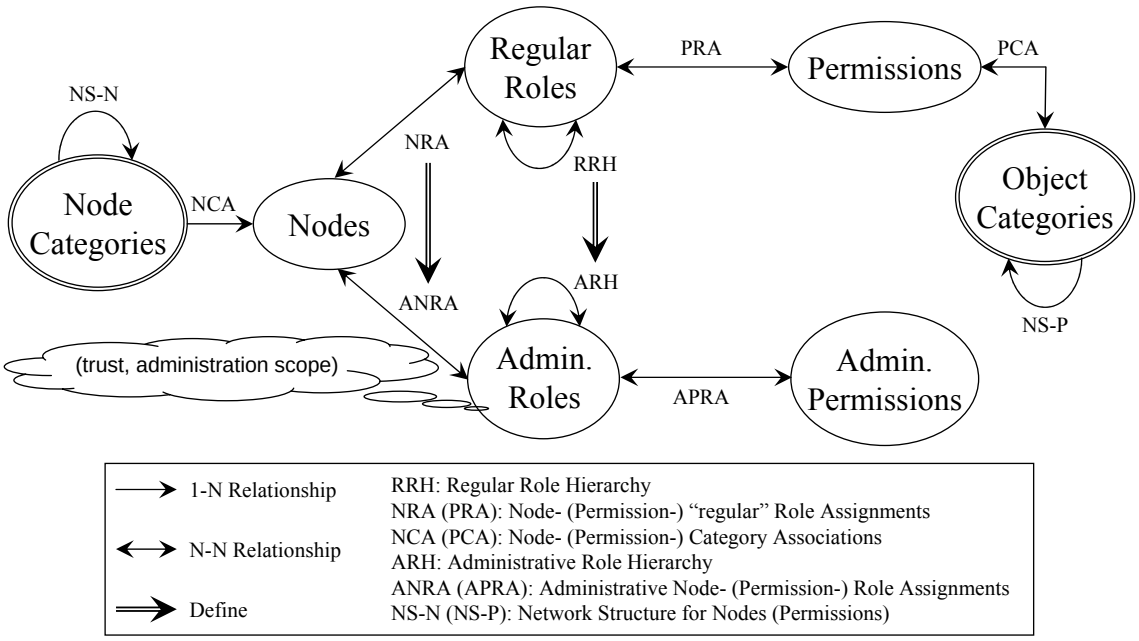


Figure 6: ASRBAC : modèle d'administration basé sur SRBAC

modèle SRBAC. En utilisant ARBAC02 comme base pour ASRBAC, on satisfait déjà certains besoins d'administration de la sécurité dans un IOrg-AutoNet ; l'administration est distribuée, et les rôles administratifs sont organisés selon un ordre partiel dans une hiérarchie qu'on appelle ARH (Administrative Role Hierarchy). Voir la Définition 19 des rôles administratifs et de leurs relations avec l'ensemble des rôles réguliers. Voir ensuite la Définition 20 de l'hiérarchie des rôles administratifs ARH.

Définition 19 *On définit l'ensemble des rôles administratifs AR et la fonction aRoles qui renvoie le sous-ensemble des rôles administratifs d'un nœud d'autorité :*

- $AR \subseteq \mathcal{T} \times 2^{\mathcal{C}}$
- $aRoles : \mathcal{N} \longrightarrow 2^{AR}$
  - $\forall x \in \mathcal{N}, aRoles(x) = \{(t, as) \mid \exists r \in allnRoles(x), \exists c \in \mathcal{C}, r = (t, c, A, as)\}$

Remarque 7 *Dans la Définition 19 :*

- *La fonction allnRoles renvoie l'ensemble de tous les rôles réguliers affectés à un nœud directement ou par héritage (cf. Définition 15).*

Définition 20 *Etant donnés les deux rôles administratifs* $ar1$ *et* $ar2$*, on dit que* $ar1$ *domine* $ar2$ *dans* AR *et on écrit* $ar1 \succeq_{AR} ar2$ *selon la définition formelle suivante :*

- $\forall ar1, ar2 \in AR, ar1 = (t1, as1), ar2 = (t2, as2),$
  $(ar1 \succeq_{AR} ar2) \Leftrightarrow ((t1 \geq t2) \wedge (as1 \succeq as2))$

Notation 6 *On utilise dans la Définition 20, ainsi que dans le reste de ce résumé :*

- *ar, avec des indices éventuels, pour représenter un rôle administratif (un élément de l'ensemble AR).*

On précise dans la liste suivante des mécanismes d'administration autonome de la sécurité spécifiques au modèle ASRBAC :

1. Auto-configuration des rôles administratifs : un rôle administratif est déduit automatiquement d'un rôle régulier correspondant identifié par le rôle de base d'autorité A (Voire Définition 19).

2. Auto-configuration des affectations nœuds-rôles administratifs : l'ensemble ANRA (Administrative Node-Rola Assignments) dans ASRBAC est le sous-ensemble de NRA (Node-Role Assignments) de SRBAC où les rôles réguliers sont caractérisés par le rôle de base d'autorité A. Voir la Définition 21 de l'ensemble des affectations nœuds-rôles administratifs.

   Définition 21 *L'ensemble des affectations nœuds-rôles administratifs ANRA (Administrative Node-Role Assignments) est défini par :*

   - $ANRA \subseteq \mathcal{N} \times AR$
     - $\forall x \in \mathcal{N}, \forall ar \in AR, ((x, ar) \in ANRA) \Leftrightarrow$
       $((ar \in aRoles(x)) \wedge (\forall ar' \in AR, ar' \neq ar, ar' \succeq_{AR} ar, ar' \notin aRoles(x)))$

3. Auto-configuration des affectations permissions-rôles administratifs : la spécification d'un rôle administratif est en effet une spécification des permissions administratives du nœud d'autorité associé. Une telle spécification indique ce qu'on appelle le domaine d'administration du rôle administratif, qui est composée de :

   (a) Une portée organisationnelle : où le nœud d'autorité pourrait modifier certaines parties des composants organisationnels NS-N et NS-P du modèle SRBAC (voir la définition formelle du modèle SRBAC : Définition 18), et employer ces modifications dans la gestion des affectations NRA (voir les Définition 13, 14 et 15) et PRA (voir les Définitions 16 et 17).

   (b) Une portée hiérarchique : où le nœud d'autorité pourrait modifier certaines parties de l'hiérarchie des rôles réguliers RRH (voir les Définitions 11 et 12), et employer ces modifications dans la gestion des affectations NRA (voir les Définition 13, 14 et 15) et PRA (voir les Définitions 16 et 17).

   La première ligne du tableau "Table 1" montre l'auto-configuration du domaine d'administration d'un rôle administratif à partir de sa spécification. La portée organisationnelle est un ensemble de branches dans les structures d'arbres NS-N et NS-P représentées par leurs racines, et la portée hiérarchique est représentée par un ensemble d'intervalles délimitant des parties de l'hiérarchie RRH.

Table 1: APRA (Affectations Permissions-Rôles Administratifs) dans ASRBAC

Rôle Administratif	Portée organisationnelle	Portée hiérarchique
$(t, as) \in AR$	$\{(t, c, nil_{\mathcal{K}}) \mid c \in as\} \cup \{(t, c, nil_{\mathcal{B}}) \mid c \in as\}$	$\{[(L, c, NA, \emptyset), (t, c, A, \{c\})[\mid c \in as\}$
AR	$\{(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}), (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}})\}$	$](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[$

Variables		Constantes	
t:	Niveau de confiance	L:	Niveau de confiance le plus bas
as:	Champ d'administration	H:	Niveau de confiance le plus haut
c:	Communauté	NA:	Rôle de base non-administratif
		A:	Rôle de base d'autorité
		$\mathcal{N}$:	L'ensemble des nœuds du réseau
		$\mathcal{C}$:	L'ensemble des communautés
		$\mathcal{T}$:	L'ensemble des niveaux de confiance
		$nil_{\mathcal{T}}$:	On néglige les niveaux de confiance
		$nil_{\mathcal{K}}$:	On néglige les classes de capacités
		$nil_{\mathcal{B}}$:	On néglige les rôles de bases
		AR:	L'ensemble des rôles administratifs

4. Collaboration : les nœuds d'autorité pourrait coopérer pour modifier n'importe quel composant SRBAC dans le cadre d'une opération administrative, si jamais l'un d'entre eux ne peut pas le faire individuellement. La deuxième ligne du tableau "Table 1" montre cette propriété de collaboration dans ASRBAC. Elle définit comme domaine d'administration pour tous les rôles administratifs en mode de coopération une portée hiérarchique étant l'hiérarchie complète des rôles réguliers RRH (voir les Définitions 11 et 12), et une portée organisationnelle composée des totalités des éléments organisationnels NS-N et NS-P du modèle SRBAC (voir la définition formelle du modèle SRBAC : Définition 18).

Les politiques ASRBAC : Les cibles des actions administratives ASRBAC sont les composants SRBAC. En effet, la spécification des politiques ASRBAC fait partie de la spécification des politiques SRBAC associées, ce qui est l'une des critères d'autogestion du modèle ASRBAC. Un exemple de spécification de politiques SRBAC et ASRBAC associées est présenté dans l'annexe A.

Les cinq prédicats ASRBAC suivants permet de réaliser les actions XACML qui seraient définies dans les spécifications des politiques ASRBAC:

1. $canAssign(ar, @nc, [r1, r2[)$: le rôle administratif ar permet de sélectionner un nœud de la catégorie nc, ou bien l'une des catégories qu'elle domines, pour lui affecter un rôle régulier de l'intervalle $[r1, r2[$.

2. $canAssignP(ar, @oc, [r1, r2[)$: le rôle administratif ar permet de sélectionner une permission associée à la catégorie d'objets oc, ou bien l'une des catégories qui la

dominent, pour l'affecter à un rôle régulier de l'intervalle $[r1, r2[$.

3. $canModify(ar, [r1, r2[)$: le rôle administratif ar permet de révoquer des nœuds et des permissions affectés à un rôle régulier de l'intervalle $[r1, r2[$, et il permet également d'ajouter ou annuler des rôles régulier ou des relations d'héritage entre rôles dans l'intervalle $[r1, r2[$.

4. $canModifyNC(ar, nc)$: le rôle administratif ar permet d'ajouter ou annuler des catégories de nœuds dans la branche de NS-N définie par la racine nc.

5. $canModifyOC(ar, oc)$: le rôle administratif ar permet de reconfigurer et redistribuer les permissions associées aux catégories d'objets dans la branche de NS-P définie par la racine oc, et il permet également d'ajouter ou annuler des catégories d'objets dans cette branche.

En utilisant les prédicats ASRBAC précédents, on définit l'ensemble des permissions administratives AP sous forme d'actions dérivées des affectations APRA (Table 1). Ces actions sont élaborées dans les deux Définitions 22 et 23 :

Définition 22 *On définit pour le rôle administratif $(t, as) \in AR$ les cinq ensembles suivants d'actions administratives :*

1. $\{canAssign((t, as), @(t, c, nil_{\mathcal{K}}), [(L, c, NA, \emptyset), (t, c, A, \{c\})[) \mid c \in as\}$
2. $\{canAssignP((t, as), @(t, c, nil_{\mathcal{B}}), [(L, c, NA, \emptyset), (t, c, A, \{c\})[) \mid c \in as\}$
3. $\{canModify((t, as), [(L, c, NA, \emptyset), (t, c, A, \{c\})[) \mid c \in as\}$
4. $\{canModifyNC((t, as), (t, c, nil_{\mathcal{K}})) \mid c \in as\}$
5. $\{canModifyOC((t, as), (t, c, nil_{\mathcal{B}})) \mid c \in as\}$

Définition 23 *On définit pour l'ensemble de tous les rôles administratifs AR les cinq actions administratives collaboratives suivantes :*

1. $canAssign(AR, @(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}),](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[)$
2. $canAssignP(AR, @(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}}),](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[)$
3. $canModify(AR,](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[)$
4. $canModifyNC(AR, (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}))$
5. $canModifyOC(AR, (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}}))$

Les nœuds d'autorité peuvent reconfigurer les affectations APRA (Table 1), les politiques ASRBAC et les actions administratives (permissions) associées (Définitions 22 et 23) sans aucune intervention humaine. En effet, la réalisation et la reconfiguration du système d'administration ASRBAC sont complètement auto-gérables, ce qui est une contribution importante de cette thèse.

La définition formelle du modèle ASRBAC : Après avoir défini ses différents composants ci-haut, on peut décrire le modèle administratif ASRBAC formellement et en totalité par la définition suivante (Définition 24) :

Définition 24 *Le modèle ASRBAC d'administration de contrôle d'accès (Administrative Secure Relation Based Access Control), qui est défini en association avec le modèle de contrôle d'accès SRBAC (Figure 4 et Définition 18) pour les réseaux IOrg-AutoNets (Figure 1 et Définition 2), et illustré par la Figure 6, est composé des éléments suivants :*

- *Tous les composants du modèle SRBAC décrits dans la Définition 18.*
- *AR: l'ensemble des rôles administratifs, où $RR \Rightarrow AR$ (cf. Définition 19).*
- *ARH: la relation d'ordre partiel définie sur AR et appelée hiérarchie des rôles administratifs (cf. Définition 20).*
- *ANRA: l'ensemble des affectations nœuds-rôles administratifs, où $NRA \Rightarrow ANRA$ (cf. Définition 21).*
- *APRA: l'ensemble des affectations permissions-rôles administratifs (cf. Table 1).*
- *AP: l'ensemble des permissions administratives (cf. Définitions 22 et 23), qui sont des actions administratives ayant pour cibles les différents composants du modèle SRBAC décrits dans la Définition 18.*

Les propriétés d'informatique autonome dans ASRBAC : L'architecture d'un système autonome basé sur le modèle ASRBAC est présentée par la Figure 7, qui montre les propriétés suivantes de l'informatique autonome (Autonomic Computing [56, 50]) :

- **Boucle de contrôle autonome (Autonomic Control Loop)** :
  - Ce mécanisme donne au modèle ASRBAC la propriété d'auto-adaptation.
  - Les variables d'environnement "*Trust (Confiance), Availability (Disponibilité) et Authority (Autorité)*", qui caractérisent le contexte des nœuds d'un réseau IOrg-AutoNet, peuvent changer. Les nœuds d'autorité détectent ces changements et utilisent leurs permissions administratives respectives pour appliquer des changements critiques sur le composant organisationnel NS-N (Network Structure for Nodes) du modèle SRBAC (Définition 18), ce qui veut dire des changements dans des catégories de nœuds, voire dans l'hiérarchie reliant les catégories de nœuds.
  - Les nœuds d'autorité utilisent leurs permissions administratives respectives pour adapter les composants suivants du modèle SRBAC (Définition 18) aux changements de NS-N:
    - Le composant organisationnel NS-P (Network Structure for Permissions), ce qui veut dire des changements dans des catégories d'objets, voire dans l'hiérarchie reliant les catégories d'objets.

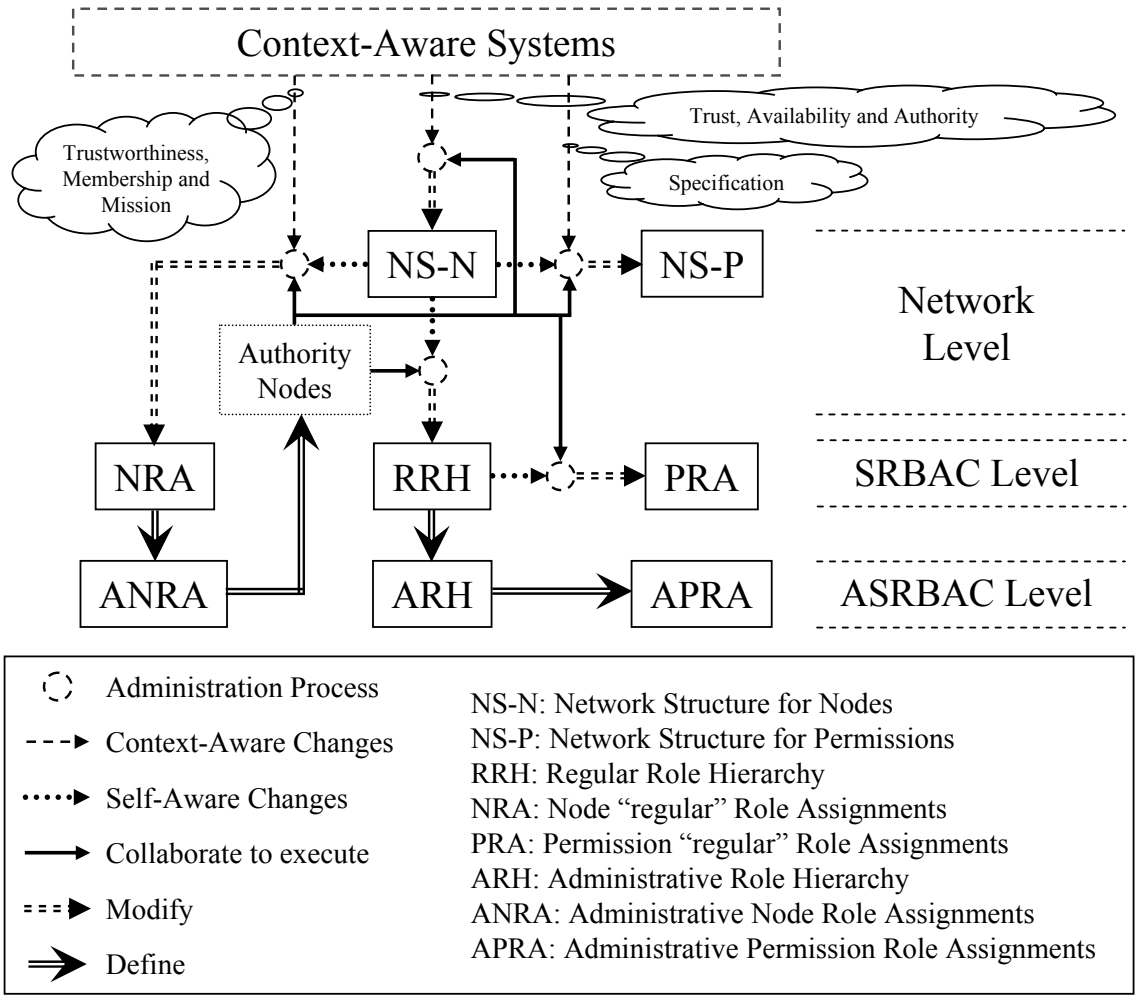


Figure 7: Les propriétés d'informatique autonome dans ASRBAC

  - Les spécifications des rôles réguliers (Définition 10) et leur hiérarchie RRH (Définition 12), ce qui mène les nœuds d'autorité à adapter également les affectations permissions-rôles réguliers PRA (Définitions 16 et 17) aux changements de RRH.
  - Les affectations nœuds-rôles réguliers NRA (Définitions 13, 14 et 15).
- Les composants du modèle ASRBAC (Définition 24) s'adaptent alors automatiquement, ce qui définit le nouvel ensemble des nœuds d'autorité, ainsi que de nouveaux rôles et permissions administratifs. On obtient alors de nouvelles politiques d'administration reconfigurées par le système lui-même (boucle de contrôle autonome) :
  - Les affectations nœuds-rôles administratifs ANRA (Définition 21) s'adaptent par définition aux changements des affectations nœuds-rôles réguliers NRA.
  - Les rôles administratifs (Définition 19) et leur hiérarchie ARH (Définition 20) s'adaptent par définition aux changements des rôles réguliers et de leur hiérarchie RRH respectivement.
  - Les nouvelles permissions administratives se reconfigurent éventuellement, étant donné que les affectations permissions-rôles administratifs APRA (Table 1) s'adaptent par définition aux changements des rôles administratifs et de leur hiérarchie ARH.

- **Autogestion prédéfinie (Predefined Self-Management)** :
  - Un système ASRBAC peut également s'adapter à des changements non-critiques selon des politiques d'administration prédéfinies. Dans ce cas là, les rôles administratifs (Définition 19), leur hiérarchie ARH (Définition 20) et éventuellement les affectations permissions-rôles administratifs APRA (Table 1) restent intacts (les politiques d'administration ne changent pas), ceci étant la différence principale entre autogestion prédéfinie et boucle de contrôle autonome.
  - Les variables d'environnement "*Trustworthiness (Fiabilité), Membership (Appartenance) et Mission*", qui caractérisent les nœuds individuellement dans un réseau IOrg-AutoNet, peuvent changer. Les nœuds d'autorité détectent ces changements et utilisent leurs permissions administratives respectives pour adapter les affectations nœuds-rôles réguliers NRA (Définitions 13, 14 et 15).
  - Les affectations nœuds-rôles administratifs ANRA (Définition 21) s'adaptent par définition aux changements des affectations nœuds-rôles réguliers NRA.
  - La variable d'environnement "*Specification (Spécification)*", qui caractérise les objectifs de sécurité du haut-niveau, peut changer. Les nœuds d'autorité détectent ces changements et utilisent leurs permissions administratives respectives pour adapter le composant organisationnel NS-P (Network Structure for Permissions) du modèle SRBAC (Définition 18), ce qui veut dire la reconfiguration et la redistribution des permissions et la modification des associations entre permissions et catégories d'objets.

- **Coopération entre nœuds d'autorité** :
  - Quand une action administrative est hors de la porté d'un nœud, le travail peut quand-même être effectué selon une distribution de tâches et une politique de collaboration entre nœuds d'autorité, ce qui est illustré par la deuxième ligne de "Table 1" qui spécifie les affectations permissions-rôles administratifs APRA, ainsi que par la Définition 23 des actions administratives collaboratives.
  - Particulièrement, les nœuds d'autorité coopèrent dans les cas suivants :
    * La validation des changements critiques à effectuer sur le composant organisationnel NS-N (Network Structure for Nodes) du modèle SRBAC (Définition 18) dans le contexte d'une boucle de contrôle autonome.
    * L'élection de nœuds pendant la gestion de nouvelles affectations nœuds-rôles réguliers NRA (Définitions 13, 14 et 15), que ce soit en réponse à des changements de NS-N dans le contexte d'une boucle de contrôle autonome ou en réponse à des changements de certaines variables d'environnement dans le contexte d'une autogestion prédéfinie.
    * La négociation de politiques de sécurité pendant le changement du composant organisationnel NS-P (Network Structure for Permissions) du modèle SRBAC (Définition 18) en réponse à des changements dans les objectifs de sécurité du haut-niveau dans le contexte d'une autogestion prédéfinie.

d) Conclusion

L'objectif de cette thèse est d'établir une base pour l'administration de la sécurité dans les réseaux autonomes. On a commencé par construire le contexte de la recherche en définissant un modèle de réseau autonome qu'on appelle IOrg-AutoNet (Infrastructureless Organizational Autonomic Network), ainsi que des composants d'une plateforme pour le système d'administration de la sécurité associé.

Pour parvenir à l'objectif, il était nécessaire de définir un modèle de sécurité propre aux réseaux IOrg-AutoNet. On a donc introduit le modèle SRBAC (Secure Relation Based Access Control). SRBAC est un modèle de contrôle d'accès basé sur les rôles qui prend en compte le contexte d'un réseau IOrg-AutoNet et sa structure variable.

La contribution principale de la thèse a été alors présentée sous forme d'un modèle d'administration associé au modèle de contrôle d'accès proposé. On a appelé ce modèle d'administration ASRBAC (Administrative SRBAC). Il est un modèle d'administration collaborative basé sur son modèle de sécurité associé SRBAC. ASRBAC définit des réactions autonomes aux changements contextuels dans un réseau IOrg-AutoNet.

L'introduction du modèle SRBAC et de son modèle d'administration ASRBAC implique plusieurs pistes de recherche future. On en précise notamment la négociation de politiques de sécurité et la réalisation de systèmes basés sur ces deux modèles. Cette thèse présente les bases d'un algorithme de négociation de politiques (Section 5.6/Chapitre 5), et une conception préliminaire d'un système basé sur SRBAC/ASRBAC (Chapitre 7).

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Dissertation

Submitted in fulfillment of the requirements for the

Ph.D. degree in Computer Science of

TELECOM ParisTech

l'École Nationale Supérieure des Télécommunications

By

Mohamad ALJNIDI

Toward a Security Administration System for Autonomic Networks

Defended on December 14, 2009

Dissertation Committee:

Isabelle Chrisment	**Reviewers**
Hatem Bettahar	
Dominique Gaïti	**Examiners**
Christophe Bidan	
Pascal Urien	
Jean Leneutre	**Thesis Advisor**

Preface

It is complex enough for humans to efficiently manage security in infrastructureless networks. Our goal is to make this security management autonomic. In this context, our PhD thesis proposes an autonomic access control system.

We provide a definition of autonomic networks, and a set of bases of autonomic security. Afterwards, we define a type of autonomic networks that we call IOrg-AutoNet (Infrastructureless Organizational Autonomic Network). The nodes of an IOrg-AutoNet are classified according to three attributes: trustworthiness, availability and capabilities. This classification allows nodes to acquire different roles, and certain roles make certain nodes able to cooperate for managing the network.

We define an access control model for IOrg-AutoNets and we call it SRBAC (Secure Relation Based Access Control). Its policies are applied during communications between any couple of nodes already bound by a secure relation assigning certain roles to them. SRBAC is an enhanced, adapted version of RBAC. We eventually propose an extension of the RBAC profile of the specification language XACML v2.0 for writing SRBAC policies.

We define for SRBAC the administrative counterpart model ASRBAC to achieve our autonomic access control system. ASRBAC is an extension of the distributed administrative model ARBAC02, which is associated to RBAC. This extension adds aspects of collaboration, context-awareness, self-awareness, adaptability and autonomic computing. ASRBAC is based on SRBAC itself, which constitutes the basis of the autonomic behavior in our solution.

An example of an SRBAC/ASRBAC system of a home network, and an enforcement model point out and validate our contributions.

Abstract

By the beginning of the twenty-first century, academia and industry decided to work more on better solutions for computer system administration, in response to the increasing complexity and heterogeneity in modern applications [51]. The need for self-managing systems arose, and the IBM's perspective of Autonomic Computing [56] has become one of the well-known solutions in this field. A considerable number of research challenges were identified by time [55], and recent specific studies focused on such challenges in the communication domain [41]. Our work concerns security in autonomic networks.

Infrastructureless networks present several complexity issues in terms of network administration. The lack of a preestablished infrastructure, the possibility of evolution of the topology and the heterogeneity of nodes are the main reasons. Besides, administrators may not be available in certain application fields of infrastructureless networking. Therefore, we opted for working in the context of infrastructureless networks, where autonomic administration is more likely to be needed. Our work aims at establishing the bases of an autonomic access control system for infrastructureless networks, as a step toward a security administration system for autonomic networks.

In a first part presenting the research context, and after discussing the theoretical background and certain interesting related work, we point out our research objectives through a certain vision of autonomic networks and certain autonomic security bases. Afterward, we introduce a definition and an organizational structure for infrastructureless autonomic networks. We call an autonomic network having this structure IOrg-AutoNet (**I**nfrastructureless **Org**anizational **Auto**nomic **Net**work). The IOrg-AutoNet structure classifies the network nodes with respect to certain attributes based on the network context. These attributes are trustworthiness, availability and heterogeneity. Such a classification helps assigning different roles to certain nodes, which allows them to collaborate to manage the network instead of humans.

The second part of the thesis describes the autonomic access control system that we propose for IOrg-AutoNets. A first chapter defines a collaborative access control model. It concerns a communicating couple of nodes aiming to share certain resources, in the context of a secure relation binding them. We call it Secure Relation Based Access Control (SRBAC). The contribution of SRBAC model is the adaptation of the well-recognized Role Based Access Control (RBAC) model [89] to the requirements of evolution and self-management in an IOrg-AutoNet. Besides, in terms of policy specification, we make another contribution by extending the OASIS RBAC profile of XACML v2.0 [15] with SRBAC-specific entities.

We define the Administrative counterpart of SRBAC (ASRBAC) in the second chapter of the access control system part. The ASRBAC model extends the distributed ARBAC02 model [73] with a support for autonomic computing. ASRBAC is expressed using SRBAC itself, which essentially provides the basis for the autonomic administration. We may summarize the contributions of ASRBAC as an extension of ARBAC02 by the following:

1. The network nodes that have administration privileges collaborate to accomplish administration tasks related to the whole network while managing their respective administration domains.

2. The network context is monitored in order to detect and analyze systematic and critical changes while the network evolves ([50] describes those two types of changes).
3. The access control system is self-aware, which allows its components to adapt to each other.
4. The access control components representing the network structure adapt to context-aware changes.
5. The roles and the role hierarchy adapt to changes in the network structure components.
6. The node-role assignments adapt to context-aware changes and/or changes in the network structure components.
7. The access control system optimizes the low-level permission specifications to adapt to changes in the high-level security rules and/or changes in the network structure components.
8. The permission-role assignments adapt to changes in the node roles.
9. The access control system employs policy negotiation mechanisms to allow nodes to collaborate to perform the previous self-management operations.
10. Roles, role hierarchy and node-role assignments in ASRBAC are self-configured, with respect to changes in roles, role hierarchy and node-role assignments in SRBAC respectively (autonomic control loop [41]).
11. Permission-role assignments in ASRBAC are self-configured, with respect to changes in ASRBAC roles.

In a last part, a chapter is dedicated to a detailed case study about the specification, enforcement and administration of access control policies based on SRBAC and ASRBAC in a Home Network. Besides, in terms of a proof of concepts, a second chapter introduces a prototype that is based on the layered Policy-Enforcement-Implementation (PEI) framework [87]. We try in this third part of the thesis to point out a maximum of our contributions. We particularly focus on the autonomic functionality of the ASRBAC administration mechanisms.

Lists of Terminology

Acronyms

IOrg-AutoNet	Infrastructureless Organizational Autonomic Network
SRBAC	Secure Relation Based Access Control
RRH	Regular Role Hierarchy
NS-N	Network Structure for Nodes
NS-P	Network Structure for Permissions
SSD	Static Separation of Duty
DSD	Dynamic Separation of Duty
SoD	Separation of Duty
RPS	Role Policy Set
PPS	Permission Policy Set
NCPS	Node Category Policy Set
OCPS	Object Category Policy Set
ASRBAC	Administrative Secure Relation Based Access Control
ARH	Administrative Role Hierarchy
SC	Similarity Computation
LD	Local Decision
SC	Similarity Computation
RE	Representative Election
HM	High-trust oriented Multicasting
LM	Low-trust oriented Multicasting

Notations

c	community (element of $\mathcal{C}$)
x, y, z	node (element of $\mathcal{N}$)
t	trust level (element of $\mathcal{T}$)
k	capability class (element of $\mathcal{K}$)
b	basic role (element of $\mathcal{B}$)
ρ	secure relation (element of SR)
as	administration scope (element of $2^{\mathcal{C}}$)
r	regular role (element of RR)
p	permission (element of $\mathcal{P}$)
nc	node category (element of NC)
oc	object category (element of OC)
s	access session (element of $\mathcal{S}$)
ar	administrative role (element of AR)

Sets

$\mathcal{N}$	Network nodes
$\mathcal{T}$	Trust levels
$\mathcal{C}$	Communities and their subdivisions
$\mathcal{K}$	Capability classes
NC	Node categories
$\mathcal{F}$	Classification functions
SR	Secure relations
$\mathcal{B}$	Basic roles
RR	Regular roles
OC	Object categories
Att_o	Access scopes associated with an object o
Att_p	Access scopes associated with a permission p
$\mathcal{P}$	Permissions
$\mathcal{S}$	Access sessions
CS	Communication sessions
NRA	Direct node-role assignments
PRA	Direct permission-role assignments
NCA	Associations between nodes and node categories
PCA	Associations between permissions and object categories
AR	Administrative roles
$ANRA$	Direct administrative node-role assignments
$APRA$	Administrative permission-role assignments
AP	Administrative permissions

Constants and Variables

L	Low-trust: the lower bound of $\mathcal{T}$
H	High-trust: the upper bound of $\mathcal{T}$
LD	Light-Duty: the lower bound of $\mathcal{K}$
HD	Heavy-Duty: the upper bound of $\mathcal{K}$
$nil_{\mathcal{T}}$	Used instead of a trust level to ignore trust classification
$nil_{\mathcal{K}}$	Used instead of a capability class to ignore capability classification
NA	Non-Administrative: the lower bound of $\mathcal{B}$
A	Authority: the upper bound of $\mathcal{B}$
DA	Delegated Authority: an example element of $\mathcal{B}$
$\alpha_{\mathcal{T},t}$	Access scope based on the trust level t
$\alpha_{\mathcal{C},c}$	Access scope based on the community c
$\alpha_{\mathcal{B},b}$	Access scope based on the basic role b
$nil_{\mathcal{B}}$	Used instead of a basic role to ignore basic role assignment

Functions and Predicates

$tLevel$	Returns the trust level of a node
$nComm$	Returns the (innermost sub-)community of a node
$cClass$	Returns the capability class of a node
$nodeInsertion(x, c)$	Insertion of node x in community c
$nodeRemoval(x)$	Removal of node x from the network
$communityIntegration(c)$	Integration of community c in the network
$communityRevocation(c)$	Revocation of community c from the network
$communityMerging(c1, c2, c)$	Merging of two communities $c1$ and $c2$ into one community c
$communitySplitting(c, c1, c2)$	Splitting of community c into two communities $c1$ and $c2$
$trustChange(x, t)$	Changing the trust level of the node x to t
$capabilityChange(x, k)$	Changing the capability class of the node x to k
$networkMerging(net)$	Merging the network with another network net
$networkSplitting()$	Splitting the network into two networks
$bAbility$	Returns the capability class corresponding to a basic role
$sRels$	Returns the secure relations of a node
$aComm$	Returns the subset of $\mathcal{C}$ under the control of a basic role
$bRole$	Returns the basic role of a node in a secure relation
$rRole$	Returns the regular role of a node in a secure relation
$nPool$	Returns the nodes of a node category
$pPool$	Returns the permissions of an object category
$nRoles$	Returns the regular roles assigned to a node
$rNodes$	Returns the nodes assigned to a regular role
$dRoles$	Returns the direct regular roles of a node
$allnRoles$	Returns the direct regular roles of a node and their junior roles
$rObjects$	Returns the object categories mapped to a regular role
$rPerms$	Returns the permissions assigned to a regular role
$pRoles$	Returns the regular roles assigned to a permission
$node$	Maps an access session to a single node
$activeRoles$	Maps an access session to a set of active regular roles
$canAccess(s, p)$	Permission p can be granted in access session s
$aRoles$	Returns the administrative roles assigned to an authority node
$canAssign(ar, @nc, [r1, r2[)$	ar can assign regular roles from $[r1, r2[$ to nodes in nc
$canAssignP(ar, @oc, [r1, r2[)$	ar can assign permissions from oc to regular roles in $[r1, r2[$
$canModify(ar, [r1, r2[)$	ar can manage regular roles and hierarchy in $[r1, r2[$
$canModifyNC(ar, nc)$	ar can manage node categories and NS-N starting by nc
$canModifyOC(ar, oc)$	ar can manage object categories and NS-P starting by oc

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation and Goals

Nowadays technologies make it quick and easy to build and deploy different types of computer networks, or to join existing ones. This ease of use will continue to improve, and more user-friendly communication technologies are likely to emerge. This will result in different kinds of complex, heterogeneous, decentralized networks, built or extended by non-expert users. Such networks need to be able to manage themselves, because human intervention for network administration is expected to be more and more undesired, expensive, inefficient, or at least time-consuming, according to the application field.

In the security context, self-managing networks should be able to protect themselves and their resources against both internal and external potential attacks. In case of a successful attack, a network should be able to heal over the damage by itself. In other words, the security system of a self-managing network must be a self-managing system. In addition to providing self-protection and self-healing properties to the network in which it is implemented, such a security system should be able to reconfigure and optimize itself.

In order to achieve self-protection, self-healing, self-configuration and self-optimization, a computing system needs rules telling when and how to react, using what tools and on which assets. In other words, a self-managing system needs policies. According to the application field, a self-managing system might need to depend on policies specified directly by end users in the form of high-level objectives. It should be able to enforce such high-level policies, and to apply self-management operations on them to adapt them to changes in its context.

Our general objective is to propose security solutions specific to autonomic networks. The need for self-management solutions had been recognized by several specialists in different relevant domains, and many corresponding initiatives were launched, as elaborated in [55] and [41]. Our work is based on the initiative of IBM [51] about Autonomic Computing. In our research, we study the realization of the Autonomic Computing vision [56], to build a platform for an autonomic security system for infrastructureless networks. We believe that such a platform would be a step **Toward a Security Administration System for Autonomic Networks**.

In this thesis, we specifically aim at establishing the bases for an autonomic access control system in infrastructureless networks. *Why access control?* Because this is a domain where we will be able to study most of the **Autonomic Computing** concepts in terms of security, particularly the ability to maintain coherent context-aware policies, which is emphasized throughout the thesis and elaborated in Chapter 5. *Why infrastructureless networks?* Because they present several complexity issues, such as the lack of a preestablished infrastructure, the possibility of evolution in the topology and the heterogeneity of nodes, which raises the need for self-management and makes of them the most likely candidates to be **Autonomic Networks** as explained in Chapter 3.

1.2 Thesis Structure

The thesis is divided into three parts. It achieves its goal at the end of the second part, while the first one puts the reader in the exact context of the presented research, and the third part tries to convince her/him of the feasibility of the proposed solutions. In the first part, a theoretical background is provided before giving our view of autonomic networks and their security, and eventually the definition of our network model. The second part first introduces an access control model for autonomic networks, and eventually proposes the bases of an autonomic system for access control administration. A detailed case study focusing on home networks is provided in the third part pointing out the main concepts of our work and comparing it with certain competing works, as well as the specifications and design of a prototype serving as a proof of concepts.

Context: The goal of the first part of the thesis is to draw the context in which we propose our solutions. At the end of the first part, we will have proposed a variable organizational structure for infrastructureless networks that allows them to be autonomic, after presenting the elements of a corresponding platform for an autonomic security system. Those bases will be eventually introduced after presenting some related work and our views of autonomic networks and autonomic security.

Solution: The autonomic network model and the autonomic security platform, which are introduced in the first part, constitute bases for many research challenges. In order to emphasize the important aspect of policy management in autonomic computing systems, in addition to several other interesting concepts, we opted for working on an access control system for autonomic networks. In the second part, we detail our solution, which is an access control model and its administrative model, proposed as foundations of an autonomic security system for infrastructureless networks.

Feasibility: The access control solution that we propose in the second part is decentralized, context-aware, dynamic and collaborative. Several existing solutions have those features as well. However, our solution is more dynamic and flexible, and above all, it has the advantage of being self-aware, self-managing and self-adaptable altogether. These numerous characteristics need to be clarified and validated in terms of realization.

In the third part, we present a case study about an autonomic home network, and then we introduce a prototype in the form of enforcement architecture and implementation considerations for the realization of the policy model defined in the second part.

1.3 Summary of Contributions

As the title of the thesis, "**Toward a Security Administration System for Autonomic Networks**", indicates, we intend to make contributions in two fields, namely "Security Administration" and "Autonomic Networks". Because the security administration solution that we will propose is dedicated to autonomic networks, it turns to be a contribution in the field of "Autonomic Security". We present a set of concepts and definitions in this area, but we mainly try to contribute to the domain of "Access Control". It was however important to begin the title of our thesis with "**Toward**". This is because we do not claim that we propose a complete solution for an autonomic security system. More specifically, this thesis proposes a groundwork for such a system, which we believe to be indispensable in future networks.

Autonomic Networks: A first minor but basic contribution is a generic definition of autonomic networks (Definition 2.1). Based on this definition, we introduce a set of features defining an **Infrastructureless Organizational Autonomic Network (IOrg-AutoNet)** (cf. Definition 3.2). The IOrg-AutoNet model is mainly characterized by an evolving organizational structure and its evolution scheme (cf. Section 3.2). The IOrg-AutoNet model is our another contribution in the domain of autonomic networks. We already presented our first trials for defining autonomic networks and a corresponding evolving organizational model in early publications [8, 9].

Autonomic Security: In the context of a platform for an autonomic security system, we introduce a set of research challenges. Afterward, we present the groundworks for two contributions to such a platform. A first one is a suggestion of an intra-node security architecture (Figure 2.1). The second one, which is more related to the solution we propose in this thesis, is a set of bases for a security policy system that supports autonomic analysis and respecification of low-level security rules (Figure 2.2). Our bases for an autonomic security architecture and an autonomic security policy systems were already introduced and elaborated in early publications [10, 11].

1.3.1 Access Control Model

Actually, the essential contributions in this thesis are specifically in the field of access control. Hence, we define an access control model for IOrg-AutoNets. A complete chapter is dedicated to this model (Chapter 4). We do not claim however that we propose a completely new access control model. Our model is a variant of RBAC [89] adapted to the requirements of the organizational and autonomic computing aspects in IOrg-AutoNets (see Figure 4.5). Nevertheless, this adaptation implied a set of enhancements to the RBAC

model and to a related policy specification language. We consider those enhancements as contributions to the access control field. An exhaustive list of those contributions is given in Section 4.3. Here we give a quick list of the essential ones:

1. Use of context-aware information in role specification (Definition 4.4).
2. Integration of context-aware components representing the evolving organizational structure of the network (Definitions 4.10 and 4.11).
3. Extending the RBAC Profile of XACML V2.0 [15] to express the components that represent the organizational structure of the network in our model (Figure 4.6).

We already introduced our access control model in a publication about our security solutions for autonomic networks in general [12]. Yet, our model of access control is not our main contribution. We actually defined it in order to present our concepts of an autonomic access control administration system. We presented those concepts in the context of a definition of an administrative counterpart of our access control model.

1.3.2 Administration Model

We also define our access control administration model using an existing administrative model, which is ARBAC02 [73], as a basis. By extending ARBAC02, we already can specify organizational, distributed, access control administration policies as parts of the policies of the access control system itself. We still need to fulfill the requirements of a support for Autonomic Computing, which we consider as the main extension of ARBAC02. A complete list of the details of this extension is provided in Section 5.3. Section 5.5 and Figure 5.3 give a more elaborated description of the autonomic computing support in our solution. Here, we may summarize this support by the following:

1. Certain nodes collaborate to accomplish administration tasks related to the whole network, in parallel with performing other administration actions specific to their respective administration scopes.
2. Those nodes are able to monitor the output of context-aware systems while the network evolves in order to detect any changes concerning access control.
3. They are also able to detect changes in the access control components themselves.
4. They adapt the access control components in response to changes detected by analyzing the context-aware and/or self-aware information.
5. They optimize the low-level access control policies in response to changes in the end-user, high-level security specifications.
6. They negotiate the modifications they may perform on the access control components and policies, in order to maintain a consistent low-level security configuration.
7. The access control administration system is self-configured with respect to its own effects on access control components.

In a recent publication [13], we described in details our access control model and its administrative counterpart. However, their description in this thesis is more rigorous and much more enhanced. Particularly, we recently refined the definition of our network model, and thoroughly revised its formal representation, which allowed us to better point out the advantages and contributions of our solution.

Part I

Research Context

Chapter 2

Background and Objectives

In this chapter, we present a theoretical background in the fields related to our work, and eventually specify our objectives. The first section of the chapter discuss some related work, and existing concepts and solutions, in the fields of autonomic computing, autonomic networks, autonomic security administration, and access control models and their administrative models. The second section presents our own views and concepts, and by the end of this section, we will have eventually specified our research goals. The access control domain particularly interests us, because our specific contribution concerns the administration of access control in certain autonomic networks. Chapter 3 elaborates the specific model of autonomic networks on which we work, and Chapters 4 and 5 present our access control solution dedicated to those networks, in the form of a security model and its administrative counterpart respectively.

2.1 Background and Related Work

Our general objective in this thesis, as its title indicates, is to study the bases of security administration in autonomic networks. In other words, we want to define the basic components and concepts of an autonomic computing system that enforces and manages security in autonomic networks. We will specifically discuss autonomic networks and autonomic security later in this section. Let us first see what an autonomic computing system is.

2.1.1 Autonomic Computing

In 2001, IBM launched the *Autonomic Computing* initiative [51], which is the basis of our work. Since then, many studies elaborated the IBM initiative and the related concepts [56, 20, 55, 82, 40, 16, 39, 83]. According to the IBM vision, we can figure out eight characteristics of an autonomic computing system:

1. It knows itself, and its components possess a system identity as well.
2. It configures and reconfigures itself under varying and unpredictable conditions.
3. It always looks for ways to optimize its workings.

4. It is able to recover from routine and extraordinary events that might cause some of its parts to malfunction.

5. It is an expert in self-protection.

6. It knows its environment and the context surrounding its activity, and acts accordingly.

7. It functions in a heterogeneous world and implements open standards.

8. It anticipates the optimized resources needed while keeping its complexity hidden from users.

Certain other related works point out the required functionality of the autonomous agents of the future autonomic systems [75, 42, 78, 63, 53, 81]. Particularly, the authors of [81] present many advantages and types of autonomous agent functionality in a multi-agent context. Their Collaborative Object Notification Framework for Insider Defense using Autonomous Network Transactions (CONFIDANT) framework aims at fail-safe and trusted detection of unauthorized modifications to executable, data, and configuration files. CONFIDANT proposes interesting models of agent interaction between four types of autonomous agents. *Sensor agents* report file digest data, *beacon agents* verify file integrity based on the information reported by the sensor agents, and *watchdog behavior agents* dispatch *probe agents* to implement alarm signaling.

According to an interesting vision of agent-based autonomic computing in the field of distributed applications [97], crowds of microscopic robots will cooperate in the context of decentralized federations composed of autonomic agents, which will give new meanings to the terms "distributed systems" and "peer-to-peer". This study is based on the following vision: Computers will evolve from flexible and universal machines, but complicated and not reliable enough, to specialized and inflexible computing devices, but simple and reliable. Those specialized devices will be the autonomic agents of the future. Decentralization is an important aspect of autonomic computing in a communication context.

A relatively recent study [50], which is based on the initiative of IBM described above, gives more conceptual details and clarifies basic concepts, through a mapping between an autonomic computing system and the human autonomic nervous system. This study of the Autonomic Computing Paradigm elaborates the main features adopted by our solution, which will be detailed in Chapters 4 and 5. In brief, the Autonomic Computing Paradigm presented in [50] is characterized by the following:

1. Autonomic computing uses alternative programming paradigms and management techniques based on strategies used by biological systems to deal with complexity, dynamism, heterogeneity and uncertainty.

2. The environment and the organism always exist in a state of stable equilibrium and any activity of the organism is triggered to maintain this equilibrium.

3. Sensing, analyzing, planning, wisdom and execute are the keywords used to identify an autonomic system.

4. Properties of an autonomic computing system are: Self-Aware, Self-Protecting, Self-Optimizing, Self-Healing, Self-Configuring, Contextually Aware, Open and Anticipatory.

5. The environment represents the factors that can impact the desired performance of a computing system: either internal characterizing the runtime state, or external characterizing the state of the execution environment.

6. Local control loop: adding self-managing capabilities to conventional components. This loop handles known environment states.

7. The global control loop, which is triggered when one of the essential environment variables exceeds its limits, manages the behavior of the overall application, and defines the knowledge driving the local control loops. This loop can handle unknown environment states.

8. The global loop uses alternate behavior patterns from a pool of patterns (online predictive models).

9. The controller unit manages, in an integrated manner, performance, fault, security and configuration of computing systems and their applications. The integrated approach is an important feature of autonomic systems.

10. The local control loop is responsible of one autonomic component, while the global control loop is responsible of an entire application.

11. At runtime, components might be added or deleted. Application workflow dynamically changes its structure at runtime. Autonomic components should have the capability of coupling and interacting.

In a vision more specific to the communication field, authors of [41] define autonomic techniques by: deploying technology to manage and optimize the functioning of other technology on an ongoing basis. They claim that autonomic computing is seen as a way of reducing the total cost of ownership of complex IT systems by allowing reconfiguration and optimization to proceed on an ongoing basis driven by feedback on the system's ongoing behavior. Authors explain that while autonomic communication is more oriented towards distributed systems and services and to the management of network resources at both the infrastructure and the user levels, autonomic computing is more directly oriented towards application software and management of computing resources.

2.1.2 Autonomic Networks

The field of autonomic communications acquires a growing interest since the midst of the current decade [41, 96, 90, 45, 38, 80, 102, 46]. According to [41], which we use so often in our work as a comprehensive Survey of Autonomic Communications, the mathematical, economic, and technical bases of networking must be changed radically to address

the implied challenges in this field. Specifically, the next-generation network must be distributed and decentralized, self-describing, self-organizing, self-managing, self-configuring, and self-optimizing, providing a seamless communications infrastructure composed of multiple technologies and able to leverage local information and decisions without sacrificing global performance, robustness, and trustworthiness. Authors claim that the ultimate vision of autonomic communication research is that of a networked world in which networks and associated devices and services will be able to work in a totally unsupervised manner. Authors distinguish the following as important challenges for future networks:

1. Next-generation networks, which are likely to be infrastructureless, wireless, mobile and/or ad-hoc, will need to manage the trust and privacy of users and services end-to-end, without any a priori knowledge of the parties involved.

2. Information reflection and collection.

3. Lack of centralized goals and control.

4. Meaningful adaptation.

5. Cooperative behavior in the face of competition.

6. Heterogeneous services and semantics.

The ANA Project (Autonomic Network Architecture) [1] is a very interesting and promising project in the field of autonomic communications. It aims at designing and developing a novel autonomic network architecture that enables flexible, dynamic, and fully autonomous formation of network nodes as well as whole networks. The resulting autonomic network architecture will allow dynamic adaptation and re-organization of the network according to user needs. This is expected to be especially challenging in a mobile context. The ANA Project identifies fundamental autonomic network principles. The network can extend both horizontally (more functionality) as well as vertically (different ways of integrating abundant functionality).

We specifically work on networks that do not have preexisting infrastructures. Chapter 3 describes in details our network model. Many existing studies emphasize the need for self-management in infrastructureless networks [43, 67, 54, 92, 25]. For instance, the authors of [43] discuss certain self-management requirements in a wireless mobile ad hoc network which is spontaneously created for a business meeting. The main reason is the lack of an administration infrastructure. The configuration services should consider continuous modifications in addressing, naming and security management. In another instance, the authors of [67] consider that self-organizing ad hoc wireless networks can overcome the limitations of conventional networks encountered in certain applications. Those are limitations in energy consumption, implementation cost and availability of expert network administrators. The concerned applications could be for example industrial monitoring and control, intelligent agriculture or home automation.

2.1.3 Autonomic Security

It is often difficult for an end-user to acquire and use a security system [95]. Instead of working on the user-friendliness of the existing security technologies, it would rather be better to rebuild them taking into consideration the user-friendliness since the very start. One of the methods proposed by the authors of [95] is building applications with implicit security, where a user transparently performs security tasks when he or she uses the application. Deriving and adapting security configurations with respect to high-level enduser objectives and context changes is a characteristic of an autonomic security system. Such mechanisms are motivated in many existing research efforts [101, 79, 23, 62].

More dedicated studies discuss solutions of autonomic security administration [60, 108, 14, 70, 57]. For instance, the article [60] proposes a solution for introducing reconfigurability in the security architecture. The authors define an autonomic security loop depending on a security context provider, a decision making component and adaptable security mechanisms. In another instance, the authors of [108] introduce a collaborative approach for the autonomic building of security protocols. Nodes negotiate security protocols according to their requirements, and build them on the fly with the help of other nodes if necessary. The authors claim that such protocols are self-configurable and self-healing.

In the specific context of networking and communication, several dynamic security approaches are proposed in the literature for networks that need to be autonomic, especially infrastructureless ones [6, 65, 67, 24, 91]. For example, according to the authors of [65], there is no difference between a client and a server in terms of certification. The service is installed in every node, and a certain number of nodes, above a given threshold, can collaborate to work together as a virtual certification server.

Other studies of security solutions are more specific to the world of autonomic communications [41, 27, 58, 59, 81]. Particularly, the comprehensive survey of autonomic communications presented in [41] discusses security requirements and challenges in such environments. The authors state that while in a static scenario digital identity management does not present much of a problem, it emerges as an important issue when autonomic nodes dynamically join different alliances. In another perspective, authors claim that trust negotiation is particularly suitable for autonomic communication systems. They finally point out the necessity of designing a new generation of self-adaptive security solutions based on multi-agent systems and intelligent agent technology. The authors motivate the use of biological models of resilience, which provide an analogy with nervous and immune systems in biological organisms.

2.1.4 Access Control Solutions

The main objective of this thesis is the definition of an autonomic access control administration model for infrastructureless organizational autonomic networks. It is eventually introduced, and thoroughly elaborated in Chapter 5. In other words, we opted for working on the bases of an autonomic access control system as a step **Toward a Security Administration System for Autonomic Networks**. Therefore, we present in this section certain related work about access control models and their administrative counterparts in

general, and emphasize those which are relevant to the world of autonomic communications, before proposing a groundwork of a security policy system for autonomic networks later in this chapter, in order to introduce our vision of access control administration in such environments.

Access Control Models. A set of traditional access control models, called lattice-based, are used as authorization platforms in many legacy systems. They perform well and proved their conformity with the traditional requirements of security systems. However, they also presented several shortcomings, especially in terms of administration. An interesting survey of this family of models is provided by [88].

The Role-Based Access Control (RBAC) model [89] appeared as a complete solution in 1996 as a generic policy-neutral flexible model. Eventually, it has become the most well-recognized solution or basis of a solution for access control, especially in organizational applications. Several studies appeared later to prove its advantages with respect to other solutions [74]. Other research efforts proved its ability to be a basis for open distributed environments [17]. Several studies emphasized its flexibility as a basis for models dedicated to applications constrained by time and/or context considerations [22, 19]. Other research efforts made use of its powerful notion of user abstraction using roles, and extended it to support organizational requirements [5, 69]. The importance of RBAC motivated standardization efforts [85], and it became the ANSI INCITS 359-2004 NIST standard [3].

A very promising model was recently introduced in cooperation with the inventors of RBAC. It is the Usage CONtrol (UCON) model, that was completed and formalized on 2005 [105]. It is a model for access control, but also for a wide range of techniques for controlling resource usage, such as the Digital Rights Management (DRM) and the Trust Management (TM) approaches [76]. It even extends all existing approaches by providing sensitive information protection on the client side, and privacy protection on both the server side and the client side. Studies are being done currently to improve and continue this model, especially in terms of administration. Recent studies already proved its advantages in building a Usage-Based Security Framework for Collaborative Computing Systems [104].

Access Control Administration Models. Administration issues should be considered when studying access control models. Usually, in addition to an access control model, an administrative counterpart is also defined. In this context, many administrative models were proposed as counterparts for the RBAC model. The most known are proposed by the inventors of RBAC. They first defined the ARBAC97 model [84] as a variant of RBAC itself. A slight modification was done to resolve certain problems in permission-role assignments, and the ARBAC99 was defined later [86]. Finally, the very interesting model ARBAC02 [73] was defined taking organizational aspects into account, which allowed it to solve nearly all the problems of the previous models and its predecessors [71, 72]. The very important aspects of the ARBAC family is the distributed administration and the use of RBAC itself as a basis, which are essential requirements in our solution as will be explained in Chapters 4 and 5. A very recent interesting administration model,

which can be considered as an extension of the ARBAC family, is the AROBAC07 model [107]. It is proposed in a multi-organizational context, and supports a dynamic scalable functionality suitable for autonomic communications. Chapters 4 and 5 include more detailed comparative discussions about the previous models and other related work.

Other interesting administrative models were proposed as counterparts for RBAC or RBAC-based models [29, 30, 47, 37, 32, 31]. Particularly, the authors of [29] propose the administrative scope model that chooses a topological approach to solve the problem of indirect illegal role-role assignment of old ARBAC models [84, 86]. Nevertheless, in ARBAC02 [73], the integrity rules, which impose the inclusion of a user/permission pool of an administrative role in the user/permission pools of the higher administrative roles, would resolve the problem and guarantee authorized permission flows.

In terms of motivations for autonomic administration, several research efforts emphasize the concerns that imply dynamic management of security policies [26, 101, 18, 59]. For instance, according to the authors of [26], there are limitations in the classic approaches of security policy management: The static configuration approach gives security policies which become vulnerable by time, the online dynamic negotiation approach requires a continuous connectivity, the centralized configuration approach does not support end-to-end agreement and the trusted third party approach does not support adaptation to changes and it is not scalable. The authors propose a dynamic approach which uses for each interoperation an Interoperability Contract Document (ICD) as an end-to-end agreement on security policies. In another instance, the authors of [101] shows that by integrating situation-aware techniques in security policy enforcement, a system can evaluate the dynamic trust relations between its parts. The article defines situation-awareness by the ability of a system to adapt its behavior to situation changes.

Access Control Policies. Policy specification techniques present a critical issue in autonomic systems. Actually, such systems raise several challenges in this context. An autonomic system should be able to derive low-level instructions from high-level policy specifications on one hand, and, on the other hand, it should allow experts to use other machine-interpretable policy specification techniques to be able to manage the system in exceptional situations. Moreover, it may need to use another internal representation of policies to accomplish autonomic administration operations. The authors of [34] elaborate the different approaches of policy specification. Other existing works propose different relevant techniques with respect to certain access control solutions [48, 35, 49, 33, 52].

Policy negotiation should be a major concern in future autonomic security systems. Those systems will attain their objectives depending on autonomous agents which are supposed to negotiate any adaptation of the system policies. Many existing work already emphasizes the need for policy negotiation in certain applications [75, 66]. The authors of [75], for example, propose the use of a mobile agent to execute group security policy negotiations in IPSec-based communications, which allows for 1-to-N negotiations. A host launches a mobile agent on a group using a security policy server. The mobile agent travels from a group to another, passes by a dedicated system in each group, and gathers security policy negotiation information to take them back to the launching host, always through its security policy server.

2.2 Our Vision and Objectives

In this section we will eventually specify our objectives by presenting our vision of autonomic security in autonomic networks, with a focus on security policy management as an introduction to our solution of an autonomic access control system presented later in the second part of the thesis. Hence, we will present our basic ideas about sketches of an autonomic security architecture and an autonomic security policy system.

2.2.1 Definition of Autonomic Network

For the purposes of our research, we propose a specific definition of autonomic networks. Roughly speaking, an autonomic network is a network that is managed by its own nodes. We consider that this may be achieved through a set of incorporated autonomic systems, of which the network nodes are the autonomous agents. On behalf of each autonomic system, the nodes monitor the network environment. They are able to detect ordinary changes that imply predefined reactions. They are also able to detect critical changes that imply changes to certain elements in the network, and to the elements of the autonomic system itself. The nodes, as autonomic system components, are able to analyze the detected information and negotiate the decisions to be taken.

Definition 2.1 *An Autonomic Network is a network that incorporates a set of autonomic systems, of which the network nodes are autonomous agents that collaborate to reconfigure, protect, heal and optimize the network, in addition to the autonomic systems themselves.*

2.2.2 Vision of Autonomic Security

Here we present our own vision of an autonomic security system. It functions like the nervous system of the human body. For instance, the blink reflex is a neurological reflex towards an offending object (dust, bright light, iron sparks ...etc). It works as follows: First, visualizing the object by the eye's retinal layer. Second, sending the signal through specific nerves. And third, a signal of motor action is sent to the eye. A similar functionality in an autonomic security system could be to protect an object from malicious alterations using an adaptable context-aware security policy.

The autonomic security system tries to handle the results of a successful attack. This can be compared to the function of the immunity system in the human body. The purpose here is to mitigate the damage on one hand, and to treat the damaged object on the other hand. For instance, a fever happens when an offending organism tells the brain thermostat to raise the baseline body temperature. The white blood cells are the army by which the body attacks any strange structure that does not have the body's specific signature. A similar functionality in an autonomic security system could be to broadcast an alert signaling the loss of a node in a network for example, in order to reduce the possibility of damage if ever it would be controlled by an adversary, and to banish the lost node to certainly prevent any future communication with it.

The reactions of the nervous and immunity systems may not be satisfactory sometimes, but this will make them more efficient later. For instance, an adult is basically more able

to avoid harm than a kid. Acquired immunity happens when the body gets attacked and starts a war as explained herebefore. When the battle is won, the white cells who fought the war will remember the attacking organism. Similarly, the autonomic security system is supposed to perform any needed enhancements to its policies after healing over damage.

A human may need in some cases to learn explicitly how to react in certain cases, but this knowledge becomes later a part of the functionality of its nervous system. For instance, we learn during a driving course how to react in certain situations to avoid accidents, but later we do this autonomously. This is called learned reflexes. On the other hand, a human body may not be able in some cases to heal by itself because the sickness is stronger than its immunity system. The solution is obviously to ask the suitable medical staff for the needed treatment. Such as a human may need instructors and medical staff sometimes, which would basically be for minor periods in his/her life, we should expect a minimum of human expert intervention in an autonomic security system.

2.2.3 Sketch of an Autonomic Security Architecture

For the purposes of our research, which specifically concerns autonomic security in infrastructureless networks, we believe that a security architecture designed on basis of autonomic computing must be embedded in a network node. We propose here an Autonomic Security Architecture, which is designed to be compatible with the heterogeneity of nodes, transparent to the end-user and irrespective of the underlying networking technologies. Its main components, as illustrated in figure 2.1, are the following:

1. *Security Agents:* a set of software agents providing security services, such as data confidentiality and integrity, and self-management support, such as a security policy negotiation module.

2. *Security Management Kit:* a set of management modules that can be used either by a human administrator to perform ordinary planned management tasks, or in an autonomic context to perform self-management tasks.

3. *Autonomic Security Manager:* the autonomic security engine, which is responsible of self-management tasks on the node level, and on the network level in certain nodes having special roles. It is also responsible of the secure data exchange between applications of communicating nodes. Moreover, it manages the security database of the node, which provides its device capability parameters, such as storage capacity, and saves security materials, such as key information. In certain special nodes, this database stores also network security management data, such as network evolution management policies.

4. *Autonomic Security Layer:* an application-support layer encapsulating the previous three components.

5. *Security User Interface:* a set of user-friendly configuration and specification languages, to be used by security expert administrators, and to a certain extent, by non-expert end-users in the form of high-level tools.

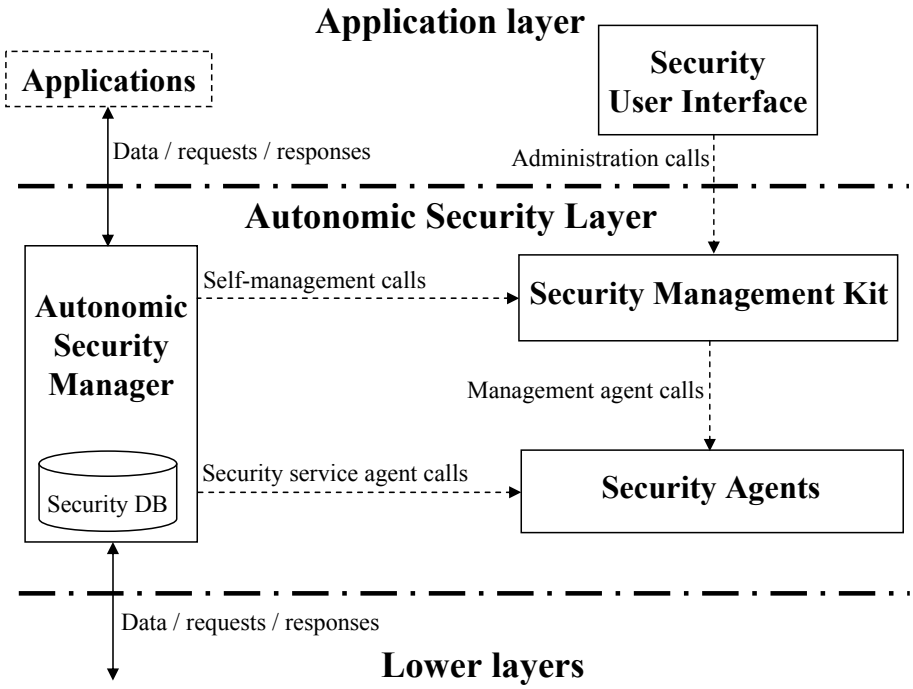


Figure 2.1: Groundwork for an Intra-Node Autonomic Security Architecture

Actually, the intra-node security architecture that we describe in this section needs more study. We only suggest some brief details about what it could incorporate. We just aim at pointing out quickly this part of the solution. In a future work, this architecture would be more elaborated and enhanced. An interesting aspect to consider will be the security of the architecture itself. Another important point, is to develop this architecture using existing standards, such as TLS (Transport Layer Security)[1].

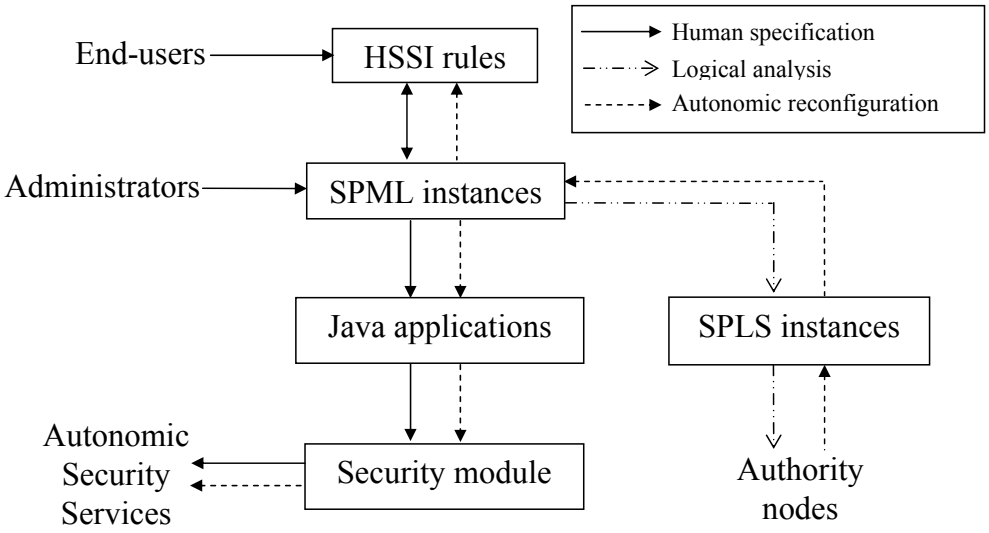
2.2.4 Sketch of a Security Policy System

A policy system is a main component of an autonomic system. It translates high-level objectives, usually through intermediary rule specifications, into low-level elementary rules, upon which autonomic decision are based. We introduce in this section our design of a security policy system for autonomic networks, which is illustrated by Figure 2.2.

An autonomic system necessarily has initial policies, which might be provided by default. Default policies would be already enforced at low level, but also they will have corresponding specifications at the administrator level and the enduser level. This is to provide a current view of the autonomic system functionality to the both types of users. Usually, default policies are modified later by the encapsulating autonomic system.

As illustrated in Figure 2.2, We identify three sources of modification for a security policy in an autonomic network: endusers, administrators and authority nodes. Endusers

[1] `http://tools.ietf.org/html/rfc5246`



HSSI: Human/Security System Interface. Example: P3P (`w3.org/P3P`).
SPML: Security Policy Management Language. Example: XACML [4].
SPLS: Security Policy Logic-based Specification. Example: ASL [52].

Figure 2.2: Groundwork for an Autonomic Security Policy System

may want to make changes to their high-level security objectives. Administrators may need to intervene in policy management in exceptional cases. Authority nodes may need to analyze, negotiate and recompose a security policy, or a certain part of it, in the context of an autonomic operation. The latter functionality is the one we expect to take place so often. In an autonomic system, human intervention is expected to be a minimum. Just as the autonomic systems of the human body, where the owner of the body (enduser) is not supposed to intervene in their functionality, while doctors (administrators) may be called in some cases. Moreover, autonomic systems learn from their experience after an administrator intervention. More details about the correspondence between autonomic computing systems and the human body systems were already provided in Section 2.2.2.

Refining goals into implementation specification could be useful for refinement of high-level policies into low-level enforceable rules [36]. Figure 2.2 illustrates human and autonomic tracks for policy reconfiguration. The human track may be launched by an enduser or an administrator. The enduser is not necessarily aware of the fact that security policies are being changed. All he knows is that he is acting on his security objectives. He does so by means of a high-level language, through a user-friendly interface, which is used for the configuration of the security environment as a whole. We use a generic name for the high-level language of the enduser interface, which is HSSI (Human/Security System Interface). The Platform for Privacy Preferences (P3P) Project[2] may provide such a language.

A common component of commercial tools is a graphical user interface which typically

[2] `http://www.w3.org/P3P/`

allows the administrator to visually select a network device or other managed element from a hierarchically arranged tree-view of policy targets, and specify the policies in the form of "`if <condition> then <action>`" rules for the selected targets [34]. Administrators are completely aware of what they are changing. This is why they use a machine-readable yet user-friendly language for managing policies. The generic name we use for this language is SPML (Security Policy Management Language). It should be suitable for the collaborative distributed environment of autonomic networks, where the operating systems of the different nodes may not be the same. We think that a language based on the Extensible Markup Language (XML)[3] fulfills SPML requirements. Actually, in our contribution, we use the XML-based standard specification language XACML (Extensible Access Control Markup Language) of OASIS [4]. More details are provided in Chapter 4.

In a human specification track, policies are modified through HSSI or SPML. A built-in interpreter transforms then the SPML instances into Java applications, which will be compiled into security agents. Those latter are distributed on the nodes where the low-level policies should be enforced. An HSSI-to-SPML translator is needed when the human specification is driven by an enduser. On the other hand, the system state should be always available to the enduser in the form of high-level configurations. So an SPML-to-HSSI translator is needed in a human specification driven by an administrator.

The autonomic analysis track is launched by an authority node to negotiate and reconfigure policies with other authority nodes in the context of an autonomic operation. An authority node can access SPML instances, search for specific information needed in the negotiation process, and reformat the extracted data in a logic language specific to autonomic mechanisms. The use of such a language is necessary for representing and analyzing the logic of the policy rules. On the other hand, SPML can not be replaced by a logic language, because SPML is supposed to be more user-friendly and easier to interpret into Java code. We use the generic name SPLS (Security Policy Logic-based Specification) for the logic language. For instance, ASL (Authorization Specification Language), which is a stratified first-order logic language [52], can be used as SPLS.

After negotiating and reconfiguring the SPLS-formatted policy parts, an autonomic reconfiguration track is launched by the involved authority nodes. It aims at reformatting those logic-based policy instructions in SPML and reintegrating them in the original SPML instances to apply the related policy modifications. On one hand, the built-in SPML interpreter is called to enforce the modified policies. On the other hand, the SPML-to-HSSI translator is called to update the state of the security system configuration at the enduser high level.

Actually, we do not handle every detail of the previous security policy system in this thesis. The overall high-level design illustrated by Figure 2.2 is provided here as an introduction to our main contribution of autonomic access control administration (cf. Chapters 4 and 5). In the context of the autonomic reconfiguration track of the above security policy system, for the specific purposes of our main work, we will elaborate the use of XACML [4] as SPML to specify access control policies in Chapter 4, and only the negotiation mechanism of such policies in Chapter 5.

[3] `http://www.w3.org/TR/xml/`

Chapter 3

Infrastructureless Organizational Autonomic Networks

In this chapter, we define a network model that we call the Infrastructureless Organizational Autonomic Network (IOrg-AutoNet) model. This is a particular type of autonomic networks for which we propose specific security administration solutions in the following chapters of the thesis. In this chapter, we describe the variable structure and the evolution scheme of IOrg-AutoNets.

Section 3.1 motivates and describes IOrg-AutoNets. It positions an IOrg-AutoNet with respect to autonomic networks in general. It models the variable structure of an IOrg-AutoNet. Because this structure is based on certain variable node attributes, a subsection is dedicated to describing each of them. The section ends by a formal definition of the IOrg-AutoNet model.

Section 3.2 describes the evolution scheme of an IOrg-AutoNet. Different kinds of evolution events are presented at the node, community and network levels. The section provides formal descriptions of the different transactions of the network with respect to different kinds of evolution events. An example of a node life cycle is elaborated focusing on the changes in the administrative roles that a node can acquire while the network evolves.

Section 3.3 concludes the chapter by a summary of the basic ideas, concepts and elements of the IOrg-AutoNet model. It also briefs some related future work.

3.1 IOrg-AutoNet Model

In infrastructureless networks, certain components may need continuous adaptation to context-aware changes. Such networks may evolve in terms of topology, population and/or high-level configuration. It is nearly impossible for human administrators to manage such networks efficiently. We think that such networks need to be autonomic (cf. Definition 2.1). We also think that by defining a dynamic organizational structure for the network, certain nodes can act as autonomous agents collaborating to manage the network according to their respective administration scopes.

Definition 3.1 *We call **I**nfrastructureless **Org**anizational **Auto**nomic **Net**work (IOrg-AutoNet) an autonomic network (cf. Definition 2.1) that has the following characteristics:*

1. *The network does not have a pre-established infrastructure.*
2. *The network may be configured by non-expert users using high-level specifications.*
3. *Nodes have evolving trustworthiness, with an initial trust-based classification.*
4. *Nodes can be categorized according to their availability for each other, and this categorization may evolve.*
5. *Nodes may have different capabilities in terms of computing and/or data storage.*
6. *The network has an evolving organizational structure based on the previous three node attributes.*

An IOrg-AutoNet can be a network of a home, a SOHO (Small Office/Home Office), a spontaneous business meeting, a rescue operation or a military mission. In all these application fields, a network infrastructure is not likely to be available, the network should be able to manage itself and the nodes need to cooperate in an organizational context.

For instance, in a home network, we may have nodes of different capabilities with respect to security administration. A resident's laptop may be able to perform asymmetric cryptography and to store public-key certificates, while her/his digital audio player may not. So we can expect that the laptop participates in security administration, but not the digital audio player. Besides, we can consider that the set of home devices belonging to a resident, and eventually his visitors or friends, are more available for each other with respect to the other devices in the network. So the network nodes related to a resident can be assigned to one category. Moreover, the network is likely to evolve in terms of population and/or the trust and capability attributes of its nodes. Furthermore, home network users are usually non-expert in terms of network and/or security administration.

In contrast with the IOrg-AutoNet model, certain autonomic networks may be deployed using a preexisting infrastructure, such as an autonomic enterprise intranet, and others may be infrastructureless but they do not need a node categorization in an organizational structure, such as a network of independent autonomic sensors. We opted for infrastructureless organizational environments because we think that most of next-generation networks will be dynamic and collaborative.

Figure 3.1 illustrates the IOrg-AutoNet model. It shows that the IOrg-AutoNet structure is characterized by the trust levels (trustworthiness-based classification), the communities and sub-communities (availability-based classification), and the capability classification of the network nodes.

3.1.1 Trust Levels

The nodes of an infrastructureless network are not supposed to trust each other in general. However, initial trust considerations may allow non-expert users to simply classify the network nodes as trusted or not. Such a high-level configuration should eventually classify

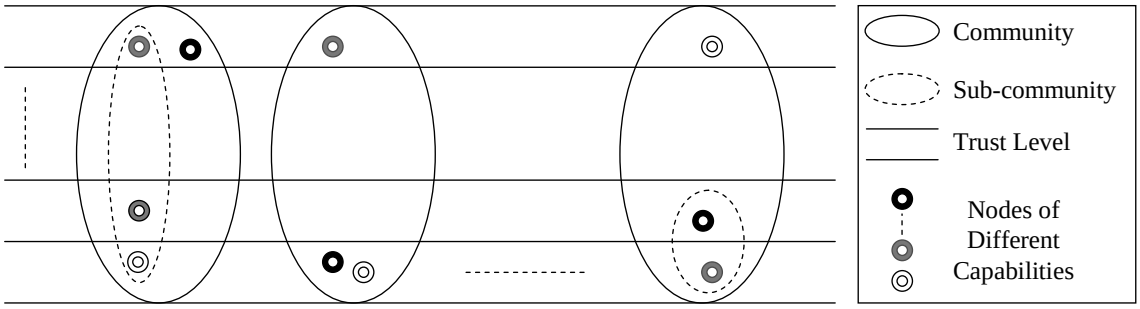


Figure 3.1: Infrastructureless Organizational Autonomic Network (IOrg-AutoNet) Model

a node in one of the two initial trust levels: Low (L) and High (H). For instance, the device of a friend of a family is not initially trusted when it integrates their home network, which assigns it to the trust level L. Later on, such a node may gain the trust of other nodes and may be autonomously assigned to some trust level between L and H.

An IOrg-AutoNet node may gain or lose a certain amount of its trustworthiness while it uses the network. This kind of dynamic measure of trustworthiness is usually based on the evolving reliability and reputation attributes of the network nodes [77, 68, 44]. We do not propose a solution for evaluating the trustworthiness of nodes. We assume that a reputation system is already implemented in the network to manage node trustworthiness and its evolution (cf. Assumption 3.1). Ideally, the trustworthiness of a node should be evaluated differently by each of the other nodes. However, we assume that the reputation system assigns each node to a trust level that has a global scope in the network (cf. Assumption 3.2).

Assumption 3.1 *We assume that an IOrg-AutoNet has a reputation system that manages node trustworthiness.*

Assumption 3.2 *We assume that the trust level of a node is the same for all the other nodes.*

3.1.2 Community Membership

An IOrg-AutoNet is divided into communities with respect to certain availability constraints. According to certain metrics related to the application field, the nodes of a community are considered relatively available for each other with regard to the other nodes of the network. For example, the set of devices of the same user may form a community in a home network. Further availability-based categorization may apply to the nodes of a community. An example is subnetworking, whereby each community may have a subdivision belonging to a subnet. For instance, a subnet can be created in a home network to be used during a holiday time in another residence. Similarly, a sub-community may be subdivided for the same or other availability reasons, and so on and so forth.

Any node in the network belongs to a community, or possibly to a subdivision of a community at a certain depth. The communities of the network do not intersect, nor do

sub-communities. Actually, when we talk about the community of a node, we mean the smallest subset of nodes it belongs to. This means that if a node should belong to another subset of nodes, this latter should be an encapsulating community or sub-community. In the rest of the thesis, we use the term "**community**" to designate either a community or a sub-community at any depth.

3.1.3 Capability Classes

According to certain capabilities in terms of computing and data storage, a node may be able to carry out certain administrative operations. Actually, for each community in an IOrg-AutoNet, there should be a node having enough capabilities to play the role of administrator. We call such a node the authority node of the community. While the use of the authority role is mandatory, other optional administrative roles requiring less capabilities may be used in an IOrg-AutoNet for purposes of administration distribution. A high-level configuration may help designating the required number of authority nodes if they have enough capabilities. Other types of administrative roles are utilized autonomously if needed, and if certain nodes of certain capabilities are still left without assignment of administration tasks.

Because the authority role requires certain advanced capabilities, which might not be found in any node in some communities, many communities may share one authority node. However, in order to have a maximum support for flexibility and decentralization, whenever possible, each community should have its authority node among its own nodes in the ideal situation. As for other types of administrative roles, they are simply not used when there are not nodes having the required capabilities, because such roles are optional. In Chapter 5, we propose an autonomic administration solution that seeks to realize an optimal node-role assignment.

According to an initial high-level configuration of node capabilities, the nodes can be simply categorized as Light-Duty (LD) or Heavy-Duty (HD). The autonomic access control system of the IOrg-AutoNet will then designate certain HD nodes as the authority nodes of the different communities (cf. Chapters 4 and 5). Later on, it may autonomously decide to assign other optional administrative roles to other nodes having capability classes between LD and HD. The authority nodes, supported by other administrative nodes of less responsibilities if any, are then supposed to collaborate to manage the network.

3.1.4 Node Categories

We define a dynamic node categorization based on the IOrg-AutoNet organizational structure. All the network nodes belong to a root node category. Then, a node category represents each one of the used trust levels. For each used trust level, a node category represents each one of the current communities. And finally, for each used trust level and current community, a node category represents each one of the used capability classes.

Node categories allow the autonomic systems of the IOrg-AutoNet to manage the network evolution in an efficient and scalable manner, because they are based on the actual classifications of nodes. Particularly, node categories allow the access control administra-

tion system to have interesting autonomic computing properties as will be explained in Chapter 5.

3.1.5 Network Model Definition

Here is a formal definition of the components of the IOrg-AutoNet model, which is illustrated in Figure 3.1, eventually including a definition of node categories:

Definition 3.2 *The IOrg-AutoNet model is defined by a tuple* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$*, with:*

- $\mathcal{N}$ *is a set of network nodes.*
- $\mathcal{T}$ *is a totally-ordered set of trust levels, having* L *and* H *as lower and upper bounds respectively.*
- $\mathcal{C}$ *is a set of network communities satisfying the following conditions:*
  - $\mathcal{C} \subseteq 2^{\mathcal{N}} \setminus \{\emptyset\}$
  - $\bigcup_{c \in \mathcal{C}} c = \mathcal{N}$
  - $\forall c1, c2 \in \mathcal{C}, (c1 \cap c2 \neq \emptyset) \Leftrightarrow ((c1 \subseteq c2) \vee (c2 \subseteq c1))$
- $\mathcal{K}$ *is a totally-ordered set of capability classes, having* LD *and* HD *as lower and upper bounds respectively.*
- $\mathcal{F}$ *is the following set of functions:*
  - $tLevel : \mathcal{N} \longrightarrow \mathcal{T}$ *returns the trust level of a node.*
  - $nComm : \mathcal{N} \longrightarrow \mathcal{C}$*, returns the community of a node.*
    - $\forall x \in \mathcal{N}, nComm(x) = \bigcap_{\{c \in \mathcal{C} | x \in c\}} c$
  - $cClass : \mathcal{N} \longrightarrow \mathcal{K}$ *returns the capability class of a node.*
- NC *is a set of node categories satisfying the following conditions:*
  - $NC \subseteq (\mathcal{T} \cup \{nil_{\mathcal{T}}\}) \times (\mathcal{C} \cup \{\mathcal{N}\}) \times (\mathcal{K} \cup \{nil_{\mathcal{K}}\})$
  - $((t, c, k) \in NC) \Leftrightarrow (((t, c, k) = (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}})) \vee$
    $((c = \mathcal{N}) \wedge (k = nil_{\mathcal{K}}) \wedge (\exists x \in \mathcal{N}, tLevel(x) = t)) \vee$
    $((k = nil_{\mathcal{K}}) \wedge (\exists x \in \mathcal{N}, tLevel(x) = t, nComm(x) = c)) \vee$
    $(\exists x \in \mathcal{N}, tLevel(x) = t, nComm(x) = c, cClass(x) = k))$

Notation 3.1 *Starting by Definition 3.2, and for the rest of the thesis:*

- *We use* c*, with eventual superscript or subscript indices to denote a community (an element of* $\mathcal{C}$*).*
- *We use* x*,* y *and* z*, with eventual superscript or subscript indices to denote a node (an element of* $\mathcal{N}$*).*

- *We use t, with eventual superscript or subscript indices to denote a trust level (an element of $\mathcal{T}$).*
- *We use k, with eventual superscript or subscript indices to denote a capability class (an element of $\mathcal{K}$).*

Remark 3.1 *In Definition 3.2:*

- *$\mathcal{N}$, $\mathcal{C}$, $\mathcal{F}$ and NC are variable sets that adapt to the network evolution.*
- *$\mathcal{T}$ and $\mathcal{K}$ are predefined sets in a given IOrg-AutoNet.*
- *The set of communities $\mathcal{C}$ may include the element $\mathcal{N}$ in certain cases, whereby the network is composed of one community which is the network itself, and it may eventually have sub-communities. In other words, in certain scenarios, a number of nodes should belong to the same community according to their availability attributes, but they cannot belong to a subdivision of the network. For example, because any community is subject to revocation (cf. Section 3.2), in certain application fields where certain nodes should always stay in the network as long as it is not totally discarded, such nodes cannot belong to a subdivision of the network.*
- *$\forall x \in \mathcal{N}, nComm(x)$ returns the deepest (smallest) subset of network nodes to which x belongs.*
- *The constants $nil_{\mathcal{T}}$ and $nil_{\mathcal{K}}$ are used in the set of node categories NC to classify a node regardless of its trust level and capability class respectively.*
- *The set of node categories NC is defined to distribute the network nodes on trust levels first, then on communities, then on sub-communities at all depths, and finally on capability classes.*

3.2 Evolution Scheme

The evolution of the network may modify the cardinality of the set of nodes, and/or their availability attributes. Such modifications may imply changes in the set of communities and sub-communities. Besides, the trustworthiness attributes of certain nodes may change by time, which may eventually modify the associations between nodes and trust levels. Moreover, there could be a need at some moment for a more fine-grained capability classification of nodes in order to use more types of administrative roles. Consequently, the IOrg-AutoNet structure may evolve in response to evolution events in the network.

We assume that a reputation system (cf. Assumption 3.1) and a resource management system are implemented in an IOrg-AutoNet to decide the changes in trust levels and communities respectively while the network evolves. The set of network nodes that have administrative responsibilities are supposed to enforce the decisions of those two systems by changing the network structure. The authority nodes of the network communities are particularly responsible of such administration tasks, especially in the security context. Chapter 5 describes in details such functionality in the context of access control.

Assumption 3.3 *We assume that an IOrg-AutoNet has a resource management system that manages node availability.*

3.2.1 Initial Structure

We assume that an initial structure is already created in an IOrg-AutoNet (cf. Assumption 3.4). We do not handle the issues of the network deployment and initialization phase. However, we may propose the following steps for the deployment phase of a home network based on the IOrg-AutoNet model. A complete study of this phase in IOrg-AutoNets in general is left to a future work. Nonetheless, Chapter 6 presents a detailed case study of the implementation of our autonomic access control solution, which is proposed in Chapters 4 and 5, in a home network. The initial IOrg-AutoNet structure of a home network may be transparently created by non-expert end-users as follows:

- The autonomic security architecture is installed in the different nodes before inserting them in the network.
- The initial trust levels and communities are derived from the high-level configuration of the network structure. In this high-level configuration, an end-user only has to classify each device as trusted or not (L or H; cf. Section 3.1.1), and to define a community for the set of devices of each user.
- The initial low-level security policies are derived from the high-level specification of the security objectives. In this high-level specification, an end-user just has to enter his security rules in the way he understands and expresses them, using the Human-Security System Interface (HSSI) of the Security Policy System (SPS; cf. Section 2.2.4). Chapter 6 gives a detailed example of such high-level specifications and their translation into low-level security policies.
- The nodes can classify each other in terms of capabilities, which helps designating the first authority nodes (see Chapter 6 for a detailed example).

Assumption 3.4 *We assume that the initial structure of the network is already created.*

3.2.2 Node-Level Transitions

We mean by a transition at the node level a change in the cardinality of the set of network nodes $\mathcal{N}$. It is also a change in the cardinality of a certain community, or eventually a set of nested communities, as long as each node belongs to a community. An IOrg-AutoNet may evolve at the node level as a result of one of the following possible events:

1. **Node Insertion:** triggered when a new node joins the network.
2. **Node Removal:** triggered when a node leaves the network by its choice.
3. **Node Banishment:** triggered when a node should not continue to use the network, and it does not know or accept that it has to leave. Node banishment is supposed

to be a decision taken collaboratively by the authority nodes in the context of self-healing. For example, a node may be banished if it is not available for a period greater than an allowed maximum, which might mean in certain scenarios that it is lost, and in case it is assigned to a certain administrative role, some other node should be elected to replace it.

4. **Node Reinsertion:** triggered when a banished node is authorized to rejoin the network. Node banishment and reinsertion will be handled in details in a future work concerning self-healing in IOrg-AutoNets.

Definition 3.3 *An IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ *may evolve at the node level and become* $(\mathcal{N}', \mathcal{T}, \mathcal{C}', \mathcal{K}, \mathcal{F}', NC)$ *according to the following transitions:*

- *nodeInsertion*(x, c)*: Insertion or reinsertion of the node* x *in a community* $c \in \mathcal{C}$*.*
  - $\mathcal{N}' = \mathcal{N} \cup \{x\}$
  - $\mathcal{C}' = \{c1 \cup \{x\} \mid c1 \in \mathcal{C}, c \subseteq c1\} \cup \{c2 \in \mathcal{C} \mid \neg(c \subseteq c2)\}$
  - $tLevel(x) = L$
- *nodeRemoval*(x)*: Removal or banishment of the node* x*:*
  - $\mathcal{N}' = \mathcal{N} \setminus \{x\}$
  - $\mathcal{C}' = \{c1 \setminus \{x\} \mid c1 \in \mathcal{C}, x \in c1\} \cup \{c2 \in \mathcal{C} \mid x \notin c2\}$

Remark 3.2 *In Definition 3.3:*

- *The initial trust level of a new node is the lowest trust level* L*, which can be modified later by the reputation system (cf. Assumption 3.1).*

3.2.3 Community-Level Transitions

We mean by an evolution at the community level a change in the cardinality of the set of network communities $\mathcal{C}$. In some cases, it is also a change in the cardinality of the set of network nodes $\mathcal{N}$. In all cases, it is a change in the set of node categories NC. An IOrg-AutoNet may evolve at the community level as a result of one of the following possible events:

1. **Community Integration:** triggered when a new community is created to insert new nodes, when a precreated community integrates the network, or when a community subdivision is defined.
2. **Community Revocation:** triggered when a community is deleted because all its nodes are removed or banished, when a community should leave the network, or when a community subdivision is canceled.
3. **Community Merging:** triggered when the resource management system (cf. Assumption 3.3) decides that new availability attributes of the nodes of two communities imply that they should belong to the same community.

4. **Community Splitting:** triggered when the resource management system (cf. Assumption 3.3) decides that new availability attributes of the nodes of a community imply that it should be split into two communities.

Definition 3.4 *An IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ *may evolve at the community level and become* $(\mathcal{N}', \mathcal{T}, \mathcal{C}', \mathcal{K}, \mathcal{F}, NC')$ *according to the following transitions:*

- *communityIntegration*(c)*: Integration of a community* c*:*
  - $\mathcal{N}' = \mathcal{N} \cup c$
  - $\mathcal{C}' = \mathcal{C} \cup \{c\}$
  - $NC' = (NC \cup \{(t, c, nil_{\mathcal{K}}) \mid \exists x \in c, tLevel(x) = t\})$
    $\cup \{(t, c, k) \mid \exists x \in c, tLevel(x) = t, cClass(x) = k\}$
- *communityRevocation*(c)*: Revocation of a community* c*:*
  - $\mathcal{N}' = \bigcup_{\{c_i \in \mathcal{C} \mid \neg(c_i \subseteq c)\}} c_i$
  - $\mathcal{C}' = \mathcal{C} \setminus \{c' \in \mathcal{C} \mid c' \subseteq c\}$
  - $NC' = (NC \setminus \{(t, c1, nil_{\mathcal{K}}) \mid t \in \mathcal{T}, c1 \in \mathcal{C}, c1 \subseteq c\})$
    $\setminus \{(t, c2, k) \mid t \in \mathcal{T}, c2 \in \mathcal{C}, c2 \subseteq c, k \in \mathcal{K}\}$

Remark 3.3 *In Definition 3.4:*

- *Integration of a community* c *implies the creation of a corresponding node category for each trust level already used in node categorization, and then the creation of a node category corresponding to each capability class already used in node categorization for each node category created for* c*.*
- *Revocation of a community* c *implies the deletion of each element in* NC *corresponding to* c *or any of its sub-communities, if any.*

Definition 3.5 *By using the transitions of Definition 3.4, we are able to describe community merging and splitting in an IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$*, which may make it evolve and become* $(\mathcal{N}', \mathcal{T}, \mathcal{C}', \mathcal{K}, \mathcal{F}, NC')$*, using intermediate steps of community integration and revocation as follows:*

- *communityMerging*$(c1, c2, c)$*: Merging of two communities* $c1$ *and* $c2$ *into a community* c*:*
  - $c = c1 \cup c2$
  - *communityIntegration*(c)
  - *communityRevocation*$(c1)$
  - *communityRevocation*$(c2)$

- *communitySplitting*$(c, c1, c2)$*: Splitting of a community* c *into two communities* $c1$ *and* $c2$*:*
  - $\forall c' \in \mathcal{C}, c' \subset c$, *communityRevocation*$(c')$
  - $c1 \subset c, c2 \subset c, c2 = c \setminus c1$
  - *communityIntegration*$(c1)$
  - *communityIntegration*$(c2)$
  - $\mathcal{C}' = \mathcal{C} \setminus \{c\}$
  - $NC' = (NC \setminus \{(t, c, nil_{\mathcal{K}}) \mid t \in \mathcal{T}\}) \setminus \{(t, c, k) \mid t \in \mathcal{T}, k \in \mathcal{K}\}$

Remark 3.4 *In Definition 3.5:*

- *After merging two communities* $c1$ *and* $c2$*, all the sub-communities of* $c1$ *and* $c2$*, if any, will be revoked by revoking* $c1$ *and* $c2$*. The result is a community including all the nodes of* $c1$ *and* $c2$ *without sub-communities.*
- *Before splitting a community, all its sub-communities must be revoked. The result is two distinct communities having no sub-communities.*

3.2.4 Network-Level Transitions

As already explained in Section 3.1.4, the set of node categories is based on the actual classification. This means that only the used trust levels and capability classes and the current communities are used in the definition of this set (cf. Definition 3.2). The reputation system of an IOrg-AutoNet (cf. Assumption 3.1) may decide to change the trust level of one or more nodes. It may then use a certain trust level for the first time. Similarly, the resource management system (cf. Assumption 3.3) may decide to change the evaluation of the capabilities of one or more nodes, which may imply the use of a capability class for the first time. So a change in trust levels or capability classes of certain nodes may cause the evolution of the set of node categories in certain cases. Such changes are considered as network-level evolution events.

On the other hand, because IOrg-AutoNets are infrastructureless by definition (cf. Definition 3.1), it is possible that two networks merge into one network, or a network splits into two networks. In both cases, $\mathcal{N}$, $\mathcal{C}$ and NC should be modified. In the case of merging two networks, even the sets of predefined trust levels $\mathcal{T}$ and the set of predefined capability classes $\mathcal{K}$ may change, as long as it is possible that each network has its own versions of such sets. Merging and splitting complete networks are considered as network-level evolution events.

Definition 3.6 *An IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$ *may evolve at the network level and become* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}', NC')$ *according to the following transitions:*

- *trustChange*(x, t)*: Changing the trust level of the node* x *to* t*, which may imply a first use of* t*:*

  - $NC' = ((NC \cup \{(t, \mathcal{N}, nil_{\mathcal{K}}) \mid \forall y \in \mathcal{N}, tLevel(y) \neq t\})$
    $\cup\{(t, c, nil_{\mathcal{K}}) \mid c \in \mathcal{C}, \forall y \in \mathcal{N}, tLevel(y) \neq t\})$
    $\cup\{(t, c, k) \mid c \in \mathcal{C}, \exists x \in \mathcal{N}, cClass(x) = k, \forall y \in \mathcal{N}, tLevel(y) \neq t\}$
  - $tLevel(x) = t$

- *capabilityChange*(x, k)*: Changing the capability class of the node* x *to* k*, which may imply a first use of* k*:*
  - $NC' = NC \cup \{(t, c, k) \mid \exists x \in \mathcal{N}, tLevel(x) = t, c \in \mathcal{C}, \forall y \in \mathcal{N}, cClass(y) \neq k\}$
  - $cClass(x) = k$

Definition 3.7 *We define the following transitions for merging and splitting an IOrg-AutoNet* $(\mathcal{N}, \mathcal{T}, \mathcal{C}, \mathcal{K}, \mathcal{F}, NC)$*:*

- *networkMerging*$(\mathcal{N}', \mathcal{T}', \mathcal{C}', \mathcal{K}', \mathcal{F}', NC')$*: Merging the network with another network* $(\mathcal{N}', \mathcal{T}', \mathcal{C}', \mathcal{K}', \mathcal{F}', NC')$*:*
  - *The resulting network is:* $(\mathcal{N} \cup \mathcal{N}', \mathcal{T} \cup \mathcal{T}', \mathcal{C} \cup \mathcal{C}', \mathcal{K} \cup \mathcal{K}', \mathcal{F} \cup \mathcal{F}', NC \cup NC')$
- *networkSplitting*()*: Splitting the network into two networks:*
  - *The network splits into* $(\mathcal{N}1, \mathcal{T}, \mathcal{C}1, \mathcal{K}, \mathcal{F}1, NC1)$ *and* $(\mathcal{N}2, \mathcal{T}, \mathcal{C}2, \mathcal{K}, \mathcal{F}2, NC2)$

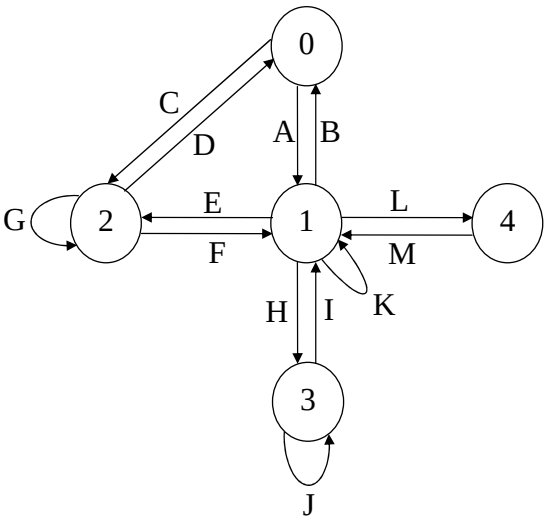
Remark 3.5 *In Definition 3.7:*

- *After a network merging transition, a series of community merging transitions may be decided by the resource management system (cf. Assumption 3.3) according to the new availability conditions.*
- *Before a network splitting transition, a series of community splitting transitions may be decided by the resource management system (cf. Assumption 3.3) according to the network splitting requirements.*

3.2.5 Life Cycle of a Node

Figure 3.2 illustrates an example of the life cycle of an IOrg-AutoNet node in terms of its role in network management, and with regard to the different evolution events mentioned above. We consider subnet exportation and reintegration in this figure to show how a third node role other than the initial ones, which is the delegated authority, can make part of a node lifetime. In terms of the evolution of the network structure, a subnet exportation is equivalent to the revocation of a set of sub-communities, and a subnet reintegration is equivalent to the integration of the same set of sub-communities as new communities, because they might have evolved during exportation time.

The life cycle is represented by an automaton, whereby a state is a role of the node at a certain moment, and a transition could be the result of a certain evolution event. For example, in the context of a community merging, a node can be non-administrative in one of the two communities and then it becomes the authority node of the new community



0: out of network
1: non-administrative
2: authority node
3: delegated authority
4: blocked node

Node insertion:	A or C.
Node removal:	B or D.
Node banishment:	L.
Node reinsertion:	M.
Community integration:	A, C, G or K.
Community revocation:	B, D, G or K.
Community merging/splitting:	E, F, G or K.
Network merging:	A, C, E, F, G or K.
Network splitting:	B, D, E, F, G or K.
Trust level changes:	G, J or K.
Capability class changes:	E, F, G, H, I, J or K.
Subnet exportation:	E, F, G, H, J or K.
Subnet reintegration:	E, F, G, I, J or K.

Figure 3.2: Administrative Life Cycle of a Node in an IOrg-AutoNet

(Transition E), or it can be the authority node of one of the two communities and then it becomes non-administrative (Transition F) or stays the authority node in the new community (Transition G).

We should note here that a change in the trust level may have effects on the scope of administrative privileges of a node, even if its basic administrative responsibilities do not change as illustrated in Figure 3.2. Such types of effects are considered by the autonomic access control system, which handles administrative rights. The integration of all node attributes in the process of granting administrative rights and other types of permissions is discussed and elaborated in Chapters 4 and 5 which present our solution for access control in IOrg-AutoNets.

3.3 Conclusion

There is a specific model of autonomic networks that we use as a basis in our work. In this chapter, we motivated and described this model. Our model is called IOrg-AutoNet (Infrastructureless Organizational Autonomic Network). As the name of the model indicates, we are interested in networks that do not have a predefined infrastructure, and we build an evolving organizational structure for them. We justified this choice and defined the model. The different concepts and elements of the organizational structure were elaborated.

The assignment of administrative roles to nodes is an essential functionality in autonomic networks. The different evolution transitions were studied, considering their effects on the variable elements of the network structure on one hand, and on node roles on the other hand. Formal definitions of structure transitions were provided in the first case, and an automaton illustrating the administrative life cycle of a node in the second case.

A lot of aspects and elements of the IOrg-AutoNet model still need to be studied. Each definition of evolution transition should be more formalized and studied in more details.

However, we focus in this thesis on access control administration, which has a basic mission of adapting the security configuration to the evolutions described in this chapter. Chapter 4 describes our access control model, and Chapter 5 describes its autonomic administrative counterpart.

Part II

Access Control Solution

Chapter 4

SRBAC: The Access Control Model

In this chapter, we define a collaborative decentralized access control model for IOrg-AutoNets, and we call it Secure Relation Based Access Control (SRBAC). We show throughout different sections of this chapter why our model is based on the secure relations that bind the nodes in an IOrg-AutoNet (see Section 4.4 for a description of secure relations). We also explain how SRBAC allows its administration system to have autonomic computing capabilities.

Section 4.1 specifies the access control requirements in an IOrg-AutoNet. It explains the need for assigning roles to nodes, defining organizational structures, integrating context information, decentralizing policy enforcement and supporting distribution, collaboration, self-management and self-adaptation in access control administration.

Section 4.2 discusses existing access control models with regard to SRBAC requirements. It first explains why traditional models are not suitable. Afterward, it addresses more advanced models fulfilling each a subset of the sought requirements. The section focuses on the Role-Based Access Control (RBAC) model [89] and its variants, but it also considers the recent Usage CONtrol (UCON) generic model [105].

Section 4.3 first justifies the choice of RBAC as a basis for SRBAC. It then presents our contribution, which consists in defining SRBAC components as an adaptation of RBAC components to the infrastructureless context with a support for autonomic computing, and enhancing the standard XACML policy specification language with SRBAC entities.

Section 4.4 defines a secure relation in an IOrg-AutoNet. It describes the different types of secure relations with respect to the attributes of the bound nodes. Therefore, it begins by defining the "Basic Role" attribute of a node. This section justifies in brief why our access control model is called "Secure Relation Based".

Section 4.5 shows how the trust level, community membership and basic role of a node define its actual role in a secure relation. This actual role is called "Regular Role". This section gives a formal specification of regular roles, and defines a regular role hierarchy that allows access control to be efficient and scalable.

Shared resources in an IOrg-AutoNet, which we call objects in the context of access

control, also have certain attributes related to SRBAC. Section 4.6 defines elementary attributes for objects, and calls them access scopes. It explains the dependencies between access scopes and secure relations. An organizational categorization of objects is then introduced using a compound attribute based on access scopes. The section explains how permissions are associated with access scopes, and consequently with object categories, in SRBAC.

SRBAC has components representing organizational structures. Section 4.7 describes those components and explains their use. Two subsections describe two network structures built on node categories and object categories respectively. Those network structures integrate the SRBAC model to support decentralization and self-management.

Section 4.8 defines the SRBAC model using the RBAC model [89] as a basis. Nodes, objects, their attributes and the organizational structures, which are described in previous sections, are essential components of SRBAC irrespectively of its basic model. This section presents their integration in the SRBAC version that is based on RBAC. Besides, it describes the other SRBAC components resulting from RBAC.

Section 4.9 presents the specification language of SRBAC policies. It explains how SRBAC policies can be specified using a standard language, which is the OASIS eXtensible Access Control Markup Language (XACML) [4]. It justifies the choice of this language with respect to SRBAC requirements. It describes enhancements of an RBAC/XACML profile aiming at fulfilling the requirements of SRBAC policy specification.

Section 4.10 concludes the chapter with a summary of the proposed access control model. The summary focuses on the contributions of SRBAC with respect to its basic model RBAC. Afterwards, the section briefly motivates the solution proposed in the next chapter for access control administration.

4.1 Access Control Requirements

IOrg-AutoNet nodes are supposed to have variable roles in the network with regard to their evolving attributes of trustworthiness, availability and authority. We need an access control model that assigns roles to nodes depending on such information related to the network context. Certain components in this model should represent the organizational structure of the IOrg-AutoNet. Besides, the sought access control model should support decentralization and collaboration between nodes. Particularly, it should support self-management and self-adaptation in access control administration. The following subsections justify those different access control requirements in IOrg-AutoNets.

4.1.1 Organizational Structure

The IOrg-AutoNet structure transforms an infrastructureless network into an autonomous organization. This means that the principals of such an organization are expected to be the network nodes themselves. Each node should have a role in this organization, and its access to the shared resources would depend on its role.

Requirement $\mathcal{R}$ 4.1 *The access control model should be based on the dependence of access privileges upon the assignment of roles to nodes.*

Furthermore, just as in any organization, all roles do not define the same privileges, and a role may define a subset of the privileges of another role. A hierarchy of node roles is therefore possible to build. By making use of the relationships between roles in such a hierarchy, which define privilege inheritance, we can achieve more efficient and scalable access control.

Requirement $\mathcal{R}$ 4.2 *the access control model should allow the use of hierarchical relationships between node roles in specification of permissions.*

As explained in Chapter 3, the IOrg-AutoNet structure evolves, which may unexpectedly change the nodes' memberships in the divisions of the corresponding organizational structure, and possibly their roles. In other words, the evolution of an IOrg-AutoNet affects its access control system. Therefore, the sought access control model should allow its administrative counterpart to react to changes in the IOrg-AutoNet structure.

Requirement $\mathcal{R}$ 4.3 *The access control model should have components that represent the organizational structure of the IOrg-AutoNet.*

4.1.2 Context-Awareness

The role of a node defines its access privileges. Such a role is basically identified by the management capabilities and responsibilities assigned to the node. These management attributes of a node may change unexpectedly due to the network evolution. Therefore, the role of a node should be based on its *up-to-date administrative characteristics*.

Besides, the availability of a node may affect its access rights. A node belonging to the same community of the resource hosting node may have certain kinds of permissions that cannot be assigned to a node belonging to a different community, even if the both access requesting nodes basically have the same authority privileges in the network. Therefore, the *up-to-date availability coordinates* of a node should be associated with its administrative attribute to identify its role.

In addition to authority and availability considerations, the trustworthiness of a node is also an access control parameter in IOrg-AutoNets, because they are basically infrastructureless networks. An access requesting node may win or lose certain access rights due to its trust credentials, which are represented in an IOrg-AutoNet by its trust level. Trust levels of nodes are also subject to changes. The *up-to-date trust level* should be associated with administrative privileges and availability coordinates to identify the role of a node.

Requirement $\mathcal{R}$ 4.4 *The access control model should be aware of the networking context and its changes in terms of authority, availability and trustworthiness of nodes.*

4.1.3 Decentralization

An infrastructureless network is not supposed to have a central authority. We try however in the IOrg-AutoNet structure to make use of a possible partial centralization through creating communities and their subdivisions (cf. Chapter 3), with respect to possible availability constraints in the application field. Even when partial centralization is possible, the decentralization is a must between the authority nodes of the network communities. Moreover, the decentralization between subdivisions should be also considered inside a community. In terms of access control, each authority of a community or a sub-community should have its own autonomous policy enforcement module.

Requirement $\mathcal{R}$ 4.5 *The access control model should allow a decentralized enforcement of its policies.*

The administration of the access control system must not be centralized either. The IOrg-AutoNet structure may evolve very frequently in certain applications, and the corresponding access control administration should be a continuous, reactive distributed process. Therefore, in an IOrg-AutoNet, where the nodes themselves manage their network, there should not be a single administrative role of the type SSO (System Security Officer).

Requirement $\mathcal{R}$ 4.6 *The access control model should allow its administrative counterpart to define distributed administration policies.*

4.1.4 Collaboration

IOrg-AutoNet nodes share the security policy either by simple replications or using some distribution algorithm. If a node is not able to host or share the security policy, it can access it on its authority node as a shared resource. An IOrg-AutoNet node is able to enforce the security policy autonomously. However, that policy is subject to changes with respect to the network evolution. Actually, the network nodes need to collaborate to accomplish such changes. They need to negotiate to reach a unified decision about the changes to apply, in order to maintain the consistency of the security policy.

Requirement $\mathcal{R}$ 4.7 *The access control model should allow its administrative counterpart to define collaborative administration.*

We can summarize the characteristics of an access control model that supports collaboration as follows:

- Generic: the nodes should be able to use and exchange different types of access control parameters, such as availability, trust and authority.
- Scalable: the nodes may need to perform big numbers of shared operations.
- Fine-grained: access control for individual subjects targeting individual objects should stay possible, even if the categorization of subjects and objects is required with respect to the organizational structure.

- Flexible: the access control policies and their modifications in the context of a collaboration should be transparent for the users, and based on high-level objectives.
- Dynamic: supports specification and modification of policies at runtime.

More details about the access control requirements for collaboration can be found in [100].

4.1.5 Self-Management

The IOrg-AutoNet nodes should be able to manage the access control system by themselves. It is required that the role of an authority node defines privileges of access control administration. In other words, the administrative model of the access control system should be based on the access control model itself.

Requirement $\mathcal{R}$ 4.8 *The access control model should be generic enough to serve as a basis of an administrative counterpart.*

As a part of a network that presents needs for autonomic computing, the access control administration system should have the following properties:

- Self-configuration: node-role and permission-role assignments should be based on mappings between the node roles and the network structure.
- Self-optimization: the optimization of the access control policies in response to the network evolution should imply the optimization of the access control administration policies as well.
- Self-protection: the access control management privileges should be defined as access control rights.
- Self-healing: the access control administration policies should be updated with respect to the reconfiguration of the access control policies in response to unexpected changes in the network or in its context.

More details about the autonomic computing properties can be found in [56].

Requirement $\mathcal{R}$ 4.9 *The access control model should allow the access control administration system to be autonomic.*

4.1.6 Self-Adaptation

The access control administration system should be able to adapt the access control material to the different types of changes in the network or in its context. This can be achieved when the access control model fulfills the previous requirements from $\mathcal{R}$ 4.1 to $\mathcal{R}$ 4.9. The entities of the access control administration system itself are also subject to changes with respect to the network evolution. Such entities are supposed to be based on the variable components of the access control model for purposes of self-management. In other words, the administration system should be able to adapt itself after adapting the components of

the access control model in response to the network evolution. Actually, self-adaptation is required in autonomic systems that should detect and survive unexpected changes, as explained in details in [50]. Our approach for self-adaptation is that an administrative role is specified in an autonomous way with respect to a corresponding role in the access control model. More details will be provided in Chapter 5.

Requirement $\mathcal{R}$ 4.10 *The access control model should support mappings between its roles and corresponding administrative roles.*

4.2 Related Work

One may think that the SRBAC model can be based on the traditional Access Control Matrix model [61], which is usually defined as a Discretionary Access Control (DAC) model. At a first glance, a DAC model, whereby access decisions are taken at the discretion of the resource owner, can be seen as suitable for infrastructureless environments because of the absence of a trusted central authority. However, there are several shortcomings in such a model with respect to SRBAC requirements. Particularly, it does not support assignment of attributes, such as roles and categories, to subjects or objects. In other words, at least, the requirements related to the organizational structure ($\mathcal{R}$ 4.1 - $\mathcal{R}$ 4.3) cannot be fulfilled.

The Lattice-Based Access Control models [88] constitute a family of traditional security models that are based on the concept of assigning security labels to subjects and objects. The use of such labels my help defining roles, categories and related structures ($\mathcal{R}$ 4.1 - $\mathcal{R}$ 4.3). However, those models are defined for controlling data flow in terms of confidentiality or integrity, while SRBAC is supposed to control resource usage. Nevertheless, an access right granted by SRBAC might result in an illegal data flow. A model for controlling data flow in IOrg-AutoNets can be a future adjacent solution used together with SRBAC.

The requirement $\mathcal{R}$ 4.1 implies the need for a model that associates nodes with roles. Therefore, the Role-Based Access Control (RBAC) model [89] can be a basis for SRBAC. Particularly, RBAC supports hierarchical relationships between roles ($\mathcal{R}$ 4.2). Some RBAC-based approaches have been proposed for multidomain, decentralized collaborative environments [93, 94] ($\mathcal{R}$ 4.3, $\mathcal{R}$ 4.5 - $\mathcal{R}$ 4.7). On the other hand, RBAC is used as a basis in certain context-aware solutions [28, 103] ($\mathcal{R}$ 4.4). However, the existing RBAC-based solutions usually propose particular enhancements to RBAC that we do not necessarily need in IOrg-AutoNets. We think that it is better to define SRBAC by adapting the original RBAC model itself, with respect to the access control requirements of IOrg-AutoNets that are not fulfilled ($\mathcal{R}$ 4.3 - $\mathcal{R}$ 4.10).

Other access control models define other types of organizational structuring of access control components. The Task-Based Access Control (TBAC) model [99] proposes structuring of access actions, which is basically required in workflow applications. However, our focus is currently on organizational structuring of subjects, roles and objects ($\mathcal{R}$ 4.2, $\mathcal{R}$ 4.3). The ongoing task-based validation of rights in a workflow-like access control process can be a future enhancement of SRBAC. The TeaM-based Access Control (TMAC) model [98] makes it possible to grant access rights at runtime, taking into account the team to

which the access requesting subject currently belongs, in addition to its role. This is actually required in dynamic organizational structures, such as the IOrg-AutoNet's, where teams of authority nodes can be temporarily created to accomplish different administration tasks. However, we currently just seek collaborations between the authority nodes that have to share administration tasks ($\mathcal{R}$ 4.7). A team-based access control feature can be a future enhancement of SRBAC.

The Usage CONtrol (UCON) model [105] could be an interesting basis for SRBAC. It is an attribute-based model, in which persistent and mutable attributes of subjects and objects, in addition to context-aware attributes, can be defined as access control parameters. Therefore, we can define node roles and categories of subjects and objects in UCON, taking into account context changes ($\mathcal{R}$ 4.1, $\mathcal{R}$ 4.3, $\mathcal{R}$ 4.4). Besides, UCON does not assume the centralization of policy enforcement or management ($\mathcal{R}$ 4.5, $\mathcal{R}$ 4.6). Moreover, other interesting concepts of UCON, such as obligations, conditions and continuity of decisions, make of it an access control model which is generic, scalable, fine-grained, flexible and dynamic ($\mathcal{R}$ 4.7). However, it is not clear how to define relationships between attributes, which presents a challenge in fulfilling the requirement $\mathcal{R}$ 4.2. Furthermore, a corresponding administrative model is not yet proposed in the literature, which means that self-management and self-adaptation features should be worked out from scratch in order to fulfill the requirements $\mathcal{R}$ 4.8, $\mathcal{R}$ 4.9 and $\mathcal{R}$ 4.10. Actually, we prefer basing SRBAC on the RBAC model. The main advantage of RBAC, with respect to UCON, is that it has an administrative model to build upon. The ongoing access control features provided by UCON can be left to a future work.

The Organization-Based Access Control (Or-BAC) model [5] is role-based, and it supports role hierarchies ($\mathcal{R}$ 4.1, $\mathcal{R}$ 4.2). It categorizes subjects and resources in an abstract manner ($\mathcal{R}$ 4.3). It defines a component dedicated to the use of context information in access control ($\mathcal{R}$ 4.4). Policy enforcement can be decentralized in Or-BAC ($\mathcal{R}$ 4.5), and its administration model AdOr-BAC [32] is distributed ($\mathcal{R}$ 4.6) and collaborative ($\mathcal{R}$ 4.7). Moreover, the fact that AdOr-BAC is expressed using Or-BAC itself ($\mathcal{R}$ 4.8), together with the use of a context entity, may establish a framework for self-management ($\mathcal{R}$ 4.9) and self-adaptation ($\mathcal{R}$ 4.10). In other words, we may just need to suitably configure the components of the Or-BAC model, and enhance them with autonomic computing support, in order to define the SRBAC model. However, a simpler enhancement of RBAC would be enough to achieve the sought access control model. We do not necessarily need all the features of Or-BAC in our work. An extension of SRBAC based on Or-BAC could be studied in a future work.

4.3 Contributions

We chose RBAC as a basis for the definition of SRBAC, as already rationalized in Section 4.2. It is role-based by definition ($\mathcal{R}$ 4.1), and defines role hierarchies ($\mathcal{R}$ 4.2). It does not clearly define organizational structures for access control elements other than roles, but such structures can easily be incorporated ($\mathcal{R}$ 4.3). For instance, organizational structures were defined for users and permissions in a previous work [73] in order to establish a

platform for the definition of the administrative model ARBAC02. No specific support of context awareness was defined in the original RBAC model, but it has abstract and flexible component specifications, which may help filling such a gap ($\mathcal{R}$ 4.4). For instance, a previous work [28] could define environment roles in RBAC for context-aware applications. Besides, RBAC is generic enough to allow configuring its components in a way that makes the access control system decentralized ($\mathcal{R}$ 4.5), distributed ($\mathcal{R}$ 4.6), collaborative ($\mathcal{R}$ 4.7) and autonomic ($\mathcal{R}$ 4.8 - $\mathcal{R}$ 4.10), as we elaborate in this chapter.

Briefly, our contribution is the adaptation of the RBAC model to the requirements of IOrg-AutoNets as infrastructureless networks on one hand, and autonomic networks on the other hand. The following list gives the main elements of this contribution, with regard to the requirements discussed in Section 4.1:

- Nodes and resources are categorized in organizational structures based on context-aware attributes ($\mathcal{R}$ 4.3, $\mathcal{R}$ 4.4).
- The role of a node is specified using the context-based attributes mentioned above ($\mathcal{R}$ 4.4).
- The role of a node defines its privileges in terms of access control policy enforcement and management. This allows nodes to enforce policies in a decentralized way and collaborate to manage the access control system by themselves in a distributed way ($\mathcal{R}$ 4.5 - $\mathcal{R}$ 4.8).
- The role of a node can be mapped to categories of nodes and resources, which supports the autonomic administration of role assignments ($\mathcal{R}$ 4.9).
- The role of an authority node autonomously defines its administrative role in the access control administration system. Therefore, the authority nodes can eventually adapt this system to certain changes they might perform on the access control materials ($\mathcal{R}$ 4.10).
- The XACML profile for RBAC [15] is enhanced to express the SRBAC components that represent the organizational structures defined for nodes and resources.

4.4 Secure Relations

The first time two IOrg-AutoNet nodes communicate, they establish a secure relation between them, which lasts as long as the both nodes remain in the network. The type of a secure relation is determined by the organizational attributes of the involved nodes, in addition to their administrative responsibilities in the network. Administrative responsibilities are identified by a node attribute called “Basic Role”, which we define in this section. Hence, a secure relation can be at different trust levels or the same trust level, inter-community or intra-community, and between nodes of different basic roles or the same basic role. A node may have different secure relations of different types with different nodes.

4.4.1 Basic Roles

An IOrg-AutoNet node may acquire certain access rights to perform administrative actions on other nodes and/or their objects. A basic role is assigned to a node in each of its secure relations to identify its administrative capabilities, if any. The assignment of basic roles to nodes is initially the result of a high-level configuration. Eventually, the access control administration system will adapt the basic roles of the different nodes to changes in the network and its environment. We call a role "basic" when it defines administrative privileges for a node irrespectively of its trust level and community membership. Whereas, the trust level, the community membership and the basic role are altogether needed to concretely specify the actual role of a node in a secure relation, as we will see later in this chapter.

Basic roles are assigned to nodes with respect to their capability classes. A node should have a minimum set of capabilities to be able to acquire a certain basic role. Hence, a basic role corresponds to the capability class defining such a minimum set. Basic roles follow a total order based on the total order of the corresponding capability classes. A basic role is greater than another basic role if it defines the same administration privileges at least, which means if it corresponds to a higher capability class. The highest basic role in an IOrg-AutoNet is denoted by A, and is assigned to an *Authority* node in each secure relation binding it to a node under its control. The lowest basic role in an IOrg-AutoNet is denoted by NA (Non-Administrative), and is assigned to a node when it has no administrative rights in a secure relation. The two basic roles NA and A are the initial ones with $NA \leq A$.

The reputation system (cf. Assumption 3.1) or the resource management system (cf. Assumption 3.3) may imply changes in the basic roles of certain nodes because of changes in trustworthiness and availability respectively. Further, those systems may use for a first time basic roles from a totally-ordered predefined set having NA and A as lower and upper bounds respectively. For instance, the resource management system may use the basic role *Delegated Authority* (DA), which defines a subset of the administrative privileges of A, in order to designate certain nodes as the delegated authority nodes of a subnet.

Definition 4.1 *Let $\mathcal{B}$ be the totally-ordered set of basic roles with NA and A as lower and upper bounds respectively and bAbility the function which returns the capability class defining the minimum requirements for acquiring a basic role:*

- $bAbility : \mathcal{B} \longrightarrow \mathcal{K}$
  - $\forall b1, b2 \in \mathcal{B}, (b1 \geq b2) \Leftrightarrow (bAbility(b1) \geq bAbility(b2))$
  - $bAbility(NA) = LD$
  - $bAbility(A) = HD$

Notation 4.1 *Starting by Definition 4.1, and for the rest of the thesis:*

- *We use b, with eventual superscript or subscript indices to denote a basic role (an element of $\mathcal{B}$).*

4.4.2 Secure Relation Definition

A secure relation type is identified by the trust levels, the community memberships and the basic roles of the bound nodes. It is enough to identify the bound nodes to determine their trust levels and communities using the set of mapping functions $\mathcal{F}$ of the network (cf. Definition 3.2). Therefore, a secure relation can be defined using the identities of the bound nodes and their basic roles in this secure relation.

Definition 4.2 *Let SR be the set of all the secure relations and $sRels$ the function which returns the set of secure relations of a node:*

- $SR \subseteq \mathcal{N}^2 \times \mathcal{B}^2$
  - $\forall \rho \in SR, \rho = (x, y, b_x, b_y), b_x$ *is the basic role of* x *in* ρ *and* b_y *is the basic role of* y *in* ρ
  - $\forall \rho 1, \rho 2 \in SR, ((\rho 1 = (x, y, b_x, b_y)) \wedge (\rho 2 = (x, y, b'_x, b'_y))) \Rightarrow ((b_x = b'_x) \wedge ((b_y = b'_y))$
- $sRels : \mathcal{N} \longrightarrow 2^{SR}$
  - $\forall x \in \mathcal{N}, sRels(x) = \{\rho \in SR \mid (\rho = (x1, x2, b1, b2)) \wedge ((x = x1) \vee (x = x2))\}$

Notation 4.2 *Starting by Definition 4.2, and for the rest of the thesis:*

- *We use ρ, with eventual superscript or subscript indices to denote a secure relation (an element of SR).*

4.4.3 Secure Relation Types

The type of a secure relation determines the security material to use to protect the confidentiality and the integrity of the exchanged data. Particularly, our research concerns the secure-relation-based access control in IOrg-AutoNets, as will be discussed in details in this chapter. Figure 4.1 illustrates examples of different types of secure relations in the organizational structure of an IOrg-AutoNet, in which we only have the two initial trust levels and the two initial basic roles. We can notice in Figure 4.1 that an authority node may be seen as a non-administrative node by a non-administrative node of another community. For a non-administrative node, the authority node is specifically the one that manages its community.

The variable set of authority nodes represents a board of security managers. They are together responsible of the self-management of the network in the security context. Besides, they represent the authentication and authorization servers of the network. The set of authority nodes is supposed to vary in an autonomic manner, in terms of the involved nodes and their number. Authority nodes should have secure relations among them in order to be able to cooperate in network administration operations. They also should have secure relations with the nodes of their respective communities in order to be able to control those latter. Hence, the set of secure relations may evolve in an unexpected manner with respect to the evolution of the set of authority nodes.

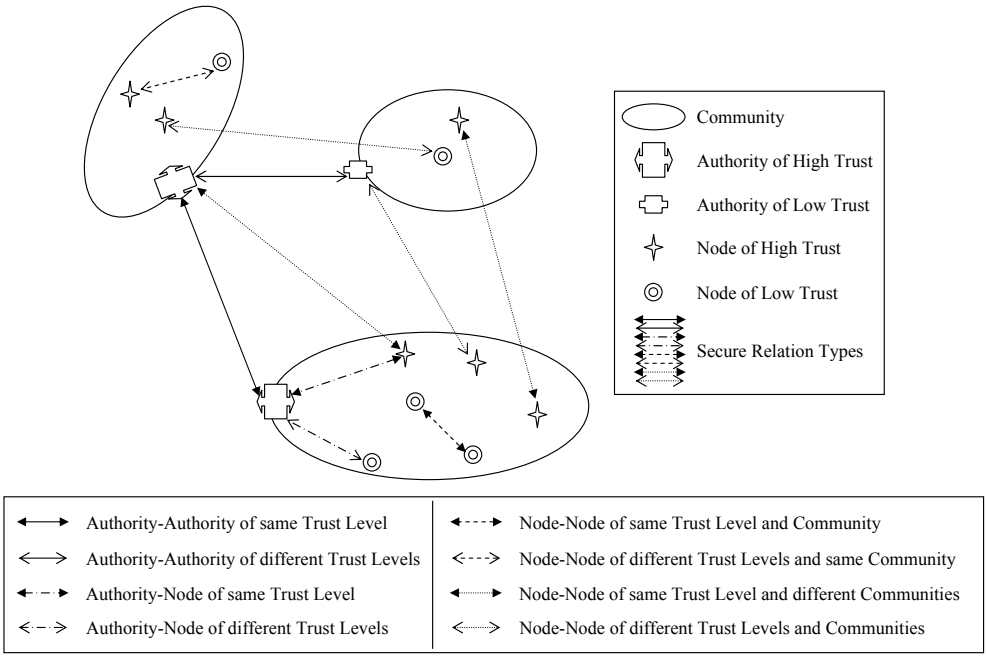


Figure 4.1: Examples of Secure Relation Types in an IOrg-AutoNet

A mutual authentication should take place before a secure relation is established between two IOrg-AutoNet nodes. Authentication between a new node and an authority node takes place during the node insertion operation. Node insertion takes place through a single-hop communication between the new node and the authority node of the selected community, and using a protected short-distance channel, such as an infrared connection. According to the capability class of the new node, it is assigned either a public-key certificate, or a secret key shared uniquely with the authority node of its community. In the first case, the new node generates its key pair, and the authority node assumes the role of the certification authority. In the second case, the authority node securely establish and share the secret key with the new node using a suitable protocol, such as the Diffie-Hellman Key Exchange Protocol.

As for authentication between two non-authority nodes, either they have certificates assigned by their authority nodes and they use them in a certificate-based mutual authentication protocol, or one at least could not be assigned a certificate and the authority nodes intervene as authentication servers. Actually, we only point out here a brief idea about the key infrastructure and the authentication system in an IOrg-AutoNet. We already worked on such authentication schemes in the context of a Master's thesis about security models and protocols in home networks [7]. The network model defined in that Master's thesis can be considered as the groundwork of the IOrg-AutoNet model. In a future work, such issues will be more elaborated, and key establishment and mutual authentication protocols will be proposed specifically for IOrg-AutoNets.

4.5 Node Roles

Let x and y be two IOrg-AutoNet nodes bound by a secure relation ρ. See the previous section for more details about secure relations. Suppose that x should perform a certain action on a certain object hosted by y. The node x may, or may not, be granted the required access rights depending on three of its attributes as evaluated by y. Those attributes are the trust level, the community membership and the basic role. The values of the basic roles of the both nodes depend on the type of ρ (cf. Section 4.4). SRBAC uses a compound node attribute built on the previous basic ones, called *Regular Role*, for assigning permissions to nodes in the context of a secure relation. Consequently, the value of a regular role also depends on the type of the encapsulating secure relation.

4.5.1 Regular Roles

When a node has administrative privileges in a secure relation, which means that its basic role in this secure relation is not NA, its regular role determines the set of communities it may manage. We call such a set of communities the administration scope of the node. Here, we first define a function that allows SRBAC mechanisms to determine the administration scope of a node, and then we define a function to determine the regular role of a node in any of its secure relations:

Definition 4.3 *Let aComm be the function which returns the set of communities or subdivisions of communities belonging to the administration scope of a node with regard to a given basic role:*

- $aComm : \mathcal{N} \times \mathcal{B} \longrightarrow 2^{\mathcal{C}}$

Definition 4.4 *Let RR be the set of regular roles, bRole the function which returns the basic role of a node in a secure relation, and rRole the function which returns the regular role of a node in a secure relation:*

- $RR \subseteq \mathcal{T} \times \mathcal{C} \times \mathcal{B} \times 2^{\mathcal{C}}$
- $bRole : \mathcal{N} \times SR \longrightarrow \mathcal{B}$
- $rRole : \mathcal{N} \times SR \longrightarrow RR$
  - $\forall x \in \mathcal{N}, \forall \rho \in sRels(x), \exists b \in \mathcal{B}, bRole(x, \rho) = b,$
    $rRole(x, \rho) = (tLevel(x), nComm(x), b, aComm(x, b))$

Figure 4.2 shows examples of regular roles belonging to a hierarchy. We will define the regular role hierarchy in the next subsection. Those regular roles belong to a subset of the possible regular roles in an IOrg-AutoNet composed of two communities $c1$ and $c2$, and using the initial trust levels and the initial basic roles. However, we should particularly notice in this figure the use of the two regular roles $(L, \mathcal{N}, NA, \emptyset)$ and $(H, \emptyset, A, \mathcal{C})$. Those two regular roles do not actually belong to RR. No node can be assigned to any of them. They are used for delimiting the lattice representing the regular role hierarchy, as illustrated in the example of this Figure 4.2.

4.5.2 RRH: Regular Role Hierarchy

Just as any organization, an IOrg-AutoNet should have an administrative hierarchy for structuring its principals. The total order of the basic roles already represents such a structure. However, the trust level and the community membership of a node may upgrade or downgrade its administrative capabilities. This is why the administrative hierarchy of an IOrg-AutoNet is actually based on the regular roles of its nodes, rather than their basic roles. Here we define a domination relationship between regular roles, which results in such a hierarchy. The total orders of trust levels and basic roles, and the partial orders defined on the community membership and the administration scope, help building this relationship.

Definition 4.5 *Let as_1 and as_2 be two sets of communities or subdivisions of communities representing two administration scopes, we write $as_1 \succeq as_2$, and we say that as_1 dominates as_2, if and only if the nodes of any community in as_2 belong to a community in as_1, which may be defined formally as follows:*

- $\forall as_1, as_2 \in 2^{\mathcal{C}}$,
  $(as_1 \succeq as_2) \Leftrightarrow (\forall c_2 \in as_2, \exists c_1 \in as_1, c_2 \subseteq c_1)$

Notation 4.3 *Starting by Definition 4.5, and for the rest of the thesis:*

- *We use as, with eventual superscript or subscript indices to denote an administration scope (an element of $2^{\mathcal{C}}$).*

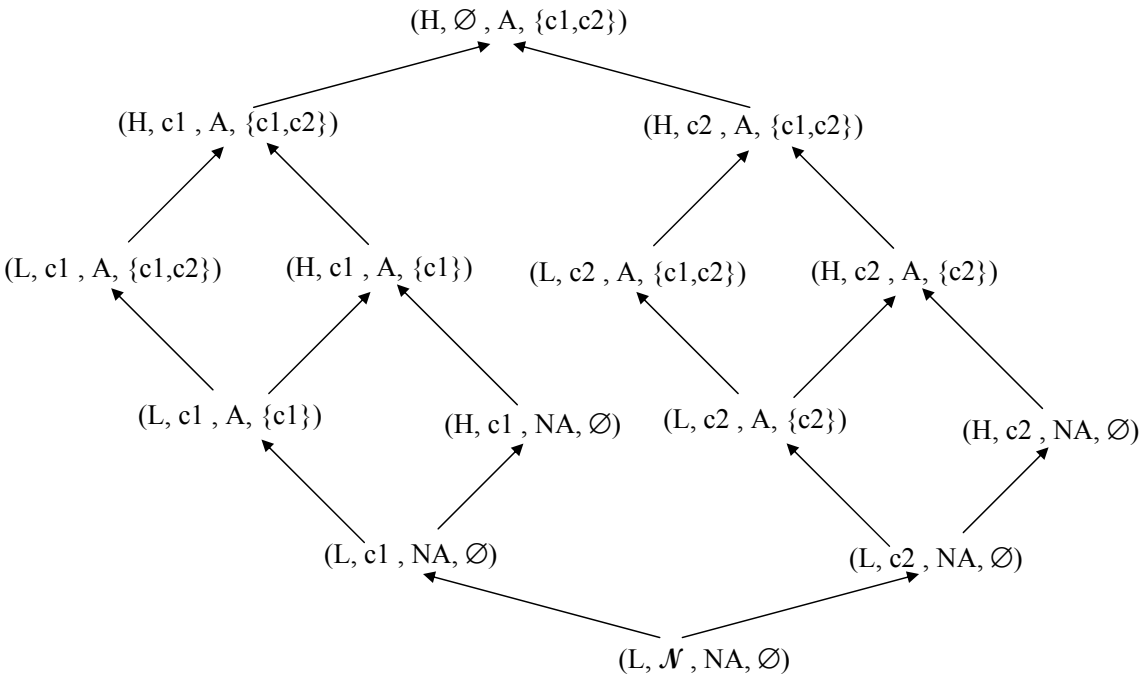
We say that a regular role $r1$ dominates another regular role $r2$ in RR, if and only if the trust level in $r1$ is higher than or equal to the trust level in $r2$, the community in $r1$ is a subset of or equal to the community in $r2$, the basic role in $r1$ is higher than or equal to the basic role in $r2$, and the administration scope in $r1$ dominates the administration scope in $r2$. We use the same rules to compare any regular roles with any of the two abstract regular role values $(L, \mathcal{N}, NA, \emptyset)$ and $(H, \emptyset, A, \mathcal{C})$, which are used for delimiting the regular role hierarchy, as illustrated in the example of Figure 4.2.

Definition 4.6 *Let $r1$ and $r2$ be two regular roles, we write $r1 \succeq_{RR} r2$, and we say that $r1$ dominates $r2$ in RR, if and only if:*

- $\forall r1, r2 \in (RR \cup \{(L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})\})$,
  $r1 = (t1, c1, b1, as1), r2 = (t2, c2, b2, as2)$,
  $(r1 \succeq_{RR} r2) \Leftrightarrow ((t1 \geq t2) \wedge (c1 \subseteq c2) \wedge (b1 \geq b2) \wedge (as1 \succeq as2))$

Notation 4.4 *Starting by Definition 4.6, and for the rest of the thesis:*

- *We use r, with eventual superscript or subscript indices to denote a regular role (an element of RR).*



$(H, \emptyset, A, \{c1, c2\})$: upper RRH bound, abstract role, having all permissions
$(L, \mathcal{N}, NA, \emptyset)$: lower RRH bound, abstract role, having no permissions

Figure 4.2: Example of a Regular Role Hierarchy (RRH)

The operation $\succeq_{RR}$ defines a partial order that creates a lattice hierarchy, in which a regular role inherits access privileges from its descendants. This is the hierarchy that we call RRH (Regular Role Hierarchy). Figure 4.2 illustrates a part of the RRH of a network composed of two communities $c1$ and $c2$, and having the initial sets of trust levels and the two basic roles NA and A. To make the figure as legible as possible, we ignored the case where a node which is not the authority of its community, can be an authority of another one, such as the case of a node having the role $(H, c1, A, \{c2\})$. The direction of the arrows indicates the propagation of access privileges from a role to its senior roles by inheritance.

4.6 Object Attributes

The IOrg-AutoNet users, who are not supposed to be experts in network security, are not likely to be able to specify SRBAC policies. They simply provide some sort of an end-user configuration, in addition to a set of high-level application-dependent objectives. The modules of the security policy system would then derive the equivalent low-level access control policy specification. The way to do this is, on one hand, to specify access scopes for each object according to the enduser configuration, and on the other hand, to associate

permissions with each access scope with respect to the high-level application-dependent objectives. This section describes access scopes and their association with permissions. It also defines object categories based on access scopes, which helps specifying the set of permissions corresponding to a set of objects.

4.6.1 Access Scopes

On one hand, a subset of basic attributes of three possible types are assigned to an object in order to determine its access scopes. On the other hand, an access scope defines the actions that can be performed on the objects with which it is associated. In other words, a set of resources having the same subset of access scopes are associated with a subset of access rights. The basic attributes are specified for an object with respect to the trust level, the community membership and the basic role of the potential access requesting nodes.

Definition 4.7 *Let Att_o be the set of basic attributes of an object o, and $\alpha_{\mathcal{T},t}$, $\alpha_{\mathcal{C},c}$ and $\alpha_{\mathcal{B},b}$ basic attributes identifying the access scopes corresponding to a trust level t, a community c and a basic role b respectively, a node x may access o if and only if:*

- $((t \in \mathcal{T}) \wedge (\alpha_{\mathcal{T},t} \in Att_o) \wedge (tLevel(x) \geq t))$
  $\vee ((c \in \mathcal{C}) \wedge (\alpha_{\mathcal{C},c} \in Att_o) \wedge (x \in c))$
  $\vee ((b \in \mathcal{B}) \wedge (\alpha_{\mathcal{B},b} \in Att_o) \wedge (\exists \rho \in sRels(x), bRole(x, \rho) \geq b))$

Remark 4.1 *In Definition 4.7:*

- *An access scope of an object may determine if a node can access or not regardless of the access type. However, the same access scope defines a set of permissions for each associated object, whereby the access type for a certain object can be determined. So a node can first know if access is authorized according to an access scope, and then it checks the associated permissions.*
- *An object having $\alpha_{\mathcal{T},t}$ as access scope is called* Confidential *at t.*
- *An object having $\alpha_{\mathcal{C},c}$ as access scope is called* Communal *in c.*
- *An object having $\alpha_{\mathcal{B},b}$ as access scope is called* Administrative *for b.*

An object is by default confidential and communal at the trust level and in the community of its hosting node respectively. Besides, the hosting node itself and its security materials are by default administrative objects for the basic role A. Such an initial specification is derived from an initial end-user configuration. Eventually, the adaptation to the network evolution, and possibly to changes in the end-user configuration, would modify the access scopes of objects.

Example 4.1 *Let x be a new node and (t, c, LD) its category. Let y be the authority node of the community c; hence responsible of adding new nodes to it. Suppose that x is*

a device that cannot perform public key operations. In this case, y creates a secret key K and shares it with x using an appropriate key exchange protocol.

The security material K should be hosted by both x and y. By default, the copy of K hosted by x should have the set of attributes $\{\alpha_{\mathcal{T},t}, \alpha_{\mathcal{C},c}, \alpha_{\mathcal{B},A}\}$. *Suppose that the category of y becomes* (t, c', HD) *after a change in its membership, and that y will always manage the community c even if it belongs to another community* c'. *The set of attributes of the copy of K hosted on x should be then updated to become* $\{\alpha_{\mathcal{T},t}, \alpha_{\mathcal{C},c'}, \alpha_{\mathcal{B},A}\}$.

4.6.2 Object Categories

A set of objects having the same subset of basic attributes, and possibly hosted on different nodes, defines an object category. As previously explained, the elements of such a set of objects should be all associated with the same set of permissions. Given that an access scope is identified by the trust level, the community membership or the basic role of the potential access requesting nodes, we can use node attributes to define object categories.

Definition 4.8 *Let* $\mathcal{P}$ *be the set of permissions,* Att_p *the set of basic object attributes associated with a permission p, OC the set of object categories and pPool the function which returns the set of permissions associated with an object category:*

- $OC \subseteq (\mathcal{T} \cup \{nil_{\mathcal{T}}\}) \times (\mathcal{C} \cup \mathcal{N}) \times (\mathcal{B} \cup \{nil_{\mathcal{B}}\})$
  - $((t, c, b) \in OC) \Leftrightarrow (((t, c, b) = (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}})) \vee$
    $((c = \mathcal{N}) \wedge (b = nil_{\mathcal{B}}) \wedge (\exists x \in \mathcal{N}, tLevel(x) = t)) \vee$
    $((b = nil_{\mathcal{B}}) \wedge (\exists x \in \mathcal{N}, tLevel(x) = t, nComm(x) = c)) \vee$
    $(\exists x \in \mathcal{N}, \exists \rho \in sRels(x), tLevel(x) = t, nComm(x) = c, bRole(x, \rho) = b))$
- $pPool : OC \longrightarrow 2^{\mathcal{P}}$
  - $pPool(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}}) = \mathcal{P}$
  - $\forall t \in \mathcal{T}, pPool(t, \mathcal{N}, nil_{\mathcal{B}}) = \{p \in \mathcal{P} \mid \alpha_{\mathcal{T},t} \in Att_p\}$
  - $\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, pPool(t, c, nil_{\mathcal{B}}) = \{p \in \mathcal{P} \mid \{\alpha_{\mathcal{T},t}, \alpha_{\mathcal{C},c}\} \subseteq Att_p\}$
  - $\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall b \in \mathcal{B}, pPool(t, c, b) = \{p \in \mathcal{P} \mid \{\alpha_{\mathcal{T},t}, \alpha_{\mathcal{C},c}, \alpha_{\mathcal{B},b}\} \subseteq Att_p\}$

Notation 4.5 *Starting by Definition 4.8, and for the rest of the thesis:*

- *We use p, with eventual superscript or subscript indices to denote a permission (an element of* $\mathcal{P}$*).*

Remark 4.2 *In Definition 4.8:*

- $nil_{\mathcal{B}}$ *is used in the place of the basic role to ignore it when needed.*
- *See Definition 3.2 and Remark 3.1 for an explanation of* $nil_{\mathcal{T}}$.
- *The set of object categories OC is defined to distribute permissions on trust levels first, then on communities, then on sub-communities at all depths, and finally on basic roles, with respect to the attributes of the potential access requesting nodes.*

4.7 Organizational Structures

The trust levels are totally ordered, and so are the capability classes and the basic roles. Besides, a partial order defined on community membership results from dividing the network into communities. The previous total and partial orders defined on node attributes imply a partial order between node categories and another one between object categories. A partial order defined on a node attribute gives a structure of relationships between its possible values. This allows SRBAC mechanisms to use node categories and object categories in an organized way. This section describes the structures built for the node categories and the object categories respectively.

4.7.1 NS-N: Network Structure for Nodes

According to the node categorization introduced in Definition 3.2, the network nodes are distributed on trust levels first, on communities and their subdivisions afterward and finally on capability classes. The nodes belonging to the same trust level are distributed on communities. The nodes belonging to the same trust level and community may belong to different subdivisions of that community. The nodes belonging to the same trust level and community, or eventually a subdivision of a community, are categorized according to capability classes. Actually, this node categorization defines a tree structure based on a domination relationship between node categories. Here, we define that domination relationship, and give an example of the tree structure.

Definition 4.9 *Let nPool be the function which returns the subset of nodes associated with a node category:*

- $nPool : NC \longrightarrow 2^{\mathcal{N}}$
  - $nPool(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}) = \mathcal{N}$
  - $\forall (t, \mathcal{N}, nil_{\mathcal{K}}) \in NC, nPool(t, \mathcal{N}, nil_{\mathcal{K}}) = \{x \in \mathcal{N} \mid (tLevel(x) = t)\}$
  - $\forall (t, c, nil_{\mathcal{K}}) \in NC, nPool(t, c, nil_{\mathcal{K}}) = \{x \in \mathcal{N} \mid (tLevel(x) = t) \wedge (x \in c)\}$
  - $\forall (t, c, k) \in NC, nPool(t, c, k) =$
    $\{x \in \mathcal{N} \mid (tLevel(x) = t) \wedge (nComm(x) = c) \wedge (cClass(x) = k)\}$

Definition 4.10 *Let nc1 and nc2 be two node categories, we write nc1 $\succeq_{NC}$ nc2, and we say that nc1 dominates nc2 in NC, if and only if the nodes of nc2 belong to nc1:*

- $\forall nc1, nc2 \in NC, (nc1 \succeq_{NC} nc2) \Leftrightarrow (nPool(nc2) \subseteq nPool(nc1))$

Notation 4.6 *Starting by Definition 4.10, and for the rest of the thesis:*

- *We use nc, with eventual superscript or subscript indices to denote a node category (an element of NC).*

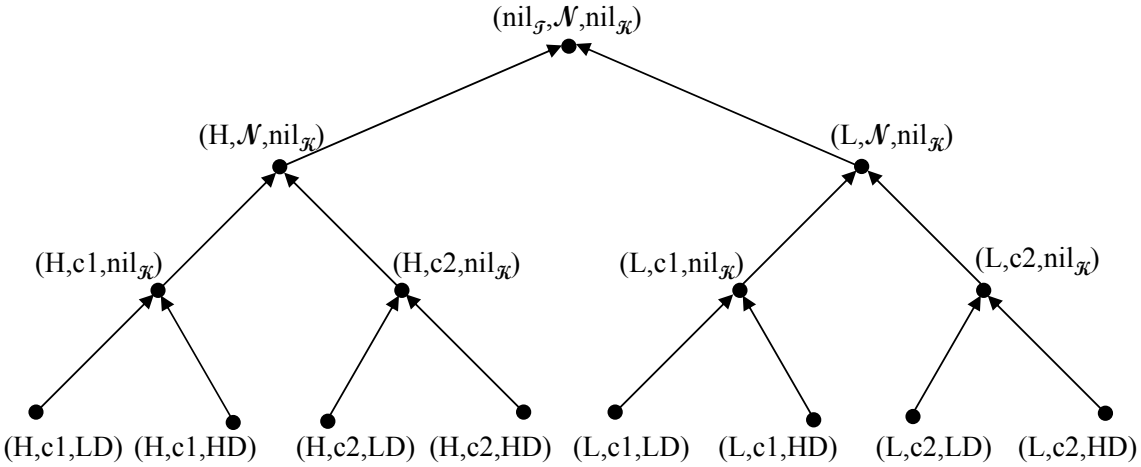


Figure 4.3: Example of a Network Structure for Nodes (NS-N)

The operation $\succeq_{NC}$ defines a partial order that creates a tree structure, in which a category inherits nodes from its descendants. We call this tree NS-N (Network Structure for Nodes). Figure 4.3 illustrates the NS-N of the network that has the regular role hierarchy illustrated in Figure 4.2. We can note in these two figures a mapping between regular roles and node categories. Actually, a node can be assigned a regular role (t, c, b, s) if it belongs to the node category $(t, c, bAbility(b))$. In other words, a node can be candidate for assignment to a regular role if it has the corresponding trust level and community membership, and fulfills the minimum capabilities associated with the corresponding basic role. This mapping allows the access control administration system to be more efficient and scalable in selecting nodes for assignment to regular roles, and to optimize the regular role hierarchy in response to the evolution of the network structure. More details about the autonomic access control administration system of IOrg-AutoNets are provided in Chapter 5. The direction of the arrows in Figure 4.3 indicates that a node category inherits nodes from its juniors.

4.7.2 NS-P: Network Structure for Permissions

According to Definition 4.8, there is an object category associated with all the possible permissions in an IOrg-AutoNet. This root object category corresponds to the root node category in NS-N. Actually, each node category corresponds to an object category. Node categories and object categories are specified using equivalent parameters, which are the basic node attributes, taking into account the mapping between basic roles and capability classes. The capability class in a node category is an estimation of node capabilities in terms of computing and data storage, while the basic role in an object category is a constraint on the administration privileges of a potential access requesting node with respect to such capabilities. This mapping is already defined using the function *bAbility* (see Definition 4.1). In terms of access control administration, node categories are prefixed to the assignment of regular roles to nodes, while object categories are prefixed to the

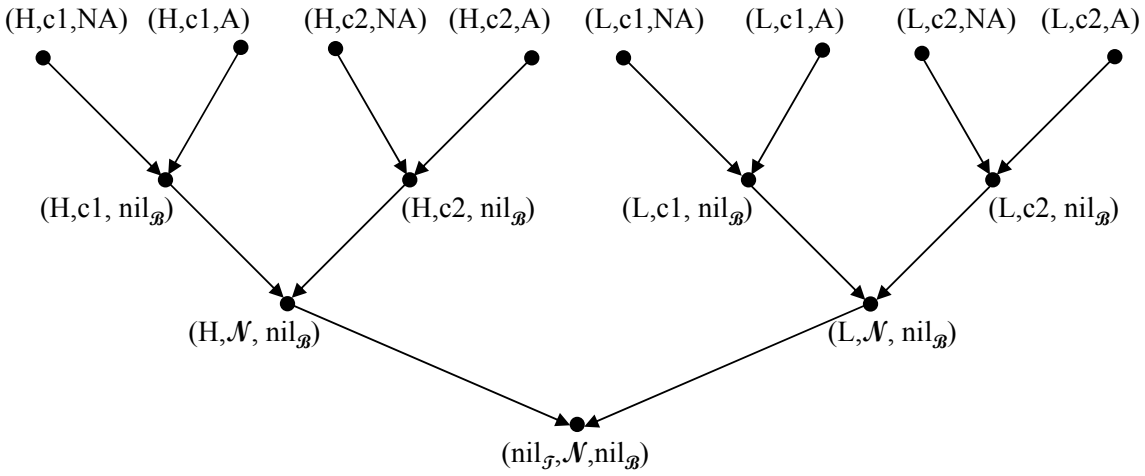


Figure 4.4: Example of a Network Structure for Permissions (NS-P)

assignment of permissions to regular roles. An organizational structure, which is similar to NS-N in the specification of its elements, can be defined for object categories and the associated permissions. This structure is based on a domination relationship between object categories. Here, we define this relationship, and then we give an example of the corresponding structure.

Definition 4.11 *Let* $oc1$ *and* $oc2$ *be two object categories, we write* $oc1 \succeq_{OC} oc2$*, and we say that* $oc1$ *dominates* $oc2$ *in* OC*, if and only if the permissions associated with* $oc1$ *belong to the set of permissions associated with* $oc2$*:*

- $\forall oc1, oc2 \in OC, (oc1 \succeq_{OC} oc2) \Leftrightarrow (pPool(oc1) \subseteq pPool(oc2))$

Notation 4.7 *Starting by Definition 4.11, and for the rest of the thesis:*

- *We use* oc*, with eventual superscript or subscript indices to denote an object category (an element of* OC*).*

The operation $\succeq_{OC}$ defines a partial order that creates a tree structure, in which a category inherits permissions from its ascendants. We call this tree NS-P (Network Structure for Permissions). Figure 4.4 illustrates the NS-P of the network that has the regular role hierarchy illustrated in Figure 4.2 and the NS-N illustrated in Figure 4.3. We can note in this example a mapping between regular roles and object categories, which results in an autonomous assignment of permissions to regular roles, as explained later in Section 4.8.2. We can also note that NS-P is the inverse tree of NS-N, regardless of their respective goals. This mapping between those two organizational structures allows the access control system to be autonomic, as will be elaborated in Chapter 5. The direction of the arrows in Figure 4.4 indicates that an object category inherits permissions from its seniors.

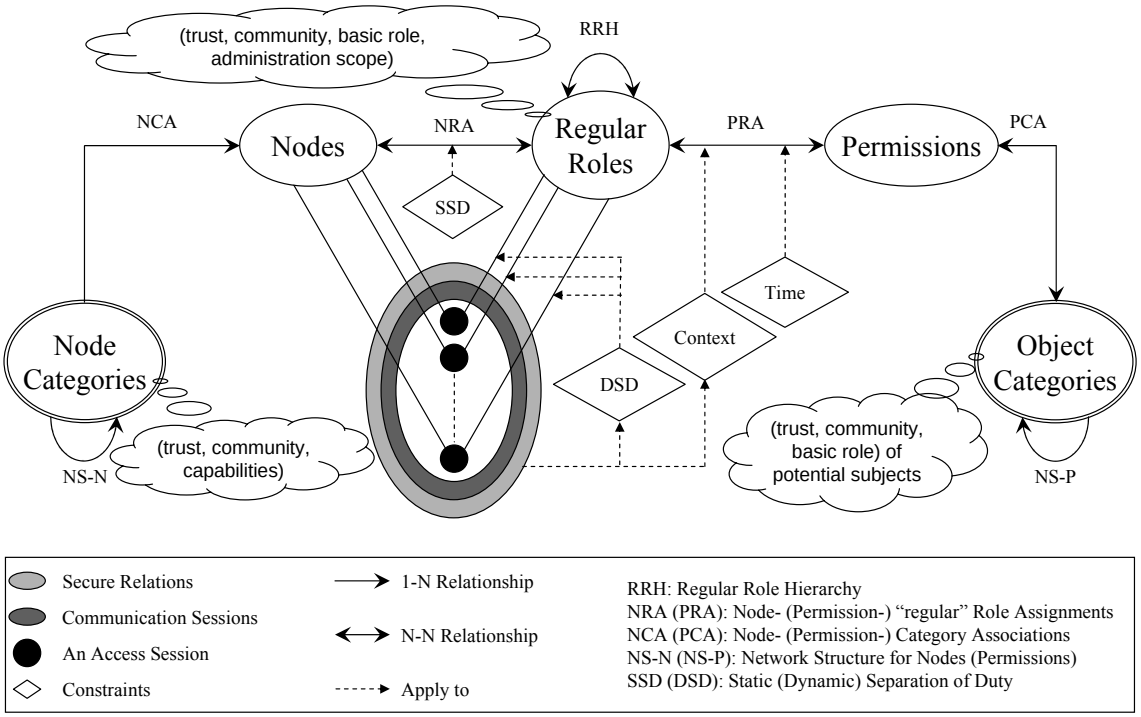


Figure 4.5: SRBAC Model as an Adaptation of RBAC

4.8 SRBAC Definition

We describe in this section the model used for controlling the access to the objects of an IOrg-AutoNet node when requested by another node of the same network within a communication session. This section defines the access control model of IOrg-AutoNets, which we call SRBAC (Secure-Relation Based Access Control). It is based on secure relations because certain node and object attributes are used to specify its policies, and the evaluation of those attributes at policy enforcement time depends on the type of the secure relation which is already established between the access requesting node and the resource hosting node.

As previously elaborated, the values of the attributes of a node in a secure relation define its access rights. Given that a regular role is composed of all the SRBAC-relevant basic node attributes, we may consider that a node acquires permissions in a secure relation depending on the value of only one of its attributes, which is its regular role. This is basically why SRBAC may be be based on the RBAC (Role-Based Access Control) model [89], as illustrated in Figure 4.5. More details about the choice of RBAC as a basis for SRBAC were already provided in Sections 4.2 and 4.3.

Figure 4.5 shows the different components of SRBAC and the relationships between them, as a result of adapting RBAC to IOrg-AutoNets. We already described the core component representing the "Regular Roles" in section 4.5.1 as a compound node attribute. Node categories and object categories were also described as compound attributes in sec-

tions 3.1.4 and 4.6.2 respectively. Moreover, the SRBAC structures NS-N, NS-P and RRH were previously elaborated in section 4.7. The SRBAC components mentioned above, in addition to the two self-explanatory components "Nodes" and "Permissions", are independent of the use of RBAC as a basis for SRBAC. This section defines the other SRBAC components and relationships in the context of RBAC adaptation to IOrg-AutoNets. However, the use of node categories and object categories to pick up nodes and permissions respectively, which is illustrated in Figure 4.5, will not be explained in this chapter. Actually, selecting nodes to assign roles to them, and selecting permissions to be assigned to roles, make part of the SRBAC administration mechanisms. Chapter 5 explains such mechanisms after defining the autonomic administration model accompanying SRBAC.

4.8.1 NRA: Node-Role Assignment

Each node has one regular role at least. Initially, a new node x has the regular role $(L, c_i, NA, \emptyset)$, as a result of the integration in a certain community c_i and the lowest trust level, and without any administration privileges. The regular role of the node x may eventually change, according to a high-level configuration or an autonomic adaptation to context changes. For example, if x becomes highly trusted and if it is selected to be the authority of its community, its regular role becomes $(H, c_i, A, \{c_i\})$. The initial regular role remains assigned to a node, in addition to other eventual regular roles, according to the inheritance relations in RRH. Besides, regular roles may be assigned to a node irrespectively of inheritance while the network evolves. For example, a node belonging to the category (H, c_i, HD), designated as the delegated authority (the basic role DA) of the sub-community s of c_i, and selected to be the authority of another community d_j, has the two independent regular roles $(H, c_i, DA, \{s\})$ and $(H, c_i, A, \{d_j\})$.

Consequently, many regular roles can be assigned to a single node. It is also true that many nodes can be assigned to a single regular role. For instance, all the members of a node category $(x, y, nil_{\mathcal{K}})$ are assigned to the regular role $(x, y, NA, \emptyset)$, either directly or by inheritance. NRA is a many-to-many relationship as illustrated in Figure 4.5. Policy enforcement mechanisms of SRBAC use a set of functions to review the NRA relationship, as detailed in the following definitions.

Definition 4.12 *Let nRoles be the function which returns the set of all the regular roles assigned to a given node, and rNodes the function which returns the set of all the nodes assigned to a given regular role:*

- $nRoles : \mathcal{N} \longrightarrow 2^{RR}$,
  $\forall x \in \mathcal{N}, \forall \rho \in sRels(x), nRoles(x) = \{r \in RR \mid rRole(x, \rho) = r\}$
- $rNodes : RR \longrightarrow 2^{\mathcal{N}}$,
  $\forall r \in RR, rNodes(r) = \{x \in \mathcal{N} \mid \exists \rho \in sRels(x), rRole(x, \rho) = r\}$

Definition 4.13 *Let NRA be the set of direct node-role assignments, and dRoles the function which returns the set of regular roles directly assigned to a node:*

- $NRA \subseteq \mathcal{N} \times RR$

  - $\forall x \in \mathcal{N}, \forall r \in RR, ((x,r) \in NRA) \Leftrightarrow$
    $((r \in nRoles(x)) \wedge (\forall r' \in RR, r' \neq r, r' \succeq_{RR} r, r' \notin nRoles(x)))$

- $dRoles : \mathcal{N} \longrightarrow 2^{RR}, \forall x \in \mathcal{N}, dRoles(x) = \{r \in RR \mid (x,r) \in NRA\}$

For performance and scalability reasons, a policy enforcement mechanism of SRBAC makes use of the inheritance relationships between regular roles when it needs to specify the set of all the regular roles currently assigned to a node. It starts by specifying the set of direct regular roles of the node by looking them up in the set NRA, and then adds all their inherited regular roles. For this purpose, it uses the function *allnRoles* described in Definition 4.14 hereafter. This function extends the function *nRoles* already described in Definition 4.12. Actually, the function *nRoles* returns the highest regular roles a node can activate according to its current secure relations, and the function *dRoles* (cf. Definition 4.13) returns the direct ones among them. Whereas, the function *allnRoles* returns all the regular roles a node can currently activate, and in an efficient and scalable way.

Definition 4.14 *Let allnRoles be the function which returns the direct regular roles of a node and all their inherited regular roles:*

- $allnRoles : \mathcal{N} \longrightarrow 2^{RR}, \forall x \in \mathcal{N},$
  $allnRoles(x) = \{r \in RR \mid \exists r' \in dRoles(x), r' \succeq_{RR} r\}$

Remark 4.3 *In Definition 4.14:*

- *See Definition 4.13 for the specification of the function dRoles.*

4.8.2 PRA: Permission-Role Assignment

The SRBAC model does not define a specific format for a permission. SRBAC essentially cares about the assignment of permissions to object categories, and eventually to regular roles. Actually, The Permission-Role Assignment (PRA) relationship, illustrated in figure 4.5, is based on the mapping between regular roles and object categories on one hand, and between object categories and permission pools on the other hand. This means that the set of access privileges represented by a regular role is automatically deduced. We explained in section 4.6.2 how a set of permissions is associated to every object category. We describe in the following definition how a regular role is associated to a set of object categories; hence to a set of permissions.

Definition 4.15 *Let rObjects be the function which returns the set of object categories associated with a regular role by mapping:*

- $rObjects : RR \longrightarrow 2^{OC}$

  - $\forall r \in RR, (r = (t, c, NA, \emptyset)) \Leftrightarrow (rObjects(r) = \{(t, c, NA)\})$
  - $\forall r \in RR, ((r = (t, c, b, as)) \wedge (b \neq NA))$
    $\Leftrightarrow (rObjects(r) = \{(t, c', b) \in OC \mid c' \in as\})$

Obviously, a single regular role may be assigned to many permissions. This is due to the possible association of more than one permission with the object categories associated with that regular role, and because this latter may inherit permissions from junior regular roles in RRH. It is also possible that a single permission be assigned to many regular roles because a permission assigned to a regular role is also assigned to its senior regular roles in RRH. Moreover, the regular roles assigned to a single permission may be independent with respect to the domination relation $\succeq_{RR}$ (cf. Definition 4.6). For instance, the authority nodes may belong to different independent communities, which means that they will have different independent regular roles in RRH, while they may have the same permission to access a set of shared resources in order to collaborate to accomplish a certain administration task. Actually, PRA is a many-to-many relationship, and SRBAC provides functions to review it as described in the following definitions. These functions are efficient and scalable because they first search a set of direct permission-role assignments, which is relatively straightforward, and then they make use of the inheritance between regular roles to achieve their respective outputs.

Definition 4.16 *Let PRA be the set of direct permission-role assignments, $rPerms$ the function which returns the set of permissions assigned to a regular role, and $pRoles$ the function which returns the set of regular roles assigned to a permission:*

- $PRA \subseteq RR \times \mathcal{P}$
  - $\forall r \in RR, \forall p \in \mathcal{P}$,
    $((r,p) \in PRA) \Leftrightarrow (\exists oc \in rObjects(r), p \in pPool(oc))$
- $rPerms : RR \longrightarrow 2^{\mathcal{P}}, \forall r \in RR$,
  $rPerms(r) = \{p \in \mathcal{P} \mid \exists r' \in RR, ((r',p) \in PRA) \wedge (r \succeq_{RR} r')\}$
- $pRoles : \mathcal{P} \rightarrow 2^{RR}, \forall p \in \mathcal{P}$,
  $pRoles(p) = \{r \in RR \mid \exists r' \in RR, ((r',p) \in PRA) \wedge (r \succeq_{RR} r')\}$

4.8.3 Access Sessions

A node must create an access session to perform certain actions on certain shared resources. Many access sessions may be created during a session of communication between two nodes already bound by a secure relation, and for any of the both nodes. The type of the secure relation defines the permissions of the access requesting node, as previously explained. A node activates the regular role assigned to it in the secure relation, which allows it to acquire the required privileges. The fact that a secure relation is the context of an access session is illustrated in figure 4.5.

In the RBAC model [89], an access session is assigned to only one user on one hand, and to many activated roles of that user on the other hand. However, a node (a user in SRBAC) activates only one regular role in an SRBAC access session, which is its regular role in the encapsulating secure relation. Actually, a node may also have other regular roles in a secure relation by inheritance, but there is no need to activate them because the activated regular role already inherits their permissions. Therefore, an access session in

SRBAC is assigned to only one node and to only one regular role as illustrated in figure 4.5. Nevertheless, a node may create parallel access sessions, encapsulated each by one of its secure relations, and may activate a different regular role in each of them. In other words, a node in SRBAC may activate many regular roles simultaneously, such as a user in RBAC, but in a decentralized and distributed context.

4.8.4 Constraints

As illustrated in Figure 4.5, different types of constraints may be applied on the relationships that bind the components of SRBAC. A constraint may be static, which means that it is applied permanently on a relationship and irrespectively of any access session. This is the case of SSD (Static Separation of Duty) constraints, which are applied on NRA (Node-Role Assignments). An SSD constraint in SRBAC prevents assigning two conflicting roles to the same node, just as the case of any variant of RBAC [89].

Other constraints may be dynamic, which means that they are applied on the creation of access sessions. A dynamic constraint may also be applied after the creation of an access session, which means on the access operation itself, in the case of ongoing access control. However, the current version of SRBAC, which is illustrated in Figure 4.5, concerns the traditional beforehand access control. Some dynamic constraints are based on the requirements of the encapsulating secure relations. This is the case of DSD (Dynamic Separation of Duty) constraints, which are applied on session-role assignments. A DSD constraint in SRBAC prevents a node from activating two of its regular roles in two simultaneous access sessions created in the contexts of two different secure relations, if such a parallel activation of regular roles may present a conflict of interest. There is no need for DSD constraints in one SRBAC access session, where only one regular role is activated.

There may be other dynamic constraints that depend upon secure relations in SRBAC. They are the context-based constraints, which are applied on PRA (Permission-Role Assignment), as illustrated in Figure 4.5. Actually, the context of an access session is a secure relation, which means that any dynamic constraint based on the requirements of the encapsulating secure relation is a context-based constraint. So we should have considered that DSD constraints are context-based. However, a DSD constraint in SRBAC involves many simultaneous access sessions of a single node, which means that its scope is larger than the context of a single access session. More precisely, the Separation of Duty (SoD) constraints in SRBAC, either static (SSD) or dynamic (DSD), have a network scope. They are actually based on the context of the network as a whole. As for a context-based constraint of the type illustrated in Figure 4.5, it has a single secure relation as a context. Such a constraint compares the attributes of the access requesting node and the resource hosting node, in order to cancel certain permissions of the activated regular role if needed.

Some constraints are dynamic but independent of the encapsulating secure relation. This is the case of time-based constraints, which are applied on PRA, as illustrated in Figure 4.5. They are dynamic because they define certain time conditions to check during the creation of an access session. The time here may indicate certain date and hour, but also the moment when a certain event is triggered. Time-based constraints are needed in infrastructureless environments, where any component may not be available at certain

moments, or certain access operations should be delayed for application-dependent reasons. For instance, in a mission-based network, a subdivision of a community may be created, which may imply the assignment of regular roles for the first time to certain nodes, but the activation of these regular roles should wait for the starting time of the mission.

The Separation-of-Duty (SoD) constraints are used in almost all the security systems based on RBAC [89]. Actually, SoD constraints make part of the NIST proposed standard for RBAC [85]. We particularly add the constraints based on context and time in SRBAC to fulfill the requirements of the dependence upon secure relations and the evolution of the network structure respectively. Context and time constraints are actually used in existing solutions based on RBAC. The OrBAC model [5] uses a context component that may define context-based constraints. The TRBAC model [21] uses time-based constraints for enabling or disabling roles in access sessions.

4.8.5 Formal Definition of the SRBAC Model

After describing the different components of the SRBAC model, and giving formal definitions for certain ones and their related functions, we may conclude this section by the following formal definition of the SRBAC model:

Definition 4.17 *SRBAC has the following components:*

- $\mathcal{N}$*: the set of network nodes (cf. Definition 3.2).*
- $\mathcal{S}$*: a set of access sessions.*
- CS*: a set of communication sessions.*
- SR*: the set of secure relations (cf. Definition 4.2).*
- RR*: the set of regular roles (cf. Definition 4.4).*
- RRH*: the partial order relation on* RR *called regular role hierarchy (cf. Definition 4.6).*
- NC*: the set of node categories (cf. Definition 3.2).*
- $NS-N$*: the partial order relation on* NC *called network structure for nodes (cf. Definition 4.10).*
- OC*: the set of object categories (cf. Definition 4.8).*
- $\mathcal{P}$*: the set of permissions (cf. Definition 4.8).*
- $NS-P$*: the partial order relation on* OC *called network structure for permissions (cf. Definition 4.11).*
- NRA*: the set of direct node-role assignments (cf. Definition 4.13).*
- PRA*: the set of direct permission-role assignments (cf. Definition 4.16).*

- *NCA: a set of associations between nodes and node categories.*
- *PCA: a set of associations between permissions and object categories.*
- *$node : \mathcal{S} \longrightarrow \mathcal{N}$: a function mapping an access session s_i to a single node $node(s_i)$.*
- *$activeRoles : \mathcal{S} \longrightarrow 2^{RR}$: a function mapping an access session s_i to a set of active regular roles such that $activeRoles(s_i) \subseteq allnRoles(node(s_i))$. See Definition 4.14 for the specification of the function $allnRoles$.*
- *$canAccess : \mathcal{S} \times \mathcal{P} \longrightarrow \{true, false\}$: a predicate defined as $(canAccess(s, p) = true) \Leftrightarrow (\exists r \in activeRoles(s), p \in rPerms(r))$. See Definition 4.16 for the specification of the function $rPerms$.*

Notation 4.8 *Starting by Definition 4.17, and for the rest of the thesis:*

- *We use s, with eventual superscript or subscript indices to denote an access session (element of $\mathcal{S}$).*

4.9 Policy Specification

A traditional access control policy is a set of rules granting each a right to a subject to perform an action on an object. So each rule defines a relationship between a subject and a permission. Figure 4.5 shows that in SRBAC there is not a direct relationship between a node, which is the subject in our case, and a permission. Such a relationship is actually the result of activating NRA (Node-Role Assignment), PRA (Permission-Role Assignment) and RRH (Regular Role Hierarchy) associations in an access session, taking into account possible constraints. Therefore, the rules of an SRBAC policy define the components and the relationships illustrated in Figure 4.5.

4.9.1 The XACML Standard

We propose to specify an SRBAC policy using the OASIS eXtensible Access Control Markup Language (XACML) [4]. Its rules can be derived from an enduser configuration and high-level application-dependent objectives, because it is a declarative high-level language. Human administrators can use it when necessary to manage policies, because it has a user-friendly syntax for experts. Enforcement modules can use it to derive low-level policies, because they can parse its different elements according to a predefined scheme. Autonomic administration mechanisms can use it to respecify rules in a logic-based language in order to analyze and modify them if needed, because it can provide information about the semantic. It is an open standard based on the well-recognized eXtensible Markup Language (XML), and already used in many information systems, which is suitable for exchanging rule information in the heterogeneous environment of IOrg-AutoNets.

More specifically, we use the RBAC profile of XACML v2.0 [15] to specify SRBAC policies. This OASIS standard uses the ANSI INCITS 359-2004 [2] standard defined for RBAC, which is based on the RBAC reference model [85] adopted by the NIST institute

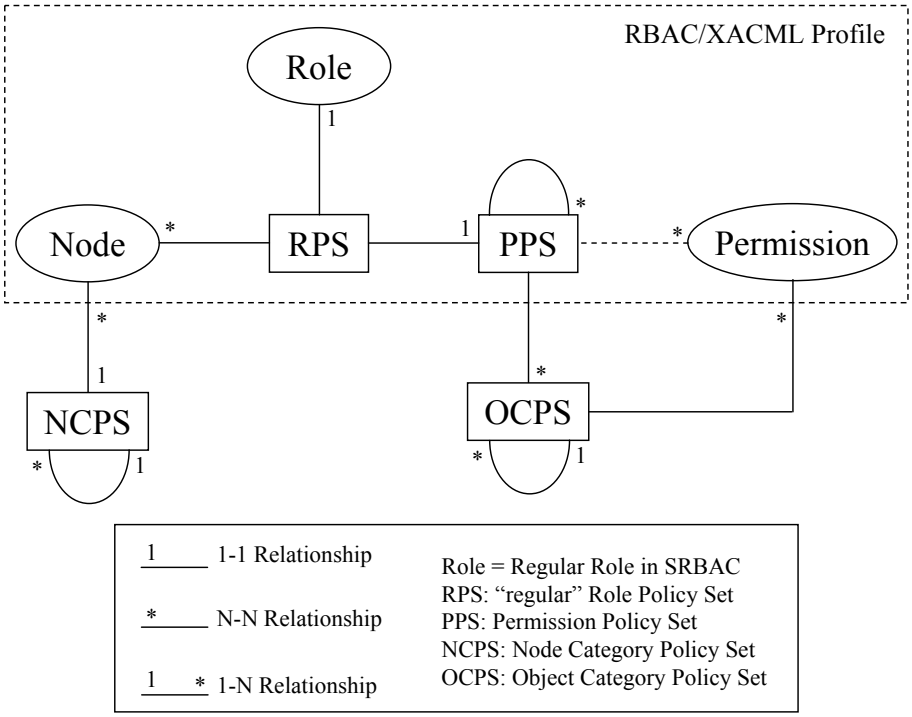


Figure 4.6: RBAC Profile of XACML V2.0 Extended with SRBAC Entities

[3]. The OASIS XACML 2.0 RBAC profile [15], which we call "the Standard" in the rest of this section, defines an XML-based language for specifying RBAC elements. However, it only covers core and hierarchical variants of the reference RBAC model [85], which may imply a lack in the resulting SRBAC policy specification if constraints should be considered. This lack does not currently make a problem in our work. Actually, our access control administration system does not currently support constraint management, so we do not necessarily need to express them in SRBAC policies in our current work.

We extend the elements of the Standard to express the components of SRBAC, which are illustrated in figure 4.5, except for constraints. Access sessions and their associations are not expressed either in SRBAC policies, because they are enforcement components. Actually, we can use the elements of the Standard without modifying their definitions because regular roles and their associations in SRBAC are variants of roles and their associations in RBAC. However, organizational components, such as node categories and object categories and the associated structures NS-N and NS-P, are not considered in the Standard. Therefore, we extend it with new XACML 2.0 elements to express those SRBAC components and their associations. Figure 4.6 illustrates the basic elements of the Standard and the SRBAC extensions.

4.9.2 Using Standard Entities

In the Standard, a role is defined as an XACML attribute for an entity representing a user in RBAC. This definition is suitable for SRBAC, where a user is a node, and a regular role is defined as a node attribute (cf. Section 4.5.1). SRBAC policies use one attribute for regular roles having the `AttributeId "&role;"`, as defined in, and recommended by, the Standard. Given a regular role $r \in RR$, the corresponding attribute value for a node assigned to r is written `"&roles;r"`, where `"&roles;"` is the XML entity defined in the Standard for expressing possible role values.

We use the `Role <PolicySet>` (RPS) of the Standard to associate the nodes sharing the same regular role with a set of permissions. This association is expressed as a reference to a `Permission <PolicySet>` (PPS) in an RPS. A PPS instance is supposed to express every detail about the permissions assigned to a given role. We make a slight modification to a PPS instance to adapt it to SRBAC requirements. Instead of referencing all the information about permissions, we only reference the target resources (object categories). However, the PPS specification does not change, because a rule can contain only resource information in XACML v2.0. Details of the permissions associated to each object category are specified in a separate policy set that we add to the Standard as explained hereafter. As for representing a role hierarchy, we use the approach of the Standard without modification. The Standard expresses role inheritance by referencing the PPS instances associated with junior roles in the PPS instance associated with their senior role.

4.9.3 Extending the Standard

We add a new attribute for users (nodes). It has the `AttributeId &nodecategory;`. We also add the entity `&nodecategories;` to express the possible values of that attribute. This new attribute is used to specify node categories and the NS-N structure. As for specifying object categories and the NS-P structure, there is no need for defining new entities because we consider a resource in the Standard as a representation of an object category. Actually, we do not generally need to handle an object individually in SRBAC, because a permission is supposed to be associated to a category of objects. However, if needed, the basic attributes of nodes and objects can be added to the Standard and used to handle individual nodes and objects. Actually, an advantage of using the XACML language is that it supports fine-grained access control through the generic use of attributes in defining access rules.

We add the `NodeCategory <PolicySet>` (NCPS) to express the NS-N structure. An instance of NCPS associates the nodes sharing the same node category with another instance of NCPS representing its senior node category in NS-N. This is to say that the nodes belonging to the node category identified by an NCPS instance also belong to the senior node category referenced in this instance. We also add the `ObjectCategory <PolicySet>` (OCPS) to express the association between an object category (resource in the Standard) and a set of permissions. A permission is specified as a relation between an action, the object category and possibly a condition. In addition to expressing permissions, an OCPS instance references the OCPS instance of the junior object category, which gives the rep-

resentation of the NS-P structure. This is to say that the permissions associated to the object category of an OCPS instance are also associated to its junior object category.

Detailed samples of the previous XACML elements are provided in Appendix A, as a part of an example about applying SRBAC in a Home Network. Those samples will help understanding how we use XACML instructions in negotiating SRBAC policies. Actually, XACML is not basically defined for decentralized collaborative systems. Our approach of filling this gap is to make nodes negotiate XACML elements instead of enhancing the XACML language with support for decentralization. More details about access control administration and security policy negotiation are provided in Chapters 5.

4.10 Conclusion

This chapter described an access control model for Infrastructureless Organizational Autonomic Networks (IOrg-AutoNets), and proposed a specification language for its policies. The model is called Secure Relation Based Access Control (SRBAC). It aims at protecting shared resources during a communication between two nodes in the context of a secure relation between them. The SRBAC model considers the IOrg-AutoNet as an organization managed by its own nodes according to different administrative roles. Therefore, authorization decisions are role-based, and a role hierarchy is used to organize node privileges. The SRBAC model is decentralized and context-aware, in conformity with the infrastructureless nature of IOrg-AutoNets. Decentralization is achieved through the integration of privileges of access control administration into node roles. Context-awareness is achieved through the integration of environment variables representing the infrastructureless context into node roles as well. Those environment variables specify the trustworthiness, the availability, the administrative capabilities and the authority scope.

For the particular purposes of an autonomic network, the SRBAC model is defined in a way that allows the access control administration system to have the properties of autonomic computing. In this context, the SRBAC model supports distribution, collaboration, self-management and self-adaptation. By incorporating components representing the network organizational structure in SRBAC, whereby nodes and resources can be distributed on categories, the nodes of the network can share the administration tasks and collaborate to accomplish them. Self-management is possible because, on one hand, certain node roles define capabilities of access control administration, which allows the administrative counterpart of SRBAC to be based on SRBAC itself. On the other hand, the mapping between node roles and categories of nodes and resources provides self-configuration capabilities for node-role and permission-role assignment administration. Finally, self-adaptation is possible because the context-aware components of SRBAC can be modified in response to changes in the network context, which eventually changes the components of the administrative model itself because they are based on the components of SRBAC.

The SRBAC model is based on the Role-Based Access Control (RBAC) model [89]. The RBAC model already provides required components, such as roles, role hierarchies and role associations. This chapter explained how RBAC was adapted to the other requirements of IOrg-AutoNets, basically by using context information in role specification and by

incorporating components representing the organizational structure of the network. This choice of RBAC was also motivated by the fact that a distributed administrative model is already defined for RBAC, which can be extended to define an administrative counterpart for SRBAC. The Administrative SRBAC (ASRBAC) model, which is described in Chapter 5, is an extension of the Administrative RBAC (ARBAC02) model [73] based on adding support for context-awareness, collaboration, self-management and self-adaptation.

SRBAC policies should be specified using a standard language that is flexible and platform-independent, which is required in a heterogeneous environment. We proposed in this chapter a specification language based on the OASIS eXtensible Access Control Markup Language XACML v2.0 [4]. More specifically, the language we proposed is an extension of the RBAC profile of XACML v2.0 [15]. We extend this profile with elements used to express the SRBAC components that represent the organizational structure of the network. However, XACML assumes the presence of a central authority, which cannot be the case in IOrg-AutoNets. Therefore, the nodes may need to negotiate possible policy modifications, when they manage changes in the network context, before applying them locally. Section 5.6 in Chapter 5 proposes a sketch of a policy negotiation algorithm for IOrg-AutoNets.

An interesting future work could be an enhancement of SRBAC to avoid potential limitations related to performance and scalability. There might be for example such kinds of problems in building the RRH (Regular Role Hierarchy) in a scenario where a considerable number of differences between nodes might result from the highly variable topology of the network. For instance, a mission-based military MANET may include thousands of tiny information gathering robots thrown in a hostile area. We have to consider each node as a community by itself in such a scenario, which means that there will be at least as much regular roles as nodes. It might be a formidable task to manage the RRH in such a network.

Chapter 5

ASRBAC: The Administration Model

After defining an access control model for IOrg-AutoNets, which we denoted by SRBAC (cf. Chapter 4), we need to study the administration of the corresponding access control system. More specifically, we need to know how to configure the components modeled by SRBAC, specify SRBAC policies, and adapt the component configuration and the policy specification to the evolution of the network. We also need to know who will perform the previous tasks, and what techniques it should have to accomplish them. This chapter explores the SRBAC administration requirements, defines the Administrative SRBAC (ASRBAC) model that fulfills those requirements, elaborates the administration techniques modeled by ASRBAC, specifies the ASRBAC policies in a generic language-independent manner, and finally makes a focus on node collaboration in terms of negotiating the adaptation of access control policies.

Section 5.1 recalls the SRBAC requirements related to administration, which were described in Section 4.1. Those requirements are distribution, decentralization, self-management and self-adaptation. Afterward, it goes into the details of more requirements specific to the administrative model, which are context-aware updates, self-aware mappings and policy negotiation.

Section 5.2 discusses existing access control administration models. It mainly refers to ARBAC02 [73] as a basis for ASRBAC, AdOr-BAC [5] as a competing model and AROBAC07 [107] as a basis for future extensions.

Section 5.3 explains our contributions to access control administration with regard to the Autonomic Computing Paradigm [50]. It talks about using context-awareness and self-awareness to monitor the network and the access control system itself. It points out the ability of ASRBAC to adapt certain access control components to the detected changes by analyzing monitoring information. It emphasizes the self-configuration and self-adaptation of certain components. Besides, it points out the node cooperation mechanisms employed in an ASRBAC-based system.

Section 5.4 defines the ASRBAC model. It mainly explains how ASRBAC is actually a variant of SRBAC itself, and how the ASRBAC components are autonomously defined

by the SRBAC components. The section shows how this dependency between ASRBAC and SRBAC is behind the self-adaptation capabilities of the access control system. The self-configured specification of administrative roles and their hierarchy are presented. The administrative node-role and node-permission assignments are described. Finally, a language independent specification of generic ASRBAC policies is given.

Section 5.5 describes the autonomic functionality supported by ASRBAC. After specifying the environment variables used in monitoring the context, it explains four different types of autonomic actions. They are the context-aware predefined self-management, the autonomic control loop, the mapping-based self-adaptation and the autonomous evolution. Finally, it points out the collaboration of nodes to accomplish those different autonomic operations.

Section 5.6 describes how authority nodes negotiate SRBAC policies while collaborating to accomplish one of the administrative tasks described in the previous sections, which is the modification of permissions. It proposes a negotiation algorithm based on evaluating the similarity of XACML rules in two policies, and then using the evaluation result in unifying the SRBAC policy modifications.

Section 5.7 concludes the chapter with a summary of ASRBAC focusing on the details of our contribution to access control administration. It also points out the issues to be handled in future works.

5.1 Administration Requirements

Because an IOrg-AutoNet is basically an infrastructureless network, we cannot depend on a central network authority for SRBAC administration. Actually, there should be a set of authorities collaborating to maintain a consistent implementation of SRBAC components and policies. The SRBAC requirements $\mathcal{R}$ 4.6 and $\mathcal{R}$ 4.7 already indicated the need for distribution and collaboration in the administration of the access control system.

Requirement $\mathcal{R}$ 5.1 *The SRBAC administration model should define distributed and collaborative techniques.*

In an autonomic network, the security administration should be basically assumed by the network nodes themselves. In other words, certain administrative privileges defined by specific regular roles should allow a set of qualified nodes to share the access control administration and collaborate to achieve its objectives. This actually fulfills the Requirement $\mathcal{R}$ 5.1 described above. This also indicates that the SRBAC model should be used as a basis for its own administration model. Such a linkage allows the administration model to make use of the autonomic computing properties supported by the SRBAC components, and makes its components able to adapt to the effects of its own mechanisms. Actually, the previous autonomic computing requirements were already indicated by the SRBAC requirements $\mathcal{R}$ 4.8 to $\mathcal{R}$ 4.10.

Requirement $\mathcal{R}$ 5.2 *The SRBAC administration model should be based on the SRBAC model itself, essentially through a mapping between regular roles and administrative roles, which helps acquiring self-management and self-adaptation properties.*

In addition to the previous requirements $\mathcal{R}$ 5.1 and $\mathcal{R}$ 5.2, which are based on the SRBAC requirements $\mathcal{R}$ 4.6 to $\mathcal{R}$ 4.10, this section describes hereafter further specific requirements based on the techniques that should be employed for SRBAC administration.

5.1.1 Context-Aware Updates

The SRBAC component NS-N (Network Structure for Nodes; cf. Section 4.7.1) represents the IOrg-AutoNet structure in the access control system. Initially, NS-N is specified according to a high-level configuration done by the endusers at the network deployment time. Afterward, it should adapt to the evolution of the network. The SRBAC administration system is supposed to update the NS-N component in response to certain critical changes in the network context, such as using a trust level or a capability class for the first time, or integrating or revoking a community (cf. Section 3.2).

Context-awareness may also be needed in the administration of Node-Role Assignments (NRA; cf. Section 4.8.1). Actually, when a basic attribute of a node changes, the set of regular roles assigned to that node may change (cf. Section 4.5.1). A basic node attribute changes in response to a modification in the high-level security configuration, or an evolution of the network. In both cases, the SRBAC administration system should be able to capture the nodes' contextual information that may imply changes to NRA.

Permission-Role Assignments (PRA) are self-configured, due to associations between permissions and object categories on one hand, and associations between object categories and regular roles on the other hand (cf. Section 4.8.2). The evolution of the network may change Permission-Role Assignments indirectly, because it may imply the modification of one or both of those associations of object categories. Actually, context-awareness is needed for modifying the associations between permissions and object categories. Such a modification needs certain context information representing the up-to-date high-level security configuration and application objectives.

Requirement $\mathcal{R}$ 5.3 *The SRBAC administration model should support the detection of changes in the network context.*

5.1.2 Self-Aware Mappings

The regular roles and the set of inheritance relationships between them, which we call the Regular Role Hierarchy (RRH), are specified according to a mapping between regular roles and node categories (cf. Section 4.5.2). This means that they should be respecified using that same mapping when the node categories change.

The assignment of regular roles to nodes is also based on the mapping between node categories and regular roles. Actually, a node is looked up in a corresponding node category representing certain administrative capabilities (cf. Section 3.1.4), to be assigned to a certain regular role.

Changes in basic node attributes may modify the access scopes, which implies a recategorization of objects (cf. Section 4.6), and eventually a rederivation and redistribution of permissions for each object category. Changes in node attributes are detected as changes

in node categories. So updating object categories may be a result of a mapping with modified node categories.

Permission-Role Assignments (PRA; cf. Section 4.8.2) are expressed using autonomous associations between regular roles and object categories, given that an object category is already associated with a set of permissions. In other words, when regular roles change, PRA is modified by mapping. Whereas, changes in the associations between permissions and object categories do not cause changes in the way PRA are expressed, but the privileges of regular roles change implicitly.

Requirement $\mathcal{R}$ 5.4 *The security administration model should allow the authority nodes to detect changes in the SRBAC components.*

5.1.3 Policy Negotiation

When a critical change takes place while the network evolves, the SRBAC component that represents the Network Structure for Nodes (NS-N) changes. Each authority node changes the NS-N part corresponding to its administration scope, but it needs to validate changes with other authority nodes that may share with it certain subdivisions of that NS-N part. Moreover, the final changes on all the NS-N tree should be mutually validated by all the authority nodes.

When a regular role is assigned to a node, the computing and storage capabilities of that latter are used to select it among others in its node category. However, many nodes in a category may share the same capabilities allowing them to acquire a regular role, while the number of instances of that regular role in the node category is limited with respect to the number of candidate nodes. For example, we need one authority in a community, while this latter may include many nodes that can assume that role. Node election mechanisms are used to select a node for role assignment. Nevertheless, many authority nodes may need to work together on such an election with respect to their respective authority scopes.

When the Network Structure for Permissions (NS-P) component adapts to NS-N changes, or when the high-level security specifications change, the set of permissions associated to each object category should be respecified. Each authority node can apply such respecification autonomously. Nevertheless, SRBAC policy instances should be the same in the whole network. Therefore, after each NS-P modification, the authority nodes need to unify their permission specifications and the associations between permissions and object categories.

Requirement $\mathcal{R}$ 5.5 *The security administration model should allow the authority nodes to negotiate SRBAC policies.*

5.2 Related Work

The ASRBAC model is based on the ARBAC02 model [73]. This latter is an administrative counterpart of the RBAC model [89]. This was one reason to choose it as long as the SRBAC model is based on the RBAC model. A second reason is the use of RBAC

itself to define ARBAC02, which is compatible with our requirement of basing ASRBAC on SRBAC itself for self-management and self-adaptation purposes (cf. Requirement $\mathcal{R}$ 5.2). The other reasons are related to the conformity of ARBAC02 with the organizational properties of IOrg-AutoNets. Actually, ARBAC02 is a distributed model that makes use of the structure of the encapsulating organization in assigning nodes and permissions to roles. As elaborated in [73], the integration of organizational aspects provided solutions for many problems raised by the predecessor models ARBAC97 [84] and ARBAC99 [86]. So the distribution of administration in ARBAC02 and the support of organizational structures are other two reasons for choosing it as a basis for ASRBAC. Moreover, RBAC and ARBAC02 are generic and flexible in specifying and using the organizational components. This allowed us to define such components in SRBAC and use them in ASRBAC in a way to extend ARBAC02 with a support of Autonomic Computing, as will be explained later in Section 5.5 (see Figure 5.3).

In another research effort [29, 30], the authors define an alternative model of ARBAC97 [84] that is more complete and flexible. They define the administrative scope concept that associates each role in the role hierarchy to a set of roles under its control. In a more recent work [37], the concept of administrative scope is further enhanced to support a large distributed system composed of multiple administrative subsystems. Such solutions essentially concern managing the role hierarchy. As for the administration of user-role and permission-role assignments, they do not provide a generic organization-based solution such as the one provided by ARBAC02 [73]. Hence, we can not build on such solutions for managing the mutable structure of IOrg-AutoNets. Actually, we define in ASRBAC the equivalent of the administrative scope in [29, 30, 37], but we still call it role range as in ARBAC02 because it is autonomously specified using a range of regular roles. The role range in ASRBAC is associated with an organizational scope to provide an adaptable specification of the rights of an administrative role to manage regular roles, their hierarchy and also their assignments to nodes and permissions (cf. Section 5.4.5).

In addition to the decentralization and distribution of administration, collaboration is also required in IOrg-AutoNets (cf. Requirement $\mathcal{R}$ 5.1). An interesting existing work [64] addresses distributed administration of RBAC in collaborative environments. This work defines the DARBAC model, according to which each administrative role has an administrative domain, and collaborates with other administrative roles to manage the whole access control system. However, it considers traditional static structures. In ASRBAC, the regular role range associated with an administrative role defines a community or a set of communities as the administrative domain, but also takes the trust level into consideration (cf. Section 5.4.1). This trust consideration is one of the aspects that give ASRBAC the ability to adapt to the mutable structure of an IOrg-AutoNet.

The Administrative Organization Based Access Control (AdOr-BAC) model [32] could be an interesting candidate for managing security in IOrg-AutoNets, whereby SRBAC could be extended to make use of additional organization-based features provided by the Or-BAC model [5]. A recent work on AdOr-BAC [31], which particularly handles aspects of administration confinement constraints and multi-grained administration, meets our expectation concerning the extensibility of this model in terms of self-administration (cf. Section 4.2). However, we consider that our current definition of ASRBAC as an extension

of ARBAC02 already provides a comprehensive support for many Autonomic Computing properties, including self-administration, as we will elaborate in Section 5.5.

The most recent administration solution proposed for RBAC-based access control systems is the Administrative Role and Organization Based Access Control (AROBAC07) model [107]. It particularly concerns decentralized security administration in B2B (Business - to - Business) and B2C (Business - to - Customer) applications. Actually, this solution aims at improving the scalability and performance of the administration of access control systems in a multi-organizational context. It is a flexible and generic solution, which can be enhanced to achieve an autonomic security administration system. However, we currently handle an IOrg-AutoNet as a single organization, which makes ARBAC02 more suitable for our requirements. Nevertheless, in certain IOrg-AutoNet applications, roles and permissions are likely to increase dramatically. In such cases, it would be better to handle each community as an organization, and the whole network as a multi-organizational foundation. Hence, it would be an interesting future work to extend SRBAC with the multi-organizational aspects of the scalable Role and Organization Based Access Control (ROBAC) model [106], and then to use AROBAC07 as a basis for ASRBAC by extending it with a support for Autonomic Computing.

5.3 Contributions

By defining ASRBAC as an extension of ARBAC02 [73], we already can specify distributed access control administration policies as parts of the policies of the access control system itself, and in which the mutable organization structure of IOrg-AutoNets is taken into consideration. See Section 5.2 for the motivations of choosing ARBAC02 as a basis for ASRBAC. We still need to fulfill the other requirements of SRBAC administration (cf. Section 5.1), which correspond to a support of Autonomic Computing. Roughly speaking, the use of autonomic computing mechanisms in access control administration is our contribution with respect to the basis model ARBAC02. More specifically, we may summarize the contributions of ASRBAC as an extension of ARBAC02 in the following list that takes the administration requirements specified in Section 5.1 into account (see Section 5.5 and Figure 5.3 for more details):

1. *Collaboration* (Requirement $\mathcal{R}$ 5.1): The authority nodes collaborate to accomplish administration tasks related to the whole network, in parallel with performing other administration actions specific to their respective administration scopes.

2. *Context-Awareness* (Requirement $\mathcal{R}$ 5.3): An authority node is able to monitor the output of context-aware systems while the network evolves in order to detect systematic and critical changes in the network context concerning the node and object attributes relevant to SRBAC (cf. Sections 4.5.1 and 4.6).

3. *Self-Awareness* (Requirement $\mathcal{R}$ 5.4): An authority node is able to detect changes in the components and policies of SRBAC resulting from any administration operation performed by itself, by another authority node in the network or, in exceptional cases, by an expert user.

4. *Autonomic Computing Properties* (Requirement $\mathcal{R}$ 5.2):
   (a) *Self-Adaptation*:
      i. Network Structure for Nodes (NS-N) adapts to context-aware changes considered as critical.
      ii. Network Structure for Permissions (NS-P) adapts to NS-N changes.
      iii. Regular roles and Regular Role Hierarchy (RRH) adapt to NS-N changes.
      iv. Node-Role Assignments (NRA) adapt to systematic context-aware changes in the set of nodes and/or node attributes, and/or to NS-N changes.
      v. Permission-Role Assignments (PRA) adapt to changes in regular roles and RRH.
   (b) *Self-Optimization*: Low-level permissions are rederived from high-level security objectives and redistributed on the elements of NS-P in response to systematic context-aware changes in the end-user specifications of objects, object attributes and/or security rules.
   (c) *Self-Configuration*: (Autonomic Control Loop)
      i. Changes in regular roles and RRH result in the reconfiguration of administrative roles and the Administrative Role Hierarchy (ARH).
      ii. Changes in NRA result in the reconfiguration of the Administrative Node-Role Assignments (ANRA), which redefines the set of authority nodes.
      iii. Changes in administrative roles and ARH result in the reconfiguration of the Administrative Permission-Role Assignments (APRA).
5. *Policy Negotiation* (Requirement $\mathcal{R}$ 5.5): In the context of their collaborative network administration, and to maintain a consistent low-level security configuration, authority nodes negotiate the changes they may perform on SRBAC policies.

5.4 ASRBAC Definition

ASRBAC is an access control model, whereby the subjects are the authority nodes of the network, the objects are the components of SRBAC and the permissions are the rights to perform administrative actions to eventually adapt the SRBAC components to the network evolution. ASRBAC is an application of SRBAC itself, in which authority nodes acquire administrative permissions through administrative roles, and an Administrative Role Hierarchy (ARH) defines inheritance relationships between administrative roles. Figure 5.1 illustrates the components of ASRBAC and its relation with SRBAC. We can see in this figure that the Administrative Node-Role Assignments (ANRA) and the Administrative Role Hierarchy (ARH) in ASRBAC are derived from the Node-Role Assignments (NRA) and the Regular Role Hierarchy (RRH) in SRBAC respectively. It is this dependency between ASRBAC and SRBAC that will allow us to propose self-adaptation solutions, as will be elaborated later in this chapter.

As introduced in Sections 5.2 and 5.3, ASRBAC is a variant of ARBAC02 [73] extended with autonomic behavior. ARBAC02 defines solutions for the administration of the role

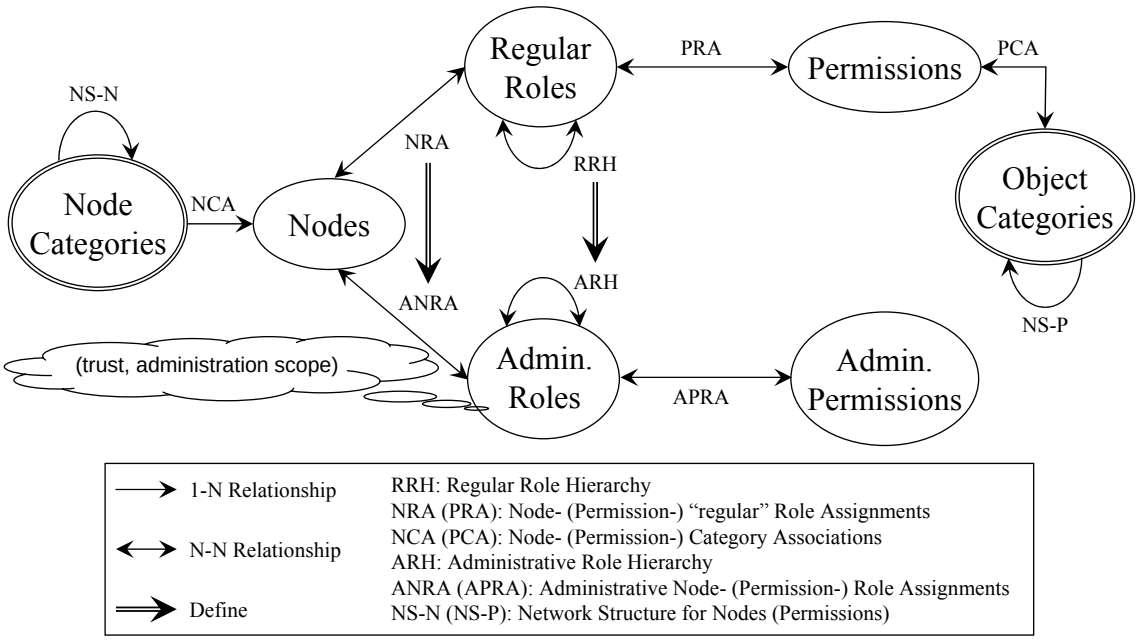


Figure 5.1: ASRBAC as a Model Based on SRBAC

hierarchy and the assignment of roles to users and permissions in RBAC. Similarly, ASRBAC defines solutions for the administration of RRH (Regular Role Hierarchy), NRA (Node-Role Assignments) and PRA (Permission-Role Assignments) in SRBAC. As for managing the organizational components, ARBAC02 actually leaves the administration of the Organization Structure for Users (OS-U) and the Organization Structure for Permissions (OS-P) to the principals of the Human Resources (HR) and Information Technology (IT) respectively. As for ASRBAC, it defines solutions for the administration of NS-N (Network Structure for Nodes) and NS-P (Network Structure for Permissions), which correspond to OS-U and OS-P respectively. ASRBAC makes use of the support of distributed administration in ARBAC02, which is required in infrastructureless environments, and extends it with Autonomic Computing properties, which is specific to IOrg-AutoNets. More details about this extension will be provided in Section 5.5.

A node may be simultaneously assigned to a set of regular roles in the context of SRBAC, and to a set of administrative roles in the context of ASRBAC. More specifically, a regular role assigned to a node and making of it an authority node autonomously imply the assignment of this node to a corresponding administrative role. In other words, the Administrative Node-Role Assignments (ANRA) are self-configured because they are actually a subset of the Node-Role Assignments (NRA) in SRBAC. We can also understand that the administrative roles are actually the subset of regular roles assigned to the authority nodes and having A as the value of the basic role. On the other hand, the Administrative Permission-Role Assignment (APRA) relationships are spontaneously defined according to the correspondence between the elements of an administrative role and the attributes of the SRBAC components under its control.

5.4.1 Administrative Roles

The authority nodes share the administration of SRBAC components in the communities under their respective control. For specifying the administrative roles of an authority node, we do not need to know about its community membership. We are rather interested in the set of communities under its control. We do not need to specify its basic role either, because it is A by definition in an administrative role. However, we still need to know its trust level, because even if a node has administrative privileges, it must not be able to use them in trust levels higher than its own one. Consequently, an administrative role is represented by a trust level and an administration scope.

Definition 5.1 *Let AR be the set of administrative roles and $aRoles$ the function which returns the subset of administrative roles of an authority node:*

- $AR \subseteq \mathcal{T} \times 2^{\mathcal{C}}$
- $aRoles : \mathcal{N} \longrightarrow 2^{AR}$
  - $\forall x \in \mathcal{N}, aRoles(x) = \{(t, as) \mid \exists r \in allnRoles(x), \exists c \in \mathcal{C}, r = (t, c, A, as)\}$

Remark 5.1 *In Definition 5.1:*

- *The function $allnRoles$ returns the set of all the regular roles assigned to a node directly or by inheritance (cf. Definition 4.14).*

5.4.2 Administrative Role Hierarchy

Here we define a domination relationship between administrative roles, which results in an administrative role hierarchy. The total order of trust levels, and the partial order defined on the administration scope, help building that relationship. An administrative role $ar1$ dominates an administrative role $ar2$ if and only if the trust level in $ar1$ is higher than or equal to the trust level in $ar2$ and the administration scope in $ar1$ dominates the administration scope in $ar2$.

Definition 5.2 *Let $ar1$ and $ar2$ be two administrative roles, we write $ar1 \succeq_{AR} ar2$, and we say that $ar1$ dominates $ar2$ in AR:*

- $\forall ar1, ar2 \in AR, ar1 = (t1, as1), ar2 = (t2, as2),$
  $(ar1 \succeq_{AR} ar2) \Leftrightarrow ((t1 \geq t2) \wedge (as1 \succeq as2))$

Notation 5.1 *Starting by Definition 5.2, and for the rest of the thesis:*

- *We use ar, with eventual superscript or subscript indices to denote an administrative role (element of AR).*

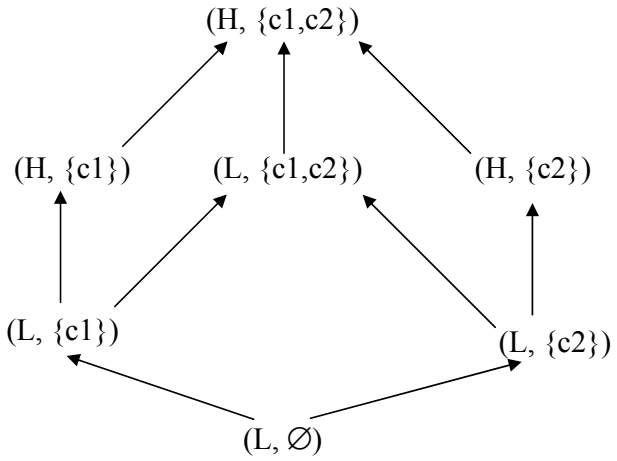


Figure 5.2: Example of an Administrative Role Hierarchy (ARH)

The binary operation $\succeq_{AR}$ defines a partial order over the set of administrative roles. We represent this partial order by a hierarchy, in which an administrative role is senior of the administrative roles it dominates. The *Administrative Role Hierarchy* (ARH) is an inheritance hierarchy, where each administrative role inherits administrative permissions from its junior administrative roles. For instance, Figure 5.2 illustrates the ARH of an IOrg-AutoNet having two communities $c1$ and $c2$, and the initial set of trust levels.

5.4.3 ANRA: Administrative Node-Role Assignments

The function *aRoles* described above in Definition 5.1 also defines the set of Administrative Node-Role Assignments (ANRA). As previously explained, the nodes in this set are the authority nodes of the network and the administrative roles are their regular roles that make of them authority nodes (by incorporating the basic role A). Actually, an authority node is supposed to collaboratively manage the network with other authority nodes for many purposes including access control administration. The administrative roles in ASRBAC particularly concern the capabilities of access control administration assigned to authority nodes.

Definition 5.3 *The set of Administrative Node-Role Assignments (ANRA) is defined as follows:*

- $ANRA \subseteq \mathcal{N} \times AR$
  - $\forall x \in \mathcal{N}, \forall ar \in AR, ((x, ar) \in ANRA) \Leftrightarrow$
    $((ar \in aRoles(x)) \wedge (\forall ar' \in AR, ar' \neq ar, ar' \succeq_{AR} ar, ar' \notin aRoles(x)))$

Remark 5.2 *In Definition 5.3:*

- *The set $ANRA$ associates an authority node with the set of administrative roles directly assigned to it without using the inheritance relationship in ARH. This provides a scalable representation of administrative node-role assignment.*

Table 5.1: APRA in ASRBAC

Administrative Role	Organizational Scope	Hierarchical Scope
$(t, as) \in AR$	$\{(t, c, nil_{\mathcal{K}}) \mid c \in as\} \cup \{(t, c, nil_{\mathcal{B}}) \mid c \in as\}$	$\{[(L, c, NA, \emptyset), (t, c, A, \{c\})[\mid c \in as\}$
AR	$\{(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}), (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}})\}$	$](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[$

Variables		Constants	
t:	Trust Level	L:	Lowest-trust level
as:	Administration Scope	H:	Highest-trust level
c:	Community	NA:	non-administrative basic role
		A:	Authority basic role
		$\mathcal{N}$:	The set of network nodes
		$\mathcal{C}$:	The set of communities
		$\mathcal{T}$:	The set of trust levels
		$nil_{\mathcal{T}}$:	Trust levels are not considered
		$nil_{\mathcal{K}}$:	Capability classes are not considered
		$nil_{\mathcal{B}}$:	Basic roles are not considered
		AR:	The set of administrative roles

5.4.4 APRA: Administrative Permission-Role Assignments

An administrative role defines a set of role ranges in RRH, a set of branches in NS-N and a set of branches in NS-P over which it may imply administrative privileges. Hence, an authority node can manage a certain set of regular roles and their assignments to each other. Besides, it manages their assignments to nodes and permissions selected from certain node an object categories. Moreover, it can manage those categories and their assignments to each other in their respective structures. The role ranges and the branches of the organizational structures altogether define the domain of administration of an authority node. In certain cases, the target of an administrative action can not be managed by a single authority node. Therefore, the set of authority nodes may form together an administration board that has the whole RRH, NS-N and NS-P structures as an administration domain.

So in order to define the set of Administrative Permission-Role Assignments (APRA), we need to specify for an administrative role, the administration domain in form of a set of node/object categories and a set of regular role ranges. Besides, we need to add a specification entry that represents the collaboration between authority nodes to manage the SRBAC components that a single authority node cannot manage alone. Table 5.1 defines the set of Administrative Permission-Role Assignments (APRA) in an ASRBAC-based access control administration system.

As elaborated in Section 5.5.5 and depicted in Figure 5.3, APRA is self-configured with respect to the configuration of the administrative roles and their hierarchy ARH. The first line in Table 5.1 states that an administrative role $((t, as), t \in \mathcal{T}, as \in 2^{\mathcal{C}})$ defines its administration domain with respect to its attributes, namely the trust level (t) and the

administration scope (as). An administration domain is a combination of an organizational scope and a hierarchical scope. An organizational scope is a set of branches in NS-N and NS-P represented by the set of the root node categories and the root object categories respectively. A hierarchical structure is a set of regular role ranges in the Regular Role Hierarchy (RRH).

More specifically, for an administrative role ($(t, as) \in AR$), the trust level (t) and each one of the communities in the administrative scope (as) define together a branch of NS-N/NS-P and a regular role range. The branch of NS-N is represented by a root composed of the trust level and the community and ignoring the capability class ($(t, c, nil_{\mathcal{K}}), c \in as$). The branch of NS-P is represented by a root composed of the trust level and the community and ignoring the basic role ($(t, c, nil_{\mathcal{B}}), c \in as$). The regular role range begins by, and includes, the regular role corresponding to the non-administrative nodes of the lowest trust level in the community, and ends by, but does not include, the regular role corresponding to the authority node that has the same trust level of the administrative role and can manage the community ($[(L, c, NA, \emptyset), (t, c, A, \{c\})[, c \in as$).

A regular role range in the hierarchical scope of an administrative role is open at the upper bound side because an authority node must not be able to control its own security administration privileges. Therefore, there is at least one regular role assigned to an authority node and out of the administration domains of all the administrative roles. This is one of the reasons for which the authority nodes may need to work together as an administration board in certain cases.

The second line in Table 5.1 allows authority nodes to collaborate to manage certain SRBAC components when needed. This generic APRA entry states that the set of all the administrative roles (AR) has for administration domain a combination between the whole NS-N and NS-P trees represented by the tree roots ($\{(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}), (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}})\}$), and the whole RRH hierarchy represented by an open regular role range that has for bounds the delimiters of RRH ($](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[$), which are abstract regular roles used for specification purposes only.

5.4.5 ASRBAC Policies

As illustrated in Figure 5.1, administrative permissions are not associated with the object categories of NS-P. The targets of administrative actions are the SRBAC components, including the object categories and the NS-P themselves. Those administrative targets are also categorized, but according to the specifications of the relevant administrative roles (cf. Section 5.4.4).

ASRBAC policies can be expressed using the extended version of the RBAC Profile of XACML V2.0 [15] that we propose for the specification of SRBAC policies (cf. Section 4.9), as long as ASRBAC is a variant of SRBAC itself (cf. Figure 5.1). Actually, ASRBAC policy specification is a part of the SRBAC policy specification itself, which is one of the self-management features of ASRBAC. The `Role <PolicySet>` (RPS) that defines and assigns an authority regular role to a node, implicitly defines and assigns the corresponding administrative role to that node. According to Figure 4.6, the RPS is then associated with a `Permission <PolicySet>` (PPS), which in its turn is associated on one

hand to one or more other PPS instances to express the inheritance of permissions from junior regular roles, and on the other hand to one or more `ObjectCategory <PolicySet>` (OCPS) instances where the actual permissions are specified.

However, administrative permissions are not associated with object categories, as illustrated in Figure 5.1. So the PPS of an authority regular role will directly include the permissions related to the administration of SRBAC components. These administrative permissions are actually specified in a rule that defines an action called `manage`, and a set of resources representing the organizational scope and the hierarchical scope of the administrative role. Appendix A gives in Section A.2 an example of the XACML entity corresponding to the PPS of an authority regular role in a Home Network.

As for the collaborative management accomplished by all the authority nodes, which is represented in the second line of Table 5.1, a specific RPS uses the "basic role" attribute and its value A to state that all the authority nodes can acquire the collaborative administrative permissions specified in an associated PPS when they work together on an administrative task. This specific SPS instance only includes a rule that defines the action `manage`, and two resources representing respectively the whole NS-N/NS-P trees as organizational scope and the whole RRH as hierarchical scope.

The XACML action `manage` used for the specification of ASRBAC policies in the `Permission <PolicySet>` (PPS) instances of the authority regular roles is implemented using the following five ASRBAC predicates, which have as parameters an administrative role ar, a root nc of a branch in NS-N and a root oc of a branch in NS-P belonging to the organizational scope of ar, and a regular role range $[r1, r2[$ belonging to the hierarchical scope of ar:

1. $canAssign(ar, @nc, [r1, r2[)$: ar can select a node from the node category nc, or any of its junior node categories, to assign it to a regular role in $[r1, r2[$.

2. $canAssignP(ar, @oc, [r1, r2[)$: ar can select a permission associated with the object category oc, or any of its senior object categories, to assign it to a regular role in $[r1, r2[$.

3. $canModify(ar, [r1, r2[)$: ar can revoke users or permissions assigned to a regular role in $[r1, r2[$, and it can add or remove regular roles and role-role inheritance associations in $[r1, r2[$.

4. $canModifyNC(ar, nc)$: ar can add or remove node categories in the branch of NS-N defined by the root nc.

5. $canModifyOC(ar, oc)$: ar can rederive and redistribute the permissions associated with the object categories of the branch of NS-P defined by the root oc, and add or remove object categories in that latter.

Definition 5.4 *Let $(t, as) \in AR$ be an administrative role in the access control administration system of an IOrg-AutoNet, we define for it the following five sets of administrative actions based on ASRBAC predicates:*

1. $\{canAssign((t, as), @(t, c, nil_{\mathcal{K}}), [(L, c, NA, \emptyset), (t, c, A, \{c\})[]) \mid c \in as\}$
2. $\{canAssignP((t, as), @(t, c, nil_{\mathcal{B}}), [(L, c, NA, \emptyset), (t, c, A, \{c\})[]) \mid c \in as\}$
3. $\{canModify((t, as), [(L, c, NA, \emptyset), (t, c, A, \{c\})[]) \mid c \in as\}$
4. $\{canModifyNC((t, as), (t, c, nil_{\mathcal{K}})) \mid c \in as\}$
5. $\{canModifyOC((t, as), (t, c, nil_{\mathcal{B}})) \mid c \in as\}$

Definition 5.5 *We define for the set of all the administrative roles the following five collaborative administrative actions based on ASRBAC predicates:*

1. $canAssign(AR, @(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}),](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[])$
2. $canAssignP(AR, @(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}}),](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[])$
3. $canModify(AR,](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[])$
4. $canModifyNC(AR, (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}))$
5. $canModifyOC(AR, (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}}))$

Table 5.1 can be implemented in any IOrg-AutoNet to express all the possible Administrative Permission-Role Assignments (APRA) and to integrate the related administrative policies in the access control policies of the network using the extended version of the RBAC Profile of XACML V2.0 proposed for SRBAC. Definitions 5.4 and 5.5 can be applied in any IOrg-AutoNet to perform all the possible administrative actions of ASRBAC. As elaborated in this section, the authority nodes can perform APRA configuration, ASRBAC policy specification and administrative actions without any human intervention. Actually, a very important contribution of ASRBAC is that its configuration and enforcement are completely self-managed.

5.4.6 Formal Definition of the ASRBAC Model

On one hand, after describing the different components of the SRBAC model, giving formal definitions for certain ones and their related functions, and finally giving a formal definition of SRBAC at the end of Section 4.8 (cf. Definition 4.17). On the other hand, after describing and giving formal definitions for the different components of the SRBAC model and their related functions, we may conclude this section by the following formal definition of the ASRBAC model:

Definition 5.6 *ASRBAC has the following components:*

- *All the components of SRBAC as stated in Definition 4.17.*
- *AR: the set of administrative roles, where $RR \Rightarrow AR$ (cf. Definition 5.1).*
- *ARH: the partial order relation on AR called administrative role hierarchy (cf. Definition 5.2).*

- *ANRA: the set of administrative node role assignments, where $NRA \Rightarrow ANRA$ (cf. Definition 5.3).*
- *APRA: the set of administrative permission role assignments (cf. Table 5.1).*
- *AP: the set of administrative permissions, which are administrative actions on SRBAC components (cf. Definitions 5.4 and 5.5).*

5.5 Autonomic Computing Support in ASRBAC

An administrative action in ASRBAC is either based on a collaborative decision taken by the authority nodes, or on an autonomous mapping between access control components. The authority nodes may adapt NS-N, and in certain cases NRA and NS-P, to the network evolution according to decisions taken after analyzing certain detected changes in the network context. This is the case of decision-based administrative actions. RRH and PRA, and in certain cases NRA and NS-P, must be adapted to the evolution of other components of SRBAC for mapping reasons. The authority nodes accomplish such adaptations in the context of autonomous mapping-based administrative actions. For instance, changes in the availability conditions of the network nodes, which can be detected and analyzed using a dedicated environment variable, imply modification decisions for NS-N because the community composition must change. A modification in NS-N implies in its turn autonomous changes to NS-P to map to NS-N changes.

We describe in this section the support of Autonomic Computing in the ASRBAC model by exploring the different types of administrative actions and other related aspects. We explain how an ASRBAC-based system detects different kinds of changes in the network context while the network evolves, and how it responds to the detected changes. Figure 5.3 illustrates the Autonomic Computing functionality features in ASRBAC.

5.5.1 Context-Aware Information

The authority nodes continuously evaluate those variables, which allows them to monitor the network and its context in order to detect and analyze evolution events (cf. Section 3.2). Table 5.2 describes the environment variables that we consider in our work, with respect to the evolution of the structure (cf. Section 3.2), to changes in high-level security configuration or to the modification of administrative responsibilities of certain nodes. We should note that an evolution at the community level may imply a set of evolution events at the node-level (cf. Definition 3.4), and that network merging or splitting may imply a set of evolution events at the community-level and the node-level (cf. Definition 3.7). We should also note that an evolution of the structure may cause changes to the basic roles of certain nodes, as illustrated in Figure 3.2.

The trust level of a node changes when it wins or loses in terms of its trustworthiness in the network. This is a systematic evolution, to which the relevant authority node should respond by changing the set of regular roles of that node. In certain cases, a new element may be used from the predefined set of trust levels in the network. For instance, if a new

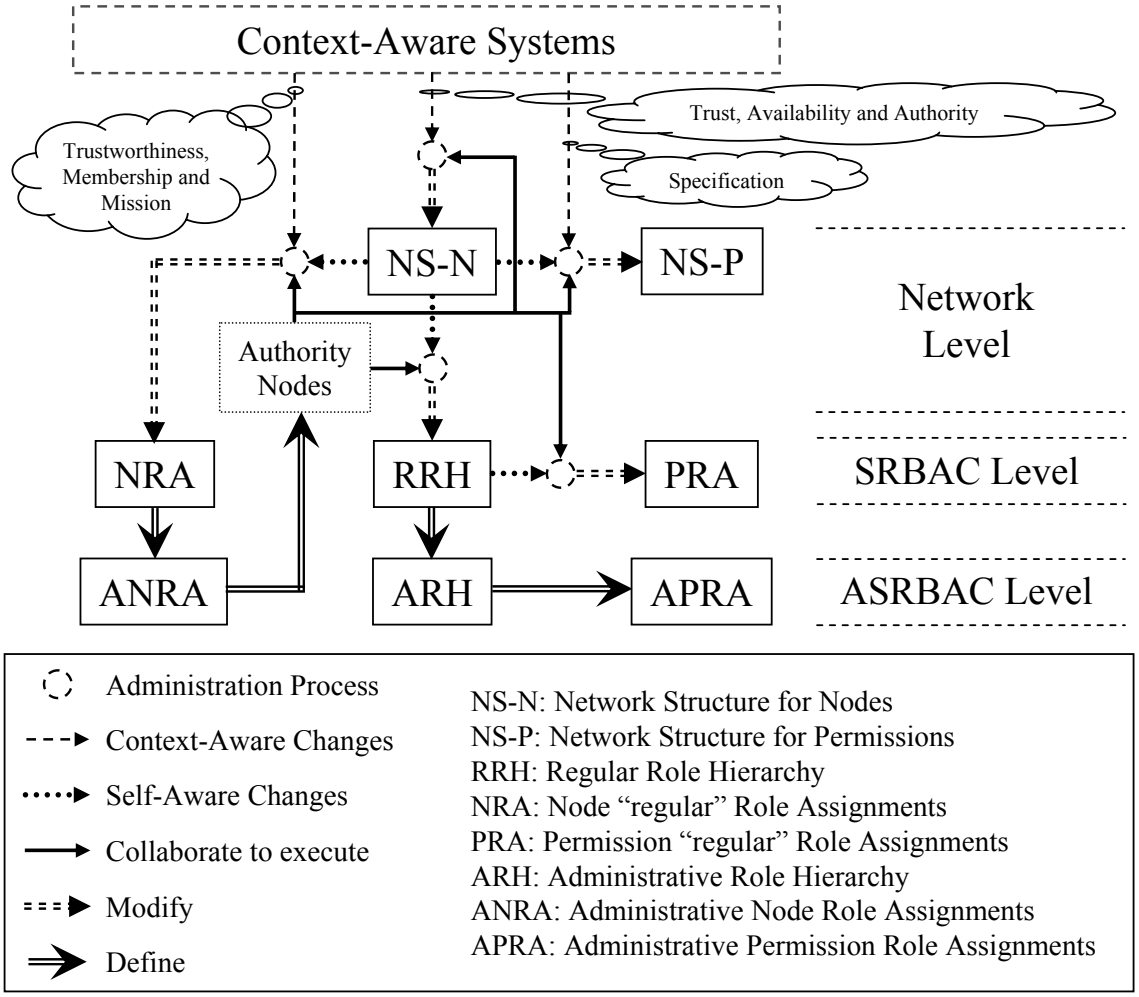


Figure 5.3: Support of Autonomic Computing in ASRBAC

Table 5.2: Environment Variables

Environment Variable	Evolution Event
Trust	Trust level used for a first time
Availability	Evolution takes place at community-level
Authority	Basic role (capability class) used for a first time
Specification	High-level security specifications change
Trustworthiness	Trust level changes for one or more nodes
Membership	Evolution takes place at node-level
Mission	Basic role changes for one or more nodes

node joins the network, it is usually assigned to the trust level L. However, there might be already nodes assigned to this lowest trust level, and the reputation system decides that their trustworthiness should be higher than the trustworthiness of the new node. It eventually uses a new trust level for those nodes and keeps the new node only assigned to L. This is a critical evolution, to which all the authority nodes should eventually respond by updating the Network Structure for Nodes (NS-N). A reputation system (cf. Assumption 3.1) may output the environment variables *Trustworthiness* and *Trust* describing changes in node trustworthiness and in the set of the currently assigned trust levels respectively.

A node may win or lose its membership in the network either individually (cf. Definition 3.3) or in the context of an evolution at the community-level (cf. Definition 3.4) or the network level (see network merging and splitting in Definition 3.7). Such changes in the membership of a node may cause changes in the basic role of the same node or other nodes (cf. Figure 3.2). For example, when the authority of a community is removed or banished, another node should acquire the authority basic role (A) to replace it. In these cases of systematic change, the relevant authority nodes should modify the set of Node-Role Assignments (NRA). A resource management system (cf. Assumption 3.3) may output the environment variables *Membership* and *Mission* describing changes in nodes' memberships and basic roles respectively.

The network evolution events may result in critical changes in the components of the autonomic access control system. For example, when a community is integrated or revoked, or when a basic role is used for a first time, such as using the basic role DA (Delegated Authority) in a first subnet exportation, all the authority nodes should respond by updating the NS-N (Network Structure for Nodes) component, which eventually modifies all the components of SRBAC and ASRBAC as illustrated in Figure 5.3. The resource management system may output the environment variables *Availability* and *Authority* describing changes in the community composition and in the set of the currently assigned basic roles respectively.

As introduced in Section 5.3, permissions are rederived from high-level security specifications and redistributed on object categories in the Network Structure for Permissions (NS-P), in response to certain detected changes in the network. Such policy reconfiguration may be implied by changes in NS-N, which are changes in NS-P as well by mapping. In other words, low-level policies may adapt to changes in the values of the environment variables *Trust*, *Availability* and *Authority*. Besides, if the endusers modify their high-level security configuration, the NS-P will be also updated in terms of associations between permissions and object categories. Such changes are critical, to which all the authority nodes should eventually respond by negotiating and performing modifications to the SRBAC access rules. The Security Policy System of the IOrg-AutoNet (cf. Section 2.2.4) is also a context-aware system that may output the environment variable *Specification* describing changes in the high-level security configuration of the network.

5.5.2 Predefined Self-Management

Certain decision-based administrative actions in ASRBAC follow a systematic predefined self-management functionality. Such administrative actions occur in response to expected

changes in the context of the network. The environment variables reporting expected changes are considered as non-critical because they do not lead to changes in the policies of the autonomic administration system itself [50]. Therefore, in ASRBAC, the non-critical changes are the ones that will not result in changing the set of Administrative Permission-Role Assignments (APRA). As depicted in Figure 5.3, we have two types of non-critical context-aware changes that do not lead to the modification of the ASRBAC policies:

1. Node-Role Assignments (NRA) adapt to systematic context-aware changes in the set of nodes and/or node attributes. Section 5.5.1 explains that such NRA self-adaptation is performed in response to detected changes in the environment variables *Trustworthiness*, *Membership* and *Mission*.

2. Low-level permissions are rederived from high-level security specifications and redistributed on the elements of the Network Structure for Permissions (NS-P) in response to systematic context-aware changes in the set of objects and/or object attributes, or in the set of high-level security rules. Section 5.5.1 explains that such permission self-optimization is performed in response to detected changes in the environment variable *Specification*.

Example 5.1 *An authority node x may leave the network because of a node removal or banishment. The environment variable* Membership *reports this event to the other authority nodes of the network. If one of them has an administrative role senior to $aRoles(x)$ it deletes the set $\{(x, r) \mid r \in RR, (x, r) \in NRA\}$. Otherwise, the authority nodes negotiate and agree on this deletion. Besides, one or more communities would have no authority node as a result, which is detected by the relevant authority nodes through the environment variable* Mission. *In response to this event, those authority nodes negotiate to select a qualified node to replace the lost authority node.*

5.5.3 Autonomic Control Loop

An autonomic system should be able to cope with critical changes in its environment by adapting itself [56]. Such functionality is achieved through autonomic control loops [41], whereby the execution of the autonomic actions result in a feedback that may change the administration policies of the autonomic system itself. Critical changes are detected through monitoring critical environment variables [50], such as *Trust*, *Availability* and *Authority* as explained in Section 5.5.1. Such changes make the authority nodes decide to modify SRBAC components, which is basically the goal of administrative actions, but they will also make ASRBAC policies adapt accordingly. Figure 5.3 illustrates the autonomic control loop of ASRBAC. We can see in this figure that the Network Structure for Nodes (NS-N) adapts to critical context-aware changes, and that NS-N self-adaptation eventually changes the set of Permission-Role Assignments (PRA) and the set of Administrative Permission-Role Assignments (APRA). In other words, the access control administration system of an IOrg-AutoNet is able to detect critical changes in the *Trust*, *Availability* and/or *Authority* conditions in the network, and to adapt the access control policies and its own policies accordingly.

Example 5.2 *A reputation system (cf. Assumption 3.1) may decide that a trust level t must be used for a first time. The authority nodes capture this critical decision through detection and analysis of changes in the environment variable* Trust. *The following modifications then take place (see figure 5.3):*

1. ***Decision-based update of NS-N:*** *Creation of the new node category* $(t, \mathcal{N}, nil_{\mathcal{K}})$ *and the set of its junior node categories. Certain nodes are then removed from their respective node categories to be added to the new ones because they should be assigned to the new trust level. NS-N will be then updated by integrating the new node categories.*

2. ***Mapping of NS-P:*** *The new node categories imply the creation of the new object category* $(t, \mathcal{N}, nil_{\mathcal{B}})$*, and its senior object categories. Low-level SRBAC permissions will be rederived from high-level security specifications and redistributed on object categories. NS-P will be then updated by integrating the new object categories.*

3. ***Mapping of RRH:*** *For each element in the new set of node categories* $\{(t, c, k) \in NC\}$*, if* $k = LD$ *then the regular role* $(t, c, NA, \emptyset)$ *is created, otherwise the set of regular roles* $\{(t, c, b, as) \mid bAbility(b) = k, as \in 2^{\mathcal{C}} \setminus \{\emptyset\}\}$ *is created. RRH will be then updated by integrating the new regular roles.*

4. ***Spontaneous update of ARH:*** *The set of regular roles* $\{(t, c, b, as) \mid c \in \mathcal{C}, b = A, as \in 2^{\mathcal{C}} \setminus \{\emptyset\}\}$*, which indicates a potential designation of new authority nodes in the network, causes the new set of administrative roles* $\{(t, as) \mid as \in 2^{\mathcal{C}} \setminus \{\emptyset\}\}$ *to be created and integrated in ARH.*

5. ***Spontaneous update of APRA*** *Each new administrative role will be assigned to the corresponding set of administrative permissions, which represents an update of the ASRBAC policies through the modification of APRA (cf. Section 5.4.4).*

5.5.4 Mapping-Based Self-Adaptation

In addition to systematic and critical context-aware decision-based administrative actions, ASRBAC defines autonomous self-aware mapping-based actions. An authority node is able to detect changes in the components and policies of SRBAC resulting from any administration operation performed by itself, by another authority node in the network or, in exceptional cases, by an expert user. Figure 5.3 illustrates the following mapping-based self-adaptation actions accomplished by authority nodes in response to detected changes in self-aware data:

1. The Network Structure for Permissions (NS-P) adapts to the changes of the Network Structure for Nodes (NS-N). Actually, NS-P elements are respectively equivalent to NS-N elements, and the NS-P tree structure is the inverse of the NS-N tree structure, taking into account the correspondence between basic roles and capability classes (cf. Sections 4.7.1 and 4.7.2). As a result of adapting NS-P to NS-N, the low-level permissions associated with the different object categories in NS-P will be

redistributed on those categories after being rederived from the high-level security specifications of the network. Example 5.2 explains this kind of self-adaptation.

2. The regular roles and the Regular Role Hierarchy (RRH) adapt to NS-N changes. One or more regular roles can be specified with regard to the attributes of a node category. Example 5.2 explains this kind of self-adaptation.

3. Node-Role Assignments (NRA) adapt to NS-N changes. A change in NS-N means that certain nodes will change their respective categories. This actually means that the *Trustworthiness*, *Membership* and/or *Mission* of such nodes could have been changed, which implies the need for updating NRA.

4. Permission-Role Assignments (PRA) adapt to changes in regular roles and RRH. The assignment of permissions to a regular role in SRBAC is actually an association between that regular role and a corresponding set of object categories (cf. Definitions 4.15 and 4.16). Because this association is based on the specification of the regular role, a self-aware change of PRA takes place in response to changes in the specification of the set of regular roles.

5.5.5 Autonomous Evolution

As an autonomic computing system, an ASRBAC-based access control administration system adapts to its environment and to its own actions without any, or with a minimum of, human intervention [51]. ASRBAC components are completely self-configured with respect to the changes applied by the ASRBAC agents themselves (authority nodes) to SRBAC components and policies, or with respect to changes in other ASRBAC components. Figure 5.3 illustrates the following self-configuration actions accomplished in terms of an autonomous evolution of ASRBAC:

1. Changes in regular roles and RRH result in the reconfiguration of administrative roles and the Administrative Role Hierarchy (ARH). This actually takes place when the set of regular roles based on the authority basic role A is modified. Usually, changes to the *Trust* and *Availability* conditions in the network may imply such modifications. According to Definitions 5.1 and 5.2 respectively, the set of administrative roles of a node is a subset of the set of its regular roles based on the A basic role, and the ARH hierarchy is based on the RRH hierarchy.

2. Changes in NRA result in the reconfiguration of the Administrative Node-Role Assignments (ANRA). This actually takes place in case of acquisition or dispossession of the authority role. More specifically, this is the case of a change in NRA in response to a change in the environment variable *Mission* captured from the output of the resource management system as a context-aware information, or as a self-aware information from the output of an operation of NS-N adaptation. As a result, the set of authority nodes (ASRBAC agents) changes as well, which can be also considered as a kind of self-adaptation in ASRBAC.

3. Changes in administrative roles and ARH result in the reconfiguration of the Administrative Permission-Role Assignments (APRA). Section 5.4.4 explains in details how the specification of an administrative role autonomously determines its administrative permissions. Reconfiguring APRA with respect to changes in ARH is actually the respecification of the administration policies of ASRBAC. Because this reconfiguration comes at the end of an autonomic process that begins by a critical context-aware adaptation of NS-N, as illustrated in Figure 5.3, it defines the autonomic control loop of an ASRBAC-based system (cf. Section 5.5.3).

5.5.6 Node Cooperation

The authority nodes cooperate when they use their administrative roles to perform different kinds of administrative actions. Figure 5.3 depicts that the authority nodes collaborate to execute the different context-aware and self-aware autonomic operations on SRBAC components. Actually, an authority node is able to perform most of such operations independently in its administrative domain, which is characterized by a set of ranges of regular roles and a set of node/object categories (cf. Section 5.4.4). However, the authority nodes may need to eventually agree on certain modifications in the context of certain autonomic operations for one of the following two reasons:

1. The target of an administrative operation may be out of the administration domain of any authority node. For example, suppose that an authority node having the regular role $(H, c1, A, \{c1\})$ must be removed from a network where $\mathcal{C} = \{c1, c2\}$ and $c2$ has its own authority. The authority node of $c1$ cannot remove itself, and the authority node of $c2$ can not remove it either. The regular role $(H, c1, A, \{c1\})$, which determines the authority node of $c1$, is not in the administration domain of any of the two authority nodes of the network. In this case, they need to collaborate to accomplish the node removal task. Section 5.4.5 explains in more details how the authority nodes can act as a board of administrators.

2. The administrative operation is based on a decision taken after detecting and analyzing certain context-aware information. This is the case of self-adaptation of NS-N, and certain cases of self-adaptation of NRA and NS-P, as illustrated in Figure 5.3. The authority nodes need to agree on the adaptation decisions before executing them because they may need to:

   (a) validate a unique configuration of the NS-N component.

   (b) elect a qualified node for assignment to a regular role.

   (c) negotiate permissions' optimization to maintain a consistent SRBAC policy specification in the network. We give a particular concern to policy negotiation, and elaborate corresponding mechanisms in Section 5.6. As for the other cases of node cooperation, they will make part of a future work.

5.6 Policy Negotiation Mechanisms

As illustrated in Figure 5.3, the Network Structure for Permissions (NS-P) may be modified with respect to context-aware and/or self-aware changes. A change in NS-P means that the authority nodes must rederive the SRBAC permissions from the high-level security specifications and redistribute them on the object categories. This SRBAC policy optimization may be a response to the detection of context-aware information indicating changes in the high-level security specifications, as explained in Section 5.5.2. It could also be a second step after adapting the NS-P structure in response to self-aware changes in the Network Structure for Nodes (NS-N), as explained in Section 5.5.4.

More concretely, authority nodes should modify the `ObjectCategory <PolicySet>` (OCPS) instances in the XACML specification of the SRBAC policies (cf. Section 4.9) when the high-level security configuration changes and/or when the network structure mutates. As explained in Section 5.5.6, this should be a collaborative administration task to maintain a consistent SRBAC policy specification in the network. The XACML specification language [4] does not support collaboration between security administrators. Therefore, each authority node will have to modify the OCPS instances independently, and then cooperate with the other authority nodes to unify their modifications.

Roughly speaking, all the authority nodes use the same modules of permission derivation and distribution of the Security Policy System (SPS) to translate and enforce the high-level security configuration (cf. Section 2.2.4). However, in order to be realistic, the access control administration system should consider the possibility of having differences between the authority nodes in the results of permission derivation and distribution. For example, in a mobile IOrg-AutoNet, the topology may be variable enough to make authority nodes unable to capture the same context-aware information, which may make them execute different changes on the same access rules. Therefore, the authority nodes should negotiate the resulting access control rules in order to unify their OCPS instances in case of a conflict. This collaboration is also necessary when the modifications concern policy parts that are out of the administration domain of a single authority regular role (cf. Section 5.4.4). Furthermore, this type of node cooperation is clearly justified when a network evolution implies the merging of two communities, the merging of two whole networks or the reintegration of an exported subnet (cf. Section 3.2).

By applying a concept of similarity computation on XACML rules, such as the one proposed in [66], we propose in this section a policy negotiation algorithm, which takes into account all the decisions of all the authority nodes, but gives the most trusted ones the privilege to take the final decisions. We should note here that we only refer to the rule similarity computation concepts proposed by the authors of [66] but not their algorithm of policy integration. In their approach, each party keeps its own policies, and the goal is to take a policy integration decision at access time in case of conflicts. In our approach, all the involved parties (authority nodes) should eventually unify their policies.

Algorithm 5.1 *We propose the following policy negotiation algorithm for unifying SRBAC permissions in the context of an adaptation of NS-P or an optimization of access control rules:*

1. PHASE 1*: Preliminary local policy modifications per authority node:*

   (a) *Each couple of authority nodes compute the similarity of their XACML rules. Each authority node will then be aware of the differences in rule specifications with respect to all the other authority nodes.*

   (b) *Each authority node takes its own decisions to modify the rules for which differences are detected.*

2. PHASE 2*: Intermediate unified policy modifications per trust level:*

   (a) *Each couple of authority nodes belonging to the same trust level compute the similarity of the modified XACML rules. Each authority node will then be aware of the decisions of all the other authority nodes belonging to the same trust level.*

   (b) *Each authority node takes its own decisions to modify the rules for which differences in decisions with other authority nodes of the same trust level are detected.*

   (c) *The previous two steps are repeated on each trust level until no more differences in rule modifications are detected on that trust level.*

   (d) *An authority node of each trust level, except the highest trust level, is elected to multicast the modification results to the set of authority nodes belonging to the highest trust level.*

3. PHASE 3*: Final global unified policy modifications:*

   (a) *Each authority node of the highest trust level takes its own decisions to remodify the rules taking into account the modification decisions reported from each lower trust level.*

   (b) *Each couple of authority nodes belonging to the highest trust level compute the similarity of the remodified XACML rules. Each authority node of the highest trust level will then be aware of the decisions of all the other authority nodes belonging to the highest trust level.*

   (c) *Each authority node of the highest trust level takes its own decisions to modify the rules for which differences with other authority nodes of the highest trust level are detected.*

   (d) *The last two steps are repeated until no more differences in rule modifications are detected on the highest trust level.*

   (e) *An authority node of the highest trust level is elected to multicast the final global modification results to the authority nodes of all the lower trust levels.*

The policy negotiation algorithm that we propose in this section (Algorithm 5.1) is illustrated in Figure 5.4. As illustrated in this figure, Algorithm 5.1 is based on the following set of policy negotiation mechanisms:

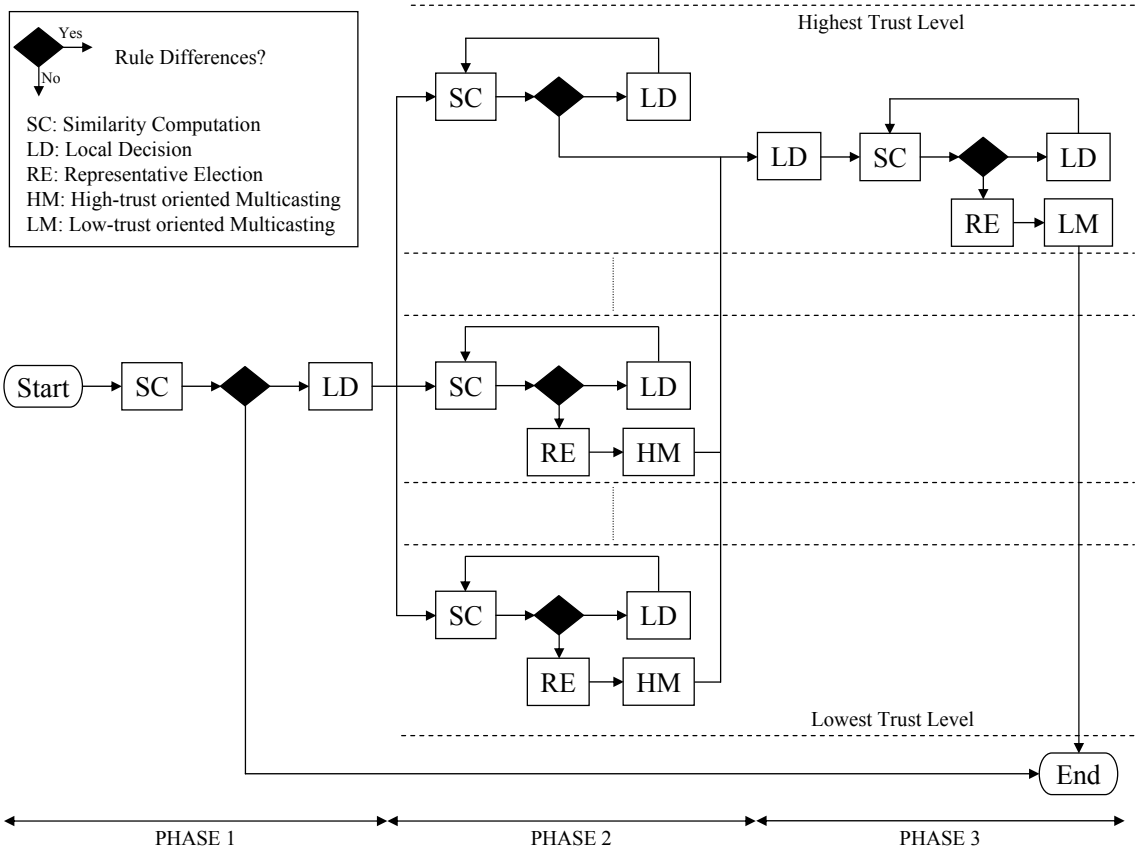
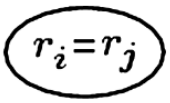
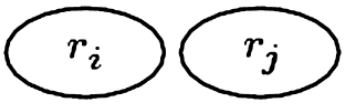
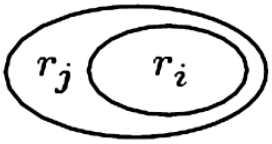
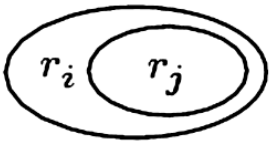
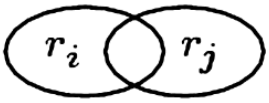


Figure 5.4: Policy Negotiation Algorithm for ASRBAC

Rule similarity type	*Authorized requests*
r_i *Converges* r_j	
r_i *Diverges* r_j	
r_i *Restricts* r_j	
r_i *Extends* r_j	
r_i *Shuffles* r_j	

r_i, r_j: XACML rules of different policies applied on the same resource

Figure 5.5: Rule Similarity Types (Extracted from [66])

- **SC (Similarity Computation):** The XACML format is structured enough to extract the details of an access control rule from a policy and compare them with the details of another rule of another policy applied on the same resource. Therefore, it is possible to define a format for the similarity between two XACML rules and then to design an algorithm to compute rule similarity. For example, the authors of [66] define a format for rule similarity and an algorithm that decides if a rule converges, diverges, restricts, extends or shuffles another rule. Figure 5.5, which is extracted from [66], illustrates those rule similarity types.
- **LD (Local Decision):** taken for rule modification.
- **RE (Representative Election):** of an authority node in a trust level.
- **HM (High-trust oriented Multicasting):** of modified XACML rules.
- **LM (Low-trust oriented Multicasting):** of modified XACML rules.

Algorithm 5.1 is actually a groundwork for a future enhancement of the access control

administration system of IOrg-AutoNets. We only intend in this section to describe the basic ideas we currently may propose in terms of a platform for a future policy negotiation protocol. In a future work, the policy negotiation mechanisms of Algorithm 5.1 should be elaborated and should take the security concerns into account, and the policy negotiation protocol should be specified, formalized and validated.

5.7 Conclusion

This chapter presented our main contribution as a step **Toward a Security Administration System for Autonomic Networks**. It is an administration model defined in the context of an autonomic access control system for a specific model of autonomic networks that we call IOrg-AutoNets (cf. Definition 3.2). Our administration model is called ASRBAC (Administrative SRBAC). It is the administrative counterpart of the SRBAC (Secure Relation Based Access Control) model, which we defined as the access control model of IOrg-AutoNets in Chapter 4. ASRBAC is a variant of SRBAC itself, and its administration policies are incorporated in the access control policies of SRBAC. This dependency between ASRBAC and SRBAC is the key feature behind the support of Autonomic Computing in ASRBAC.

We explained the access control administration requirements in IOrg-AutoNets in order to motivate the different features of ASRBAC. We can summarize those requirements by the need for context-awareness, self-awareness and node cooperation in order to achieve a distributed, collaborative, self-managing, self-adaptable, access control administration solution. Afterward, we could discuss a number of existing solutions to see if they fulfill such requirements, whereby we emphasized the choice of ARBAC02 [73] as a basis for ASRBAC. That choice was mainly based on the dependency between the administrative model ARBAC02 and the corresponding access control model RBAC [89], given that RBAC is the basis of SRBAC. It was then possible to describe the contributions of ASRBAC with respect to ARBAC02. Roughly speaking, ASRBAC extends ARBAC02 with a support for Autonomic Computing.

After introducing the requirements, motivations and contributions of ASRBAC, it was time to define it. We gave a formal definition of ASRBAC (Definition 5.6) describing its components. A formal definition for each administrative component was then provided. Definition 5.1 describes the administrative roles and explains their autonomous relationship with the regular roles of SRBAC. Definition 5.2 describes the administrative role hierarchy. Definition 5.3 describes the administrative node-role assignments and explains their autonomous relationship with the node-role assignments of SRBAC. Table 5.1 describes the administrative permission-role assignments and explains how the specification of an administrative role autonomously defines its administration scope based on organizational and hierarchical components of SRBAC. The administration scope of an administrative role identifies then its administrative permissions as a set of administrative actions on those SRBAC components. Definitions 5.4 and 5.5 describe those administrative actions for individual and collaborative administration respectively.

A very important section in this chapter (Section 5.5) elaborated then the Autonomic

Computing support in ASRBAC. Figure 5.3, which mainly illustrates the autonomic control loop of ASRBAC starting by detecting context-aware evolution events and ending by adapting all SRBAC and ASRBAC components, and Table 5.2, which describes the environment variables detected by the autonomous agents of ASRBAC (authority nodes) and the related evolution events, were the key elements of discussion in that section. Different types of autonomic administration operations were discussed, namely the predefined self-management, the autonomic control loop operations, the mapping-based self-adaptation and the autonomous evolution. The section ended by motivating node cooperation in IOrg-AutoNets in the context of access control administration, before dedicating a separate other section for describing a sketch of a policy negotiation algorithm as a future solution for node cooperation.

ASRBAC still needs a considerable work, mainly in order to elaborate the very details of each type of autonomic administration operation and validate its Autonomic Computing properties. Actually, we propose the IOrg-AutoNet, SRBAC and ASRBAC models as a platform of a security administration solution in autonomic networks. Many detailed solutions should be studied having that platform as a basis. In this chapter, we could identify many future works. In terms of node cooperation, we still have as problems to solve the validation of NS-N modifications and the election of nodes for assignment to regular roles. Particularly, after describing a groundwork for policy negotiation, we could identify many issues that need to be handled, such as rule similarity computation, taking local decisions of policy modifications considering the recommendations of other nodes at different trust levels, and the multicasting of unified decisions at a trust level to administrative nodes of different trust levels. Finally, the issue of scalability must be addresses in a future work, whereby we may redefine ASRBAC using for basis a multi-organizational administrative model such as AROBAC07 [107], and considering certain communities of high populations in an IOrg-AutoNet as independent Infrastructureless Organizational Autonomic Networks.

Part III

Feasibility and Realization

Chapter 6

Case Study: Home Network

In this chapter, we explain the configuration of the components of SRBAC and ASRBAC and the specification of their policies in a Home Network. This chapter also redevelops a part of the Home Network example using a competing access control model in the literature and compares the resulting elements with those of SRBAC and ASRBAC.

Section 6.1 provides the initial configuration of SRBAC components after the network deployment phase. As pointed out in Assumption 3.4, we do not propose a deployment solution for an IOrg-AutoNet in general. In this section, we propose a scenario for this initialization phase in a home network, particularly to show how low-level access control configuration can be derived from high-level security specifications. SRBAC policies are presented in a language-independent manner. However, a representative part of those policies is developed in the SRBAC policy specification language (cf. Section 4.9), which is an extension of the RBAC Profile of XACML V2.0 [15], and separated in Appendix A.

Section 6.2 completes the home network example in the context of access control administration. It aims at enforcing the ASRBAC policies to adapt the initial SRBAC configuration to given examples of evolution transitions. The section starts by building the initial NS-N and NS-P elements, which are not presented in Section 6.1 because they concern the administration part. Afterward, the initial ASRBAC components are developed. Finally, evolution scenario examples are discussed, showing how ASRBAC's administrative actions can be performed.

Section 6.3 aims at comparing SRBAC/ASRBAC with the competing model Or-BAC [5] in the context of enforcement and implementation. The Or-BAC model has a comparable support for organizational and context-aware concepts. It is flexible enough to provide the desired access control characteristics in many application fields.

6.1 SRBAC in a Home Network

In this section, we study the configuration and policy specification of SRBAC in a home network. A family of three persons lives in this home. They are a father, a mother and their son. Each one has a cellular phone with a BlueTooth interface. The cellular phone of the father can access the Internet. The father also has a laptop with WiFi and BlueTooth

interfaces, and his car can send alert messages using a WiFi connection. The mother has a digital camera that can communicate using WiFi as well.

Assumption 6.1 *We assume the following for the initialization of this specific IOrg-AutoNet instance:*

- *The deployment of the home network is a centralized operation performed by a resident trusted and designated by all the other residents.*
- *This resident should do a simple network configuration and a high-level specification of resource sharing rules.*
- *He needs to be aware of the meanings of initial communities and initial trust levels in order to do the network configuration.*
- *He can use a human-like high-level language to specify security rules, whereby he should consider the initial trust levels, communities and node roles.*

Right after deployment, the initial configuration of SRBAC components and the initial low-level SRBAC policies are derived from the information provided by the resident. This section describes instances of such initial configurations and specifications.

6.1.1 Regular Roles and Node-Role Assignment

The father uses his laptop to do the simple network configuration illustrated in Table 6.1. Three communities are specified according to the availability expectations in the network. In general, the set of devices of each person constitutes a community, because they are supposed to be available for each other nearly permanently, with respect to the availability of the other devices of the network. As for the trust-based classification of the network nodes, the father just needs to distinguish between low-trust and high-trust devices according to application-based trust considerations.

The initial configuration illustrated in Table 6.1 may give the impression that the members of a same community have the same trust level. Actually, the two classifications are independent because they depend on different node attributes. For instance, if the father organizes work meetings at home, the devices of the attendees are expected to join his community during the meeting according to the availability conditions, but they will generally have trust levels lower than that of the father's devices.

The father should allow the laptop to exchange information with each device in the network at the deployment time using a safe channel[1], such as an infrared connection. The laptop will use such a channel to get information from each device about its computation and storage capabilities. Such information will be used to classify the network nodes according to their capabilities. For example, the Car and the Camera will not be able to perform asymmetric cryptography or to store public-key certificates. So they will belong to the Light-Duty capability class LD (cf. Section 3.1.3). Whereas, all the other devices are

[1] A safe channel may be a short-range connection created for exchanging specific data. It helps protecting a communication before securing the network.

Table 6.1: High-Level Configuration of a Home Network

	Low trust	High trust
Father's community		fPhone, Laptop, Car
Mother's community		mPhone, Camera
Son's community	sPhone	

fPhone: father's cellular Phone
mPhone: mother's cellular Phone
sPhone: son's cellular Phone

Table 6.2: Initial Node Categories in a Home Network

Node Category	Nodes
(H, F, HD)	fPhone, Laptop
(H, F, LD)	Car
(H, M, HD)	mPhone
(H, M, LD)	Camera
(L, S, HD)	sPhone

H: High trust	F: Father's community	HD: Heavy-Duty
L: Low trust	M: Mother's community	LD: Light-Duty
	S: Son's community	

supposed to have enough computation and storage capabilities to belong to the Heavy-Duty capability class HD. The Laptop will classify itself as Heavy-Duty, because the network deployment tool would have refused to run on it otherwise. This capability-based classification is used with the trust-based and availability-based classifications specified by the father (cf. Table 6.1) to create the initial node categories (cf. Section 3.1.4). Table 6.2 illustrates the initial node categories and their members in our Home Network example.

Being the device chosen by the father to accomplish the network deployment task, the Laptop will designate itself as the authority node of the father's community. The other nodes of the community will be non-administrative, even if certain of them are classified as heavy-duty, such as the father's cellular phone. Actually, classifying a node as heavy-duty simply means that it can be designated as authority when needed.

The laptop should also specify an authority node for each community. There is only one heavy-duty device in each of the two other communities (cf. Table 6.2). So the laptop will designate the mother's cellular phone and the son's cellular phone as the authority nodes of their respective communities. In case of many candidates for the authority role in a community, the current authority nodes in the network collaborate to select one of

Table 6.3: Initial NRA in a Home Network

Regular Role	Nodes
$(H, F, A, \{F\})$	Laptop
$(H, F, NA, \emptyset)$	fPhone, Car
$(H, M, A, \{M\})$	mPhone
$(H, M, NA, \emptyset)$	Camera
$(L, S, A, \{S\})$	sPhone

A: Authority node
NA: Non-Administrative node

See Section 4.5.1 for the specification of a regular role

those candidates using a specific election algorithm[2], such as choosing the node that has the best storage capabilities.

The resulting initial set of Node-Role Assignments (NRA; cf. Section 4.8.1) is illustrated in Table 6.3. At this point, inheritance can be applied between regular according to a Regular Role Hierarchy (RRH; cf. Section 4.5.2) roles to specify all the regular roles a node can activate. Figure 6.1 illustrates the initial RRH in our Home Network example.

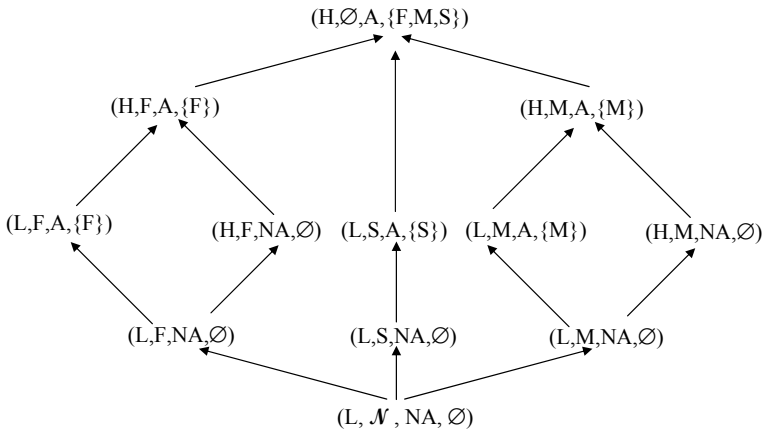
6.1.2 Permissions and Permission-Role Assignment

In addition to the simple network configuration represented by Table 6.1, the father uses his laptop to specify the following initial set of high-level policy specifications for sharing the resources of the home network:

Policy Set 6.1 *The device of a user in the home network may access a resource hosted by the device of another user according to the following rules:*

1. *All the home residents can use the Internet Connection tool (IC) on the father's cellular phone.*
2. *A resident should be as trusted as the son to read a set of Family Documents (FD) hosted by the parents' devices.*
3. *A resident should be as trusted as the father to read a set of Private Files (PF) hosted by the parents' devices.*
4. *Only the mother is trusted to remotely switch her Digital Camera (DC) on or off.*
5. *Only the father is trusted to remotely copy and delete PHotos (PH) from the mother's digital camera.*

[2]We assume that suitable distributed election algorithms are implemented in the access control system to allow authority nodes to collaborate to choose among a set of nodes for the assignment of a regular role.



$(H, \emptyset, A, \{F, M, S\})$: upper RRH delimiter, abstract role, having all permissions
$(L, \mathcal{N}, NA, \emptyset)$: lower RRH delimiter, abstract role, having no permissions

Figure 6.1: Initial RRH of a Home Network

6. *Only the father is trusted to handle his Car Alerts (CA) using a device that manages his community.*

The network deployment application will use the modules of the Security Policy System (cf. Section 2.2.4) to derive low-level access control rules from the previous high-level specifications. However, it first rewrites them in SRBAC terms. The rules for this rewriting are specified in a dedicated policy. For example, the following can be a set of rules for rewriting Policy Set 6.1, which produces the high-level SRBAC Policy Set 6.2:

- All the home residents ⇒ low-trust non-administrative nodes of every community.
- A resident should be as trusted as user u ⇒ the trust level of u is considered.
- Only the user u is trusted to ⇒ the trust level and the community of u are considered.
- Using a device that manages ⇒ the basic role A (Authority) is used. Otherwise, the basic role NA (Non-Administrative) is used by default.
- Identifying the host of a resource is needed in the implementation phase of the access control system, but it does not add any information to SRBAC policy specification.

Policy Set 6.2 *A node may access a resource hosted by another node, in the context of a secure relation binding them, according to the following rules:*

1. *Low-trust non-administrative nodes of every community can execute IC.*
2. *Non-administrative nodes having the trust level of the son in every community can read FD.*

Table 6.4: Initial Object Categories in a Home Network

Object Category	Resources	Permissions
(H, F, A)	CA	(CA,RWX)
(H, F, NA)	PF,PH	(PF,R),(PH,RW)
(H, M, NA)	PF,DC	(PF,R),(DC,X)
(H, S, NA)	PF	(PF,R)
(L, F, NA)	IC,FD	(IC,X),(FD,R)
(L, M, NA)	IC,FD	(IC,X),(FD,R)
(L, S, NA)	IC,FD	(IC,X),(FD,R)

A:	Authority node may access	R:	Read	IC:	Internet Connection
NA:	administrative role not required	W:	Write	FD:	Family Documents
		X:	eXecute	PF:	Private Files
				DC:	Digital Camera
				PH:	PHotos
				CA:	Car Alerts

3. *Non-administrative nodes having the trust level of the father in every community can read PF.*

4. *Non-administrative nodes having the trust level of the mother in the mother's community can activate/deactivate DC.*

5. *Non-administrative nodes having the trust level of the father in the father's community can copy or delete PH.*

6. *The Authority node of the father's community that has the trust level of the father can manage CA.*

The Laptop will categorize resources according to the trust level, the community membership and the basic role of the potential access requesting nodes (cf. Section 4.6.2). Table 6.4 specifies the initial object categories derived from the above high-level policy specifications, in addition to the associated resources and permissions.

The mapping between initial regular roles and initial object categories (cf. Definition 4.15) is then applied to specify the set of permissions associated to each regular role corresponding to each object category. The initial Permission-Role Assignments (PRA; cf. Section 4.8.2) of the home network of our example is illustrated in Table 6.5.

At this point, the initial SRBAC configuration is accomplished. It is time for the laptop to perform the final task related to access control in the network deployment phase. It communicates with the other authority nodes to inform them about their roles and to share the initial SRBAC configuration with them[3]. Each authority node communicates

[3]We assume that suitable cryptographic materials and protocols are used for a mutual authentication between two nodes in a first communication, after which a secure relation is established (cf. Section 4.4)

Table 6.5: Initial PRA in a Home Network

Regular Role	Permissions
$(H, F, A, \{F\})$	(CA,RWX)
$(H, F, NA, \emptyset)$	(PF,R),(PH,RW)
$(H, M, NA, \emptyset)$	(PF,R),(DC,X)
$(L, F, NA, \emptyset)$	(IC,X),(FD,R)
$(L, M, NA, \emptyset)$	(IC,X),(FD,R)
$(L, S, NA, \emptyset)$	(IC,X),(FD,R)

Table 6.6: Initial Low-Level SRBAC Policies in a Home Network

Regular Role	Nodes	Permissions
$(H, F, A, \{F\})$	Laptop	(IC,X),(FD,R),(PF,R),(PH,RW),(CA,RWX)
$(H, F, NA, \emptyset)$	fPhone, Car	(IC,X),(FD,R),(PF,R),(PH,RW)
$(H, M, A, \{M\})$	mPhone	(IC,X),(FD,R),(PF,R),(DC,X)
$(H, M, NA, \emptyset)$	Camera	(IC,X),(FD,R),(PF,R),(DC,X)
$(L, S, A, \{S\})$	sPhone	(IC,X),(FD,R)

then with the nodes of its community to inform them about their roles, and to share with them the PRA part of the configuration (cf. Table 6.5)[4].

Afterward, each node becomes able to create an access session in the context of a secure relation, in order to use the resources of another node by activating its direct regular role referenced in NRA (cf. Table 6.3) or one of its junior roles according to RRH (Figure 6.1). For example, the mother's cellular phone can activate its inherited role $(H, M, NA, \emptyset)$ to switch the mother's camera on (the permission (DC,X)), or its inherited role $(L, M, NA, \emptyset)$ to read family documents (the permission (FD,R)). Table 6.6 shows all the permissions that can be granted to a node through its direct regular role, either due to that latter itself or using inheritance between regular roles.

Table 6.6 is a language-independent specification of the initial low-level SRBAC policies corresponding to the high-level security rules in Policy Set 6.2. Appendix A provides the policy sample instances corresponding to the role $(H, F, A, \{F\})$, as can be expressed in our extended version of the RBAC Profile of XACML V2.0 [15] proposed for SRBAC. The use and extension of that profile of XACML [4] for SRBAC policy specification are motivated and described in Section 4.9.

to avoid the authentication step in future communications

[4] A node only needs the PRA specification to enforce SRBAC policies locally after identifying the regular role of the access requesting node in the secure relation binding them (cf. Section 4.8), while the authority nodes need to share all the SRBAC configuration because they are supposed to manage the access control system together (cf. Section 5.5)

Table 6.7: Initial NS-N in a Home Network

Node Category	Nodes
$(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}})$	fPhone, Laptop, Car, mPhone, Camera, sPhone
$-(H, \mathcal{N}, nil_{\mathcal{K}})$	fPhone, Laptop, Car, mPhone, Camera
$--(H, F, nil_{\mathcal{K}})$	fPhone, Laptop, Car
$---(H, F, HD)$	fPhone, Laptop
$---(H, F, LD)$	Car
$--(H, M, nil_{\mathcal{K}})$	mPhone, Camera
$---(H, M, HD)$	mPhone
$---(H, M, LD)$	Camera
$-(L, \mathcal{N}, nil_{\mathcal{K}})$	sPhone
$--(L, S, nil_{\mathcal{K}})$	sPhone
$---(L, S, HD)$	sPhone

$nil_{\mathcal{T}}$: Trust-based classification is not considered
$\mathcal{N}$: Community-based classification is not considered
$nil_{\mathcal{K}}$: Capability-based classification is not considered

6.2 ASRBAC in a Home Network

We focus here on the enforcement of ASRBAC policies in certain scenarios of our example, whereby we try to emphasize the autonomic aspects in our solution for access control administration. Beforehand, the initial NS-N (Network Structure for Nodes) and NS-P (Network Structure for Permissions), and the initial ASRBAC specific components, should be configured. Actually, those components should be created right after the network deployment, but they were not needed in the first part of the example (Section 6.1).

6.2.1 Network Structure Components

According to the descriptions of the organizational components NS-N (cf. Section 4.7.1) and NS-P (cf. Section 4.7.2), the node membership illustrated in Table 6.2 will propagate to senior node categories to form the initial NS-N as illustrated in Table 6.7. Similarly, the resource/permission membership illustrated in Table 6.4 will propagate to junior object categories to form the initial NS-P as illustrated in Table 6.8.

6.2.2 Administrative Roles and their Assignments

The authority nodes in the network will be the administrative nodes of the access control administration system. Their administrative roles are defined by their regular roles using the trust level and the administration scope parameters (cf. Definition 5.1). Table 6.9 illustrates the initial Administrative Node-Role Assignments (ANRA; cf. Definition 5.3) in our example. An initial Administrative Role Hierarchy (ARH; cf. Definition 5.2) is also

Table 6.8: Initial NS-P in a Home Network

Object Category	Resources	Permissions
$---(H, F, A)$	CA	(CA,RWX)
$---(H, F, NA)$	PF,PH	(PF,R),(PH,RW)
$--(H, F, nil_{\mathcal{B}})$	PF,PH,CA	(PF,R),(PH,RW),(CA,RWX)
$---(H, M, NA)$	PF,DC	(PF,R),(DC,X)
$--(H, M, nil_{\mathcal{B}})$	PF,DC	(PF,R),(DC,X)
$---(H, S, NA)$	PF	(PF,R)
$--(H, S, nil_{\mathcal{B}})$	PF	(PF,R)
$-(H, \mathcal{N}, nil_{\mathcal{B}})$	PF,DC,PH,CA	(PF,R),(DC,X),(PH,RW),(CA,RWX)
$---(L, F, NA)$	IC,FD	(IC,X),(FD,R)
$--(L, F, nil_{\mathcal{B}})$	IC,FD	(IC,X),(FD,R)
$---(L, M, NA)$	IC,FD	(IC,X),(FD,R)
$--(L, M, nil_{\mathcal{B}})$	IC,FD	(IC,X),(FD,R)
$---(L, S, NA)$	IC,FD	(IC,X),(FD,R)
$--(L, S, nil_{\mathcal{B}})$	IC,FD	(IC,X),(FD,R)
$-(L, \mathcal{N}, nil_{\mathcal{B}})$	IC,FD	(IC,X),(FD,R)
$(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}})$	IC,FD,PF,DC,PH,CA	(IC,X),(FD,R),(PF,R),(DC,X),(PH,RW),(CA,RWX)

$nil_{\mathcal{T}}$: Trust level of the access requesting node is not considered
$\mathcal{N}$: Community membership of the access requesting node is not considered
$nil_{\mathcal{B}}$: Basic role of the access requesting node is not considered

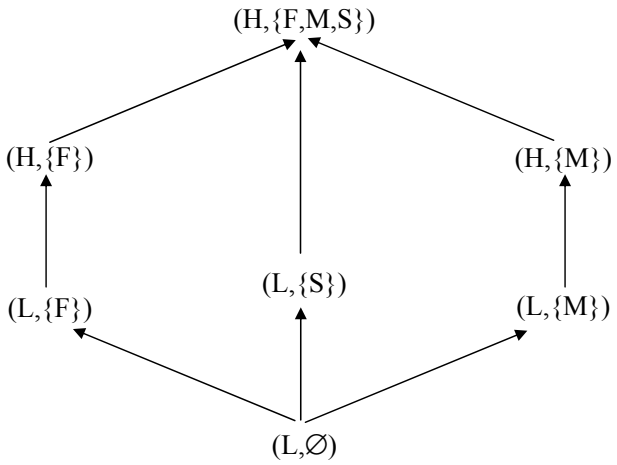
created. It is actually the result of removing the non-authority roles from the initial RRH, and ignoring the community membership and basic role parameters in each remaining regular role. Figure 6.2 illustrates the initial ARH of our example.

As for the initial Administrative Permission-Role Assignment (APRA; cf. Section 5.4.4), each authority node can use its administrative role or one of its junior roles to manage the elements of the SRBAC components NS-N (Table 6.7), NRA (Table 6.3), RRH (Figure 6.1), NS-P (Table 6.8) and PRA (Table 6.5) with respect to its administration domain, which does not concern its own access rights. Hence, a collaboration between authority nodes is required to manage certain elements of those SRBAC components when they are out of all the administration domains of all those authority nodes.

Table 6.9: Initial ANRA in a Home Network

Administrative Role	Authority Node
$(H, \{F\})$	Laptop
$(H, \{M\})$	mPhone
$(L, \{S\})$	sPhone

See Section 5.4.1 for the specification of an administrative role



$(H, \{F, M, S\})$: upper ARH delimiter, abstract role, having all administrative permissions
$(L, \emptyset)$: lower ARH delimiter, abstract role, having no administrative permissions

Figure 6.2: Initial ARH of a Home Network

Table 6.10: Initial APRA in a Home Network

Administrative Role	Organizational Scope	Hierarchical Scope
$(H, \{F\})$	$\{(H, F, nil_{\mathcal{K}}), (H, F, nil_{\mathcal{B}})\}$	$\{[(L, F, NA, \emptyset), (H, F, A, \{F\})[\}$
$(L, \{F\})$	$\{(L, F, nil_{\mathcal{K}}), (L, F, nil_{\mathcal{B}})\}$	$\{[(L, F, NA, \emptyset), (L, F, A, \{F\})[\}$
$(H, \{M\})$	$\{(H, M, nil_{\mathcal{K}}), (H, M, nil_{\mathcal{B}})\}$	$\{[(L, M, NA, \emptyset), (H, M, A, \{M\})[\}$
$(L, \{M\})$	$\{(L, M, nil_{\mathcal{K}}), (L, M, nil_{\mathcal{B}})\}$	$\{[(L, M, NA, \emptyset), (L, M, A, \{M\})[\}$
$(L, \{S\})$	$\{(L, S, nil_{\mathcal{K}}), (L, S, nil_{\mathcal{B}})\}$	$\{[(L, S, NA, \emptyset), (L, S, A, \{S\})[\}$
$\{(L, \{S\}), (H, \{M\}), (H, \{F\})\}$	$\{(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}), (nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{B}})\}$	$][(L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \{F, M, S\})[$

An administration domain is expressed using a set of NS-N and NS-P branches as node and permission pools respectively (organizational scope) and a set of regular role ranges as a hierarchical scope (cf. Table 5.1). Administrative actions are expressed using a set of predicates allowing the modification of NS-N, NRA, RRH, NS-P and PRA with respect to administration domains (cf. Definitions 5.4 and 5.5). Table 6.10 illustrates the initial APRA in our example of home networks. We will use in the following the corresponding administrative predicates to explain how ASRBAC is enforced in this Home Network.

6.2.3 Evolution Scenario 1: Visitor

A young relative will spend some months with the son at the family residence. He has a cellular phone with BlueTooth support and a digital audio player that can be accessed through BlueTooth as well. Those two devices will join the home network as new nodes in the community of the son, who takes this decision. The son will use his cellular phone, which is the authority node of its community, to add the devices of the young relative. The network administration interface on the son's cellular phone will then execute

Table 6.11: Modified NS-N in a Home Network after Evolution Scenario 1

Node Category	Nodes
$(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}})$	fPhone, Laptop, Car, mPhone, Camera, sPhone, rPhone, Player
$-(H, \mathcal{N}, nil_{\mathcal{K}})$	fPhone, Laptop, Car, mPhone, Camera
$--(H, F, nil_{\mathcal{K}})$	fPhone, Laptop, Car
$---(H, F, HD)$	fPhone, Laptop
$---(H, F, LD)$	Car
$--(H, M, nil_{\mathcal{K}})$	mPhone, Camera
$---(H, M, HD)$	mPhone
$---(H, M, LD)$	Camera
$-(L, \mathcal{N}, nil_{\mathcal{K}})$	sPhone, rPhone, Player
$--(L, S, nil_{\mathcal{K}})$	sPhone, rPhone, Player
$---(L, S, HD)$	sPhone, rPhone
$---(L, S, LD)$	Player

the two evolution transitions $nodeInsertion(rPhone, S)$ and $nodeInsertion(Player, S)$, where $rPhone$ is the new node representing the cellular phone of the relevant and $Player$ is the new node representing his digital audio player.

According to the effects of the insertion of a new node described in Definition 3.3, and to the administrative domain described in Table 6.10 for the administrative role of the son's cellular phone $(L, \{S\})$, this latter can perform the administrative action $canModifyNC((L, \{S\}), (L, S, nil_{\mathcal{K}}))$ to update the Network Structure for Nodes (NS-N) illustrated in Table 6.7. In terms of trustworthiness, the organizational scope of $(L, \{S\})$ is relevant, because a new node is assigned to the lowest trust level L initially (cf. Definition 3.3). As for capability classification, the son's cellular phone communicates with the two new devices using safe channels, and it finds out that the relative's cellular phone is heavy-duty, while his digital audio player is light-duty.

The son's cellular phone adds the two new nodes to the NS-N branch having the root category $(L, S, nil_{\mathcal{K}})$. The membership of the two new nodes will then propagate by inheritance to the dominating node categories $(L, \mathcal{N}, nil_{\mathcal{K}})$ and $(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}})$. Table 6.11 shows the resulting NS-N. The son's cellular phone, which is the authority node of its community, can take the decision of this NS-N modification independently. However, all the authority nodes should be aware of it. So the son's cellular phone multi-casts the modified parts of NS-N to the father's laptop and the mother's cellular phone. For this reason, it uses the Security Policy Logic-based Specification (SPLS) language and the autonomic reconfiguration module of the Security Policy System (cf. Figure 2.2).

After inserting the two new nodes rPhone and Player in the network structure by adding them to its community, the authority node sPhone must assign them to certain roles in order to determine their permissions. According to the administrative permissions assigned to the administrative role $(L, \{S\})$ in Table 6.10, sPhone can perform the administrative action $canAssign((L, \{S\}), @(L, S, nil_{\mathcal{K}}), [(L, S, NA, \emptyset), (L, S, A, \{S\})[)$ to assign

Table 6.12: Modified NRA in a Home Network after Evolution Scenario 1

Regular Role	Nodes
$(H, F, A, \{F\})$	Laptop
$(H, F, NA, \emptyset)$	fPhone, Car
$(H, M, A, \{M\})$	mPhone
$(H, M, NA, \emptyset)$	Camera
$(L, S, A, \{S\})$	sPhone
$(L, S, NA, \emptyset)$	rPhone, Player

regular roles to rPhone and Player, which belong to its organizational scope represented by the root node category $(L, S, nil_{\mathcal{K}})$, from its hierarchical scope represented by the regular role range $[(L, S, NA, \emptyset), (L, S, A, \{S\})[$.

The initial regular role of a node is non-administrative and identified by the trust level L (its initial trust level) and by its community. Such an initial regular role belongs to the hierarchical scope of sPhone. It is actually the included lower bound $(L, S, NA, \emptyset)$ of the regular role range of its administrative domain. The authority node sPhone assigns this regular role to rPhone and Player, by modifying the Node-Role Assignment (NRA) set illustrated in Table 6.3, which results in the modified NRA illustrated in Table 6.12, and multi-casts those modifications to the authority nodes Laptop and mPhone.

The relative wishes to access the Internet and to allow the son to download audio files from his digital audio player. The two new nodes rPhone and Player will be automatically able to use the Internet connection tool and read the family documents, as specified in the Permission-Role Assignments (PRA) in Table 6.5 for the regular role $(L, S, NA, \emptyset)$. However, that initial PRA does not allow the devices of the son to access the content of the new node rPhone. The son uses his cellular phone, which is an authority node in the network, to enhance the set of security rules. He creates the Policy Set 6.3 using the Human/Security System Interface (HSSI) of the Security Policy System (cf. Section 2.2.4). This latter rewrites the new security rule in SRBAC terms, which produces the high-level internal Policy Set 6.4.

Policy Set 6.3 *In addition to the Policy Set 6.1, we specify that:*

1. *Only the son is trusted to download Audio Files (AF) from the digital audio player of the relative.*

Policy Set 6.4 *In addition to the Policy Set 6.2, we specify that:*

1. *Non-administrative nodes having the trust level of the son in the son's community can read AF.*

According to Table 6.10 and the additional high-level Policy Set 6.4, the authority node sPhone can perform the administrative action $canModifyOC((L, \{S\}), (L, S, nil_{\mathcal{B}}))$

Table 6.13: Modified NS-P in a Home Network after Evolution Scenario 1

Object Category	Resources	Permissions
$---(H,F,A)$	CA	(CA,RWX)
$---(H,F,NA)$	PF,PH	(PF,R),(PH,RW)
$--(H,F,nil_{\mathcal{B}})$	PF,PH,CA	(PF,R),(PH,RW),(CA,RWX)
$---(H,M,NA)$	PF,DC	(PF,R),(DC,X)
$--(H,M,nil_{\mathcal{B}})$	PF,DC	(PF,R),(DC,X)
$---(H,S,NA)$	PF	(PF,R)
$--(H,S,nil_{\mathcal{B}})$	PF	(PF,R)
$-(H,\mathcal{N},nil_{\mathcal{B}})$	PF,DC,PH,CA	(PF,R),(DC,X),(PH,RW),(CA,RWX)
$---(L,F,NA)$	IC,FD	(IC,X),(FD,R)
$--(L,F,nil_{\mathcal{B}})$	IC,FD	(IC,X),(FD,R)
$---(L,M,NA)$	IC,FD	(IC,X),(FD,R)
$--(L,M,nil_{\mathcal{B}})$	IC,FD	(IC,X),(FD,R)
$---(L,S,NA)$	IC,FD,AF	(IC,X),(FD,R),(AF,R)
$--(L,S,nil_{\mathcal{B}})$	IC,FD,AF	(IC,X),(FD,R),(AF,R)
$-(L,\mathcal{N},nil_{\mathcal{B}})$	IC,FD,AF	(IC,X),(FD,R),(AF,R)
$(nil_{\mathcal{T}},\mathcal{N},nil_{\mathcal{B}})$	IC,FD,AF,PF,DC,PH,CA	(IC,X),(FD,R),(AF,R),(PF,R),(DC,X),(PH,RW),(CA,RWX)

Table 6.14: Modified PRA in a Home Network after Evolution Scenario 1

Regular Role	Permissions
$(H,F,A,\{F\})$	(CA,RWX)
$(H,F,NA,\emptyset)$	(PF,R),(PH,RW)
$(H,M,NA,\emptyset)$	(PF,R),(DC,X)
$(L,F,NA,\emptyset)$	(IC,X),(FD,R)
$(L,M,NA,\emptyset)$	(IC,X),(FD,R)
$(L,S,NA,\emptyset)$	(IC,X),(FD,R),(AF,R)

to categorize the new resource (AF) and the associated permission in the object category (L,S,NA), given that $(L,S,NA) \succeq_{OC} (L,S,nil_{\mathcal{B}})$. The new resource and permission propagate by inheritance in the Network Structure for Permissions (NS-P) illustrated in Table 6.8, which produces the new NS-P illustrated in Table 6.13.

The new PRA illustrated in Table 6.14 is deduced from the mapping between the regular role $(L,S,NA,\emptyset)$ and the object category (L,S,NA), whereby sPhone can use the administrative action $canAssignP((L,\{S\}),@(L,S,nil_{\mathcal{B}}),[(L,S,NA,\emptyset),(L,S,A,\{S\})[])$ to execute that mapping. The authority node sPhone multi-casts the modifications of PRA and NS-P to the authority nodes Laptop and mPhone, and each authority node multi-casts the modified elements of PRA to the nodes of its community.

Eventually, each node in the network can be aware of all the permissions that can be granted to a node through its direct regular role, either due to that latter itself or using inheritance between regular roles, which is illustrated in Table 6.15. This table represents the modified low-level SRBAC policies resulting from the two nod-insertion evolution transitions accomplished on behalf of the devices of the son's visitor.

Table 6.15: Low-Level SRBAC Policies in a Home Network after Evolution Scenario 1

Regular Role	Nodes	Permissions
$(H, F, A, \{F\})$	Laptop	(IC,X),(FD,R),(PF,R),(PH,RW),(CA,RWX)
$(H, F, NA, \emptyset)$	fPhone, Car	(IC,X),(FD,R),(PF,R),(PH,RW)
$(H, M, A, \{M\})$	mPhone	(IC,X),(FD,R),(PF,R),(DC,X)
$(H, M, NA, \emptyset)$	Camera	(IC,X),(FD,R),(PF,R),(DC,X)
$(L, S, A, \{S\})$	sPhone	(IC,X),(FD,R),(AF,R)
$(L, S, NA, \emptyset)$	rPhone, Player	(IC,X),(FD,R),(AF,R)

We can notice that the new permission concerning the download of audio files (AF,R) is assigned to all the nodes of the son's community, and to its authority node sPhone by inheritance. The son's device (sPhone) is specifically concerned by this new security rule, but regardless of its authority role, which grants the new permission to any device belonging to the same trust level and community.

6.2.4 Evolution Scenario 2: Mission

The resource management system (cf. Assumption 3.3) finds out that the relative's cellular phone (the non-administrative node rPhone) has more capabilities than the son's cellular phone (the authority node sPhone) to better manage the son's community. Actually, the both nodes are assigned to the capability class HD because each of them has the minimum capabilities required in this class. However, the resource management system is supposed to indicate the best candidates for a basic role each time the set of nodes evolves.

The resource management system is a distributed context-aware system in the network. Its output is monitored and analyzed by all the authority nodes (cf. Figure 5.3). One of the environment variables that can be evaluated at its output is called *Mission*. It indicates potential changes in the basic roles of the network nodes (cf. Table 5.2), which is the case in this scenario. The three authority nodes of the network Laptop, mPhone and sPhone (cf. Table 6.12) may then capture the context-aware information telling that rPhone is a better candidate for managing the son's community.

At this stage, the authority nodes analyze each the potential change of an authority node. The rules of this analysis are application-dependent. For the purpose of our example of a Home Network, we may propose the following rules for such analysis:

- A node can acquire an administrative role if it is likely to stay in the network for more than a month.
- If an authority node should be replaced, it may refuse if it is assigned to the highest trust level H.

The previous analysis rules would make each authority node accept to replace sPhone by rPhone in managing the son's community. This is accomplished by assigning the regular role $(L, S, NA, \emptyset)$ to sPhone, and then assigning the regular role $(L, S, A, \{S\})$ to rPhone.

This order of assignment is important, because a community may stay for some time without an authority node waiting for designating one, while a community may not have more than one authority node.

Table 6.10 indicates that the administrative role $(L, \{S\})$, which is assigned to the authority node sPhone (cf. Table 6.9), may use the range $[(L, S, NA, \emptyset), (L, S, A, \{S\})[$ to assign regular roles to nodes in the NS-N branch having the root node category $(L, S, nil_{\mathcal{K}})$. This means that the authority node sPhone can independently accomplish the first step of the evolution transition of this scenario by assigning the regular role $(L, S, NA, \emptyset)$ to itself. It informs the other authority nodes once it resigns from the administration of its community. This is actually not specific to this case of authority node replacement. In all cases, Node-Role Assignment (NRA) modifications performed by an authority node must be multi-casted to other authority nodes.

At this point, the network is not stable because the son's community has no authority node. This is not specific to this scenario either. When a community loses its authority node for any reason, the network reaches this unstable situation, and the other authority nodes must be aware of it. In this scenario for example, they become aware due to the multi-casting of the resigning authority node. In another scenario, an authority node might be lost, which is up to the resource management system to discover and report on its output environment variables. The solution for this unstable situation is to elect another node to replace the resigning authority node. This could be a node from another community. However, in this scenario, there is no need for the node election process because the resource management system indicates that the node rPhone is supposed to replace the resigning authority node sPhone.

Actually, the second step of this scenario is a cooperative effort between authority nodes. Even if the node sPhone can stay assigned to the administrative role $(L, \{S\})$ temporarily after accomplishing the first step in order to accomplish any required tasks, Table 6.10 states that the role of an authority node for the son's community is out of the hierarchical scope of any individual administrative role. Nevertheless, this table also states that the authority nodes can decide together how to manage a role of an authority node. In other words, they can perform together the administrative action $canAssign(AR, @(nil_{\mathcal{T}}, \mathcal{N}, nil_{\mathcal{K}}),](L, \mathcal{N}, NA, \emptyset), (H, \emptyset, A, \mathcal{C})[)$ (cf. Definition 5.5).

Usually, the authority nodes take individual decisions, and then they negotiate to unify the modified policies (cf. Algorithm 5.1). In this scenario, the negotiation algorithm will return without launching the negotiation process because the decisions will be identical, as long as the decision is already indicated by the resource management system. Eventually the modified NRA set and low-level SRBAC policies, which are illustrated in Tables 6.16 and 6.17 respectively, will be registered on the authority nodes Laptop and mPhone, the old authority node sPhone having become a non-administrative node will keep only the Permission-Role Assignment (PRA) which did not change (cf. Table 6.14), and finally the node rPhone will be informed about its new authority role by the other authority nodes and receive the whole up-to-date SRBAC configuration.

As depicted in Figure 5.3 and formally described in Definitions 5.1 and 5.3, Node-Role Assignments (NRA) in SRBAC define Administrative Node-Role Assignments (ANRA) in ASRBAC. Therefore, in this scenario, the set ANRA illustrated in Table 6.9 will au-

Table 6.16: Modified NRA in a Home Network after Evolution Scenario 2

Regular Role	Nodes
$(H, F, A, \{F\})$	Laptop
$(H, F, NA, \emptyset)$	fPhone, Car
$(H, M, A, \{M\})$	mPhone
$(H, M, NA, \emptyset)$	Camera
$(L, S, A, \{S\})$	rPhone
$(L, S, NA, \emptyset)$	sPhone, Player

Table 6.17: Low-Level SRBAC Policies in a Home Network after Evolution Scenario 2

Regular Role	Nodes	Permissions
$(H, F, A, \{F\})$	Laptop	(IC,X),(FD,R),(PF,R),(PH,RW),(CA,RWX)
$(H, F, NA, \emptyset)$	fPhone, Car	(IC,X),(FD,R),(PF,R),(PH,RW)
$(H, M, A, \{M\})$	mPhone	(IC,X),(FD,R),(PF,R),(DC,X)
$(H, M, NA, \emptyset)$	Camera	(IC,X),(FD,R),(PF,R),(DC,X)
$(L, S, A, \{S\})$	rPhone	(IC,X),(FD,R),(AF,R)
$(L, S, NA, \emptyset)$	sPhone, Player	(IC,X),(FD,R),(AF,R)

tonomously change, producing the modified ANRA illustrated in Table 6.18.

6.3 SRBAC vs. Or-BAC

The Or-BAC model [5] and its administrative model AdOr-BAC [32] incorporate and extend the access control concepts of many models in the literature including the RBAC model [89] and its administrative model ARBAC97 [84], as elaborated in the Or-BAC official site `http://www.orbac.org/`. Therefore, comparing SRBAC with Or-BAC allows us to consider a wide range of access control requirements in a variety of application fields. In this section we set up the access control system of the above example using Or-BAC, according to Table 6.1 and Policy Set 6.1, and compare the results with the previous

Table 6.18: Modified ANRA in a Home Network after Evolution Scenario 2

Administrative Role	Authority Node
$(H, \{F\})$	Laptop
$(H, \{M\})$	mPhone
$(L, \{S\})$	rPhone

Table 6.19: Initial OrBAC's NRA in a Home Network

Role	Context			Nodes
	Trust	Community	Scope	
A	H	F	{F}	Laptop
		M	{M}	mPhone
	L	S	{S}	sPhone
NA	H	F	∅	fPhone, Car
		M		Camera

H: High trust	F: Father's community	A: Authority
L: Low trust	M: Mother's community	NA: non-administrative
	S: Son's community	

SRBAC implementation of node roles and their associated privileges (cf. Table 6.6).

In order to compare the both models, we should assume that the father will not make any further configuration or specification after the simple network configuration of Table 6.1 and the high-level specifications of Policy Set 6.1. In other words, we should assume that the father's laptop will be responsible of performing all the needed next steps until reaching an initial Or-BAC implementation derived from those enduser data. Actually, deriving initial low-level elements from high-level enduser configurations is an essential characteristic of autonomic systems [56].

In Or-BAC vocabulary, the home network is the organization, and the subjects are the devices of the home residents. The role of a subject defines a set of privileges based on its administrative capabilities and responsibilities. As explained above in Section 6.1.1, the father's laptop can specify for each device a role depending on its computation and storage capabilities.

Given the need for only one authority node for a community, the initial roles in a community are "Authority" for a node having the required capabilities and "non-administrative" for the others. Other evolving attributes constitute the context component, which might reduce or increase a set of privileges during an access operation. In our example, the evolving attributes are the trust level, the community membership and the administration scope. As a result, Table 6.19 represents the configuration of Or-BAC components that can be derived from the enduser network configuration illustrated in Table 6.1. Actually, Table 6.19 illustrates the initial Or-BAC's Node-Role Assignments (NRA) taking into consideration context-based constraints.

The laptop is now supposed to derive permissions from the specifications of Policy Set 6.1 and assign them to roles taking the context into consideration. This is now possible because a classification based on roles and contextual attributes is already done as illustrated in Table 6.19.

A permission in Or-BAC is an action/activity performed on an object/view, whereby activities and views are organization-based abstractions of actions and objects respectively.

Table 6.20: Initial OrBAC's PRA in a Home Network

Role	Context			Permissions
	Trust	Community	Scope	
A	H	F	{F}	(CA,RWX)
NA	H	F	$\emptyset$	(PH,RW)
		M		(DC,X)
		any		(PF,R)
	L			(IC,X),(FD,R)

Table 6.21: Initial OrBAC's Node-Permission Assignments in a Home Network

Role	Context	Nodes	Permissions
A	H, F, {F}	Laptop	(IC,X),(FD,R),(PF,R),(PH,RW),(CA,RWX)
	H, M, {M}	mPhone	(IC,X),(FD,R),(PF,R),(DC,X)
	L, S, {S}	sPhone	(IC,X),(FD,R)
NA	H, F, $\emptyset$	fPhone, Car	(IC,X),(FD,R),(PF,R),(PH,RW)
	H, M, $\emptyset$	Camera	(IC,X),(FD,R),(PF,R),(DC,X)

For instance, an activity can be a subset of the set of actions {Read, Write, Execute}, and a view can be a set of objects undergoing the same access control rules, such as the alert management objects on the father's car. Actually, our focus is on roles and their associations, so we will simply consider an Or-BAC permission as a couple of a target and the related set of allowed operations, which is represented by the same notation already used in Section 6.1.2.

We assume that an Or-BAC system will have the necessary tools for deriving a set of rules equivalent to the high-level SRBAC rules of Policy Set 6.2, and then for applying a set of administration rules that allow it to derive the low-level permissions and assign them to the node roles with respect to context-aware attributes. Table 6.20 illustrates the initial Or-BAC's Permission-Role Assignments (PRA) taking into consideration context-based constraints.

Inheritance relationships can be defined between the elements of each type of Or-BAC components. Hence, an authority node inherits the privileges of a non-administrative node if it can put itself in the same context or an inherited context. For instance, the laptop can perform access actions as an authority node in the context (H,F,{F}), but it can also act as a non-administrative node in the inherited context (H,F,$\emptyset$). By applying this inheritance rule on roles and the context component, we can determine all the permissions a node can acquire through its role and inherited roles in different possible contexts, as illustrated in Table 6.21.

Table 6.3 (SRBAC's NRA) is equivalent to Table 6.19 (Or-BAC's NRA). Similarly, Table 6.6 (SRBAC's low-level policies) is equivalent to Table 6.21 (Or-BAC's Node-

Permission Assignments). This means that Or-BAC can be used for modeling access control in the Home Network of our example. This result is expected because Or-BAC is organization-based and context-aware, which is enough for deriving the initial access control elements in an IOrg-AutoNet. However, it might raise problems when it comes to the sought autonomic administration of the network evolution.

Actually, the application of the Or-BAC model requires the presence of security experts to accomplish low-level system configuration and policy specification. As for SRBAC, such expert users are not needed, because low-level elements can be autonomously derived as explained throughout Chapters 4 and 5, and instantiated in this Home Network example. Besides, we assumed that an Or-BAC system would have the necessary derivation mechanisms for playing the role of a security administrator, because it is possible to enhance it with such mechanisms, which allowed us to focus on the capabilities of the Or-BAC model itself. However, we can say that such derivation capabilities, which usually distinguish autonomic computing solutions, present a considerable advantage of SRBAC when compared with any other access control model.

Nonetheless, the regular roles in SRBAC have a more complex specification format because of integrating context-based information, which might cause redundancy in using contextual data. Besides, this mixture of the two different notions of role and context might require more processing for adapting roles to context changes; hence potentially causing a performance problem. Actually, we think that it is necessary to work on capability and context attributes altogether as role parameters to have an autonomic system. Therefore we intend to keep this methodology and try to solve any potential performance problems in future works. Besides, we accept some redundancy in handling context-based information to reach our goal of autonomic administration. The autonomic support in the regular role specification format is discussed in Chapter 5.

6.4 Conclusion

In order to clarify the concepts of SRBAC and ASRBAC, and to validate their applicability, this chapter provides a detailed example about the configuration of their components and the specification of their policies. It is about applying SRBAC and its administrative counterpart in a Home Network.

We can see in this case study how an access control system based on SRBAC/ASRBAC could build decentralized low-level access control components and policies depending on simple high-level network configuration and security rule specifications. We can also see how it could adapt to certain evolution transitions and context-aware changes without a human intervention.

The example was also developed in the competing model Or-BAC [5], and the result was compared with the implementation of SRBAC in order to emphasize the advantages and disadvantages of this latter.

Chapter 7

Prototype: Enforcement Model

This chapter proposes a model for enforcing SRBAC and ASRBAC policies, which defines an architecture for implementing a corresponding access control system. The proposed enforcement architecture is based on the layered PEI (Policy - Enforcement - Implementation) framework [87]. A first section describes the PEI framework, a main section presents our enforcement solution for SRBAC and ASRBAC policies, and a third section concludes the chapter with a quick summary and certain implementation considerations.

7.1 PEI Framework

This section describes the layered Policy - Enforcement - Implementation (PEI) framework. Policy is concerned with "what" security needs to be enforced, and mechanism is concerned with "how" the security is being enforced. The PEI framework bridges the gap between the what (policies) and the how (enforcement and implementation) by introducing layers of models. Figure7.1, which is extracted from [87], illustrates the PEI Models Framework.

As illustrated in Figure 7.1, between the top and bottom layers, which represent the informal requirements and the actual implementation respectively, there are three layers representing three types of models, namely the policy model, the enforcement model and the implementation model. At the layer of policy model, informal high-level objectives are translated into rigor and detail using a formal or quasi-formal notation. This is where SRBAC and ASRBAC policies are derived from high-level configuration. See Section 6.1 for a detailed example about such policy derivation. With respect to our vision of a security policy system for autonomic networks (cf. Section 2.2.4), it is at the policy model layer of PEI where the system uses instances of SPML (Security Policy Management Language) and SPLS (Security Policy Logic-based Specification). Moreover, the high-level informal security rules expressed using the so-called HSSI (Human - Security - System Interface) language in our sketch of a security policy system (Figure 2.2) should be used at the top layer of the PEI framework.

The enforcement and implementation models address the "how" aspect of enforcing a policy. The enforcement models address system architectures, block diagrams, and protocol flows, and they leave certain details to be elaborated in the implementation

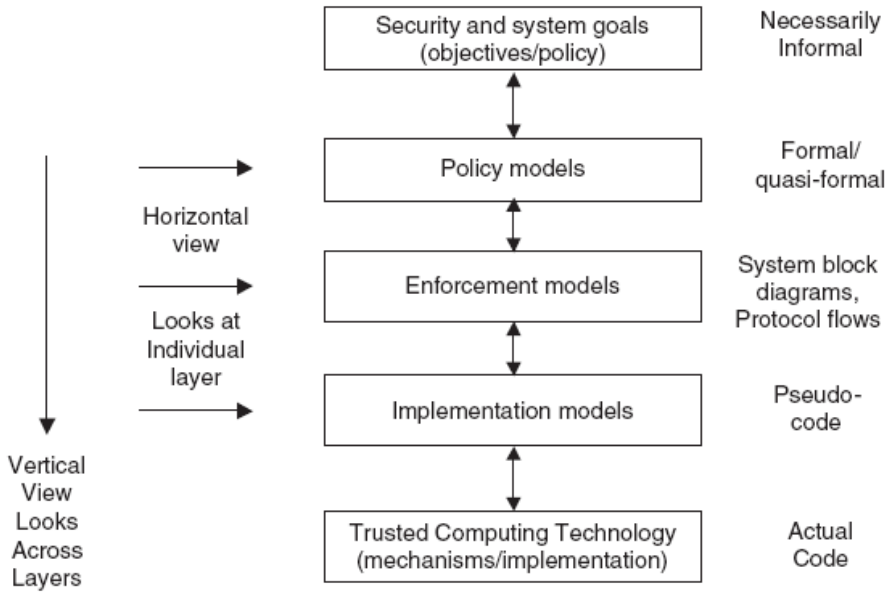


Figure 7.1: The PEI Models Framework (Extracted from [87])

models. The implementation models are focused on a pseudocode level of detail and precision. Actually, the enforcement and implementation models should define the link between an SPML specification and the actual code represented by Java Applications in our sketch of a security policy system (Figure 2.2). We should note here that we opted for the Java programming language as the expected implementation language in our solution because it is platform-independent, which is compatible with the heterogeneity of nodes in IOrg-AutoNets. However, the models that we present in this chapter can be realized using several programming languages. In general, the relationship between adjacent layers in the PEI framework is many-to-many.

7.2 Enforcement Model

This section presents the elements of an enforcement model based on the SRBAC model and its administrative counterpart ASRBAC. Actually, we describe an enforcement architecture to present our enforcement model. In terms of enforcement architecture, a typical authorization system includes a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP). Usually, an enforcement architecture defines a push mode or a pull mode. In a push mode, each subject presents his or her information to the PDP, and the decision is sent to PEP; while in a pull mode, a PEP collects the subject's information and queries the PDP for a policy decision.

Instances of both PDP and PEP are installed on each IOrg-AutoNet node. For an access, the PDP of the resource hosting node identifies the type of the secure relation binding it with the access requesting node, which allows it to determine the highest regular

role in the RRH (Regular Role Hierarchy) that can be activated by the access requesting node (cf. Section 4.5). The access requesting node presents its information (the activated role and the target resource). The PDP of the resource provider (RP) node (the host of the target resource) can then take its decision based on the information related to the secure relation type (highest regular role authorized), the target resource (object category), and the SRBAC policies (RPS, PPS and OCPS instances) that relate a regular role to a set of permissions through mapping with a set of object categories (cf. Section 4.9).

The PDP of the RP node may now ask the PEP on the same node to apply its access control decision. However, before taking its final decision, as well as during the access operation, the PDP keeps collecting and analyzing context-aware information (cf. Section 5.5.1) that may change the attributes of the access requesting node (Trustworthiness, Membership and Mission) or in certain cases the access control rules (Trust, Mobility, Authority and Specification). An authority RP node collects context-aware information from the outputs of context-aware systems (Figure 5.3). A non-administrative RP node collects such information through a communication with its authority node. If the collected context-aware information indicates a change, the access operation is blocked before or during the access session, waiting for the changes to take place, and then a new access control decision is taken accordingly. A non-administrative RP node only waits for possible policy modifications, while an authority RP node also executes them, eventually in collaboration with other authority nodes (cf. Section 5.5.6).

We propose a hybrid approach that uses both the push mode by allowing the access requesting node to push its access request information to the PDP of the RP node, and the pull mode by allowing this latter to pull context-aware information before and during an access session. Figure 7.2 illustrates the architecture corresponding to our enforcement model. Actually, it is the enforcement architecture of SRBAC policies, as depicted in the figure. Nevertheless, ASRBAC policies are integrated in SRBAC policies as already explained in Section 5.4.5 (see Appendix A for an ASRBAC policy example). Therefore, Figure 7.2 represents the enforcement architecture of both SRBAC and ASRBAC policies.

The architecture includes three main components, namely the access requesting node, the Resource Provider (RP) node (the host of the target object) and the set of context-aware systems of the IOrg-AutoNet. These latter are distributed systems in the network, whereby each authority node can use a distributed directory service to access environment variables, and each other node can use that service through its authority node. An access control monitor in the RP node collects resource information from the request of the access requesting node in order to identify the corresponding object category.

An access session is initialized by a client application on the access requesting node, in the context of a secure relation binding it with the RP node, and works as follows:

- An access request is sent to a service provider on the RP node (step 1).
- The activated regular role in the access session of the access requesting node, which is sent with the access request, is *pushed* to the Policy Decision Point (PDP) by the service provider (step 2).
- The PDP, which keeps monitoring the environment variables (step 3), *pulls* context-

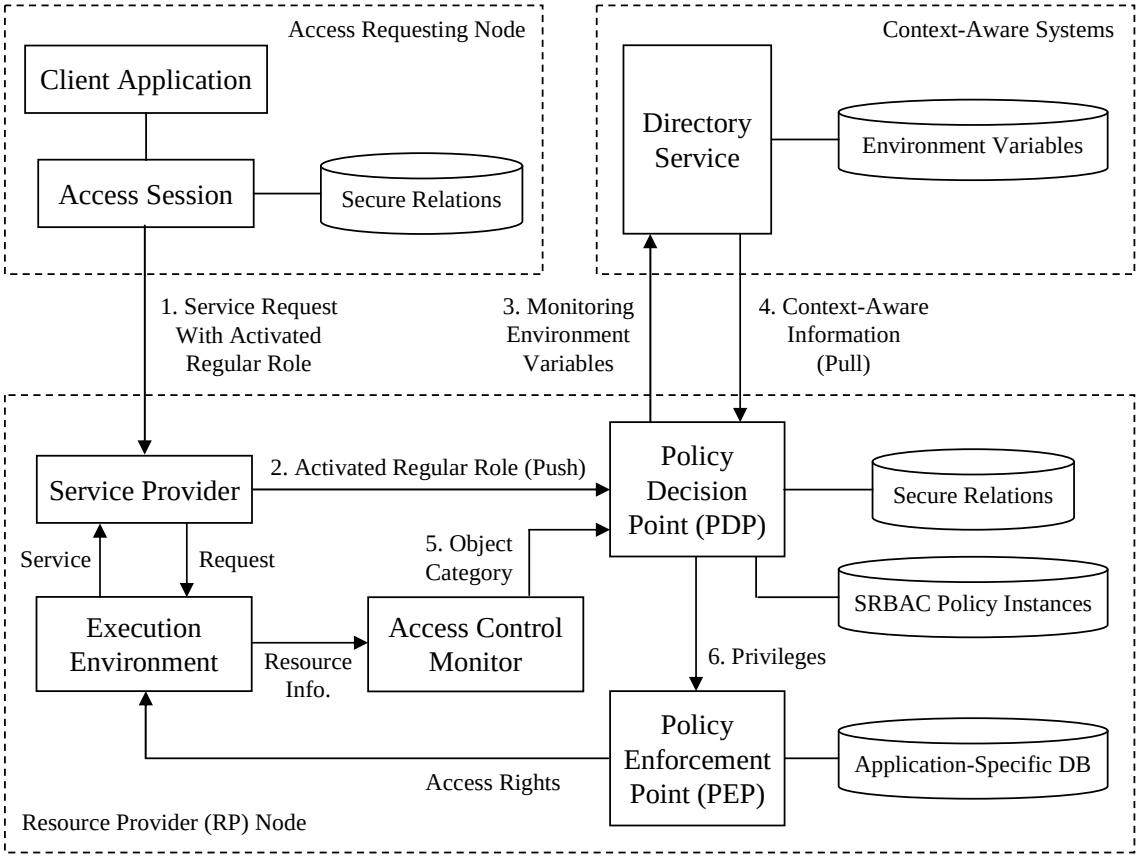


Figure 7.2: Enforcement Architecture for SRBAC Policies

aware information from the distributed directory service (step 4), and collects from the access control monitor the object category corresponding to the target resource, which is identified in the access request by the execution environment (step 5).

- The PDP has all the needed information (activated regular role, object category and context-aware changes if any). It uses its database of secure relations to identify the highest regular role allowed for the access requesting node in the secure relation binding them, and its SRBAC policy instances to identify the permissions of the activated regular role with respect to the object category of the target resource. It can then decide if the access may be granted, given that no context-aware changes are taking place. The decision is forwarded to the Policy Enforcement Point (PEP) in the form of confirmed or declined privileges (step 6), and enforced in the execution environment with regard to a database specific to the requested access.

7.3 Conclusion

We presented the layered PEI (Policy - Enforcement - Implementation) framework, which bridges the gap between the what (policies) and the how (enforcement and implementation) by introducing layers of models. Accordingly, we introduced our enforcement architecture to describe our enforcement model, considering that SRBAC and ASRBAC models define the policy model layer. We tried in this chapter to introduce an enforcement architecture that can be implemented using several techniques, given that the relationship between adjacent layers in the PEI framework is many-to-many.

In the implementation model layer of the PEI framework, several aspects should be considered, such as policy specifications, requesting node and regular role authentications, trusted update of node attributes and access control rules, and secure communications between computing components in the system. The mechanisms of a security system need also to meet the performance requirements of different access and collaborative scenarios. We leave the implementation model to a future work, whereby we will consider such aspects for realizing the enforcement architecture presented in this chapter.

Actually, we already handled many implementation issues in this thesis. In terms of policy specifications, we described how SRBAC policies (cf. Section 4.9), and eventually ASRBAC policies (cf. Section 5.4.5), can be expressed using our extension of the RBAC Profile of XACML V2.0 [15]. We give a detailed example about such XACML specifications in Appendix A. By using XACML [4], we propose an instance of SPML (Security Policy Management Language) in our sketch of a security policy system (cf. Section 2.2.4). However, we still need to study an instance language of SPLS (Security Policy Logic-based Specification) in this system. The Authorization Specification Language (ASL) [52] could be used for this purpose, as we already explained in a publication [10]. The use of ASL needs more study in a future work.

In terms of node authentication, a secure relation is established after a mutual authentication between two nodes (cf. Section 4.4). Because an access session is created in the context of a secure relation (Figure 4.5), node authentication is already achieved. We already proposed cryptographic protocols for mutual authentication between nodes in a

similar environment in a Master's thesis [7], but they need more elaboration and adaptation to the IOrg-AutoNet model in a future work. Besides, a solution is needed for the authenticity of the source of the activated regular role pushed to the PDP of the resource hosting node (the Resource Provider node).

Finally, in order to have trusted update of node attributes and SRBAC/ASRBAC policies, the access control system itself performs such modifications through its autonomous agents (the authority nodes), but we still need to study the trustworthiness of the outputs of the distributed context-aware systems (Figure 5.3). As for the secure communication between the components of the enforcement architecture, it may be achieved using our autonomic security architecture, for which we introduced a sketch in Section 2.2.3. In other words, certain implementation aspects already make part of our solution to a certain extent, but we need to complete our work on them and on other implementation issues before proposing our implementation model.

Chapter 8

Conclusion

8.1 Thesis Summary

The objective of this thesis is to establish a basis for security administration in autonomic networks. We built a research context in the first part, and tried to achieve our objective in the second part, in the context of an access control solution. The third part was introduced to prove the feasibility of the proposed solution, and to discuss its enforcement and implementation features.

8.1.1 Building the Context

In the first part, we presented our definition of autonomic networks (Definition 2.1) and proposed sketches for an autonomic security architecture (Figure 2.1) and an autonomic security policy system (Figure 2.2). This was done in the context of a theoretical review of existing related work.

Afterward, we defined a specific model of autonomic networks without preestablished infrastructures, and we denominated it IOrg-AutoNet (Definition 3.2). The IOrg-AutoNet network model defines an evolving context-aware organizational structure. We studied the evolution scheme of that structure and its potential effects on the administration responsibilities of the network nodes (Section 3.2). This research context was built in order to establish a framework for the definitions of an access control model and its autonomic administrative counterpart.

8.1.2 Achieving the Objective

In the second part, we defined the types of secure relations (Section 4.4) between the network nodes, and explained how the type of a secure relation defines the security materials to be used in its context, including access control policies. This allowed us then to define the Secure Relation Based Access Control (SRBAC) model for IOrg-AutoNets. Before defining SRBAC (Definition 4.17 and Figure 4.5) as a variant of RBAC [89], we discussed the different attributes of nodes and shared resources, and accordingly introduced a set of organizational and hierarchical access control components that should make part of

SRBAC whatever its basis model could be (Definitions 4.6, 4.10 and 4.11). In terms of enhancement of RBAC, we explained how SRBAC adapts to the context-aware and self-management requirements of IOrg-AutoNets. In order to propose a complete solution, and to point out its feasibility, we also studied the specification of SRBAC policies. Eventually, we proposed an extension of the RBAC Profile of XACML V2.0 [15] (Figure 4.6).

The definition of SRBAC was introduced in order to build the bases of an autonomic access control system. Therefore, the next step was the definition of its administrative counterpart ASRBAC. Because RBAC is the basis for SRBAC, and for many other reasons explained in a study of existing access control administration solutions (Section 5.2), we used ARBAC02 [73], which is an administrative counterpart of RBAC, as the basis of ASRBAC. The extension of the distributed organizational ARBAC02 with node collaboration, context-awareness, self-awareness, self-adaptation, self-optimization and self-configuration produced our autonomic, distributed organizational ASRBAC model. Actually, this is our main contribution in this thesis. Section 5.3 describes the details of this contribution before defining the ASRBAC model (Definition 5.6).

We defined the different components of the ASRBAC model, explaining the self-configuration aspect for each one. Definition 5.1 of administrative roles explains how they can be deduced from the regular roles of SRBAC. Eventually, Definition 5.2 of the administrative role hierarchy points out its inclusion relationship with the regular role hierarchy in SRBAC. Definition 5.3 of administrative node-role assignments proves that they are actually a subset of node-role assignments of SRBAC. And finally, the administrative permission-role assignments illustrated by Table 5.1 are proved to be autonomously implied by the specifications of the relevant administrative roles. In order to refine our access control administration solution, and to prove its autonomic computing features, we presented a generic ASRBAC policy specification in Section 5.4.5.

The definitions of the ASRBAC model and its components were not enough to emphasize all the features of autonomic computing in ASRBAC. Therefore, we dedicated Section 5.5 to the description of the details of those features. Context-awareness is a basic feature of the autonomic functionality in our solution. We began by a description of the context-aware information which are utilized in ASRBAC, and how the network evolution has effects on them, which was summarized in Table 5.2. It was then time to elaborate the different kinds of autonomic operations, from predefined self-management to autonomous evolution passing by autonomic control loop and mapping-based self-adaptation. Figure 5.3 illustrates all the features and types of autonomic administration in ASRBAC. We concluded this basic section of the thesis by explaining the need for node cooperation to accomplish the tasks defined by ASRBAC. Furthermore, we found it necessary to end the chapter with more details about node cooperation by proposing a groundwork for policy negotiation (Section 5.6).

8.1.3 Proving the Feasibility

In order to prove the feasibility of our solution, we used two methods in two respective chapters in the third part of the thesis. The first method was a case study of the details of SRBAC and ASRBAC policy enforcement in a home network, including a comparison with

the competing model Or-BAC [5]. The second method was built on describing a possible prototype as a proof of concepts through an enforcement architecture and implementation considerations for SRBAC/ASRBAC policies.

We opted for home networks in the case study because they are usually built without a preexisting infrastructure, and they need an evolving structuring, based on trust, availability and responsibility attributes, to control the access to their shared resources. They need to be autonomic, basically because they are not supposed to have expert users. In brief, a home network needs to be an IOrg-AutoNet (cf. Definition 3.1).

8.2 Future Work

Our solution is presented as a basis for an autonomic access control system. In other words, we do not claim that we propose the system itself. This is why we consider our research as a step, or maybe a number of steps, ***Toward* a Security Administration System for Autonomic Networks**. This means that a considerable amount of work must be done later. Besides, the solution has potential performance and scalability limitations, which have to be handled by eventual enhancements to the solution proposed currently, and have to be considered in future extensions. We pointed out the need for a future work whenever it was necessary in the thesis. Here we give a list of the most indispensable of those future works:

- **Autonomic Security Architecture:** The intra-node security architecture described in Section 2.2.3 needs more study. An interesting aspect to consider will be the security of the architecture itself. Another important point, is to develop this architecture using existing standards.

- **Security Policy System:** The security policy system illustrated in Figure 2.2 is just a sketch for a future work. Its different mechanisms need to be studied in details, above all the management of high-level security specifications and the autonomic reconfiguration of low-level security rules using logic-based instructions.

- **Network Deployment:** We assume that an initial network structure is already created (cf. Assumption 3.4). We just presented an expectation of this initialization phase in Section 3.2. The network deployment phase needs to be studied in details.

- **Self-Healing:** This is a very important and a basic functionality in autonomic networks. We just presented an evolution transition performed in this context, which is node banishment (cf. Definition 3.3). However, an indispensable future work should handle self-healing issues in IOrg-AutoNets.

- **Authentication:** In order to achieve a consistent security administration system for autonomic network, authentication issues should be handled as well. We already pointed out this need in Section 4.4, and proposed some initial ideas.

- **Security Validation:** The security properties of our solution need to be validated formally, and the prototype modeled in Chapter 7 should support this validation.

The dynamic behavior of our access control system, which is a response to the IOrg-AutoNet evolution transitions, may introduce certain security vulnerabilities. We need to study the vulnerabilities that may be caused by each administrative action. We need to define our security properties and perform a formal validation of the effects of each administrative action on them. For example, we may define a set of security properties representing the global security level of the access control system. Afterward, we may study the models of the different attacks that could be performed on the context-aware systems, so that the environment variables may change and the authority nodes may respond by a set of administrative actions that may decrease the global security level. A possible security property to verify in this context could be that "one authority node at least should have the highest trust level H".

- **Policy Negotiation:** The policy negotiation solution presented in Section 5.6 is just a sketch for a future enhancement, whereby the different presented mechanisms should be developed in details and a policy negotiation protocol should be formally specified and validated.

- **Node Cooperation:** As explained in Section 5.5.6, the authority nodes need to cooperate for many reasons in the access control system. In addition to policy negotiation, the collaborative validation of SRBAC component adaptation and the collaborative election of nodes for assignment to certain regular roles should be worked out.

- **Multi-Organizational Support:** A scalability issue may arise in certain scenarios where number of nodes in one or more communities may be considerable. A solution could be to handle each community as an independent infrastructureless organizational network in a multi-organizational context. The recent ROBAC model [106] and its administrative counterpart AROBAC07 model [107] could be used as bases for SRBAC and ASRBAC respectively.

- **Dynamic Attribute-Based Access Control:** The mixture of the two different concepts of role and context in the specification of a regular role in SRBAC might require more processing for adapting roles to context changes; hence potentially causing a performance problem. Actually, we think that it is necessary to work on administration privileges and context attributes altogether as role parameters to have an autonomic system. Therefore we intend to keep this methodology and try to solve any potential performance problems in a future work. An interesting solution could be to use the very recent model UCON [105] as a basis for SRBAC. It is attribute-based and supports attribute mutability, which may allow us to use all the attributes we want as access decision parameters and in a consistent and continuous manner. As for ASRBAC, we might need in this case to propose an administrative counterpart for UCON in the context of ASRBAC requirements, because UCON does not have an administrative counterpart yet.

Appendix A

XACML Sample Instances in a Home Network

In this appendix, we write some of the XACML sample instances of the Home Network example described in Chapter 6. Those instances correspond to the permissions of the role $(H, F, A, \{F\})$. This appendix allows to understand the extension we propose for the RBAC Profile of XACML V2.0 [15] to incorporate the organizational entities of SRBAC. It also gives an example of the integration of ASRBAC policy specifications in the SRBAC policies themselves. Moreover, it provides a basis to understand the realization aspects of certain access control administration operations elaborated in Chapter 5, such as the use of XACML instructions in policy negotiation (cf. Section 5.6).

A.1 RPS of a Regular Role

The following `Role <PolicySet>` (RPS) of the role $(H, F, A, \{F\})$ states that a node having this role can use the permissions specified by the associated `Permission <PolicySet>` (PPS):

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="RPS:H_F_A_F:role"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;anyURI-equal">
          <AttributeValue
              DataType="&xml;anyURI">&roles;(H,F,A,{F})</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="&role;"
              DataType="&xml;anyURI"/>
        </SubjectMatch>
```

```
      </Subject>
    </Subjects>
  </Target>
  <!-- Use permissions associated with the role (H,F,A,{F}) -->
  <PolicySetIdReference>PPS:H_F_A_F:role</PolicySetIdReference>
</PolicySet>
```

A.2 PPS of an Authority Regular Role

The following `Permission <PolicySet>` (PPS) of the role $(H, F, A, \{F\})$ states that it has the permission of managing SRBAC components because it is based on the authority basic role A (ASRBAC policy specification). The PPS instance also states that the other permissions are associated with the object category (H, F, A), and that the PPS instances associated with the junior roles $(L, F, A, \{F\})$ and $(H, F, NA, \emptyset)$ should be included:

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="PPS:H_F_A_F:role"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <!-- Permissions specifically for the authority (H,F,A,{F}) -->
  <Policy PolicyId="Permissions:specifically:for:the:H_F_A_F:authority"
      RuleCombiningAlgId="&rule-combine;permit-overrides">
    <!-- Permission to manage SRBAC components -->
    <Rule RuleId="Permission:to:manage:SRBAC:components"
        Effect="Permit">
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="&function;string-equal">
              <AttributeValue
                  DataType="&xml;string">organizational scope
              </AttributeValue>
              <ResourceAttributeDesignator
                  AttributeId="&resource;resource-id"
                  DataType="&xml;string"/>
            </ResourceMatch>
            <ResourceMatch MatchId="&function;string-equal">
              <AttributeValue
                  DataType="&xml;string">(H,F,nilK)</AttributeValue>
                  <!-- nilK = capability class ignored -->
              <ResourceAttributeDesignator
                  AttributeId="&resource;resource-NCroot"
                  DataType="&xml;string"/>
            </ResourceMatch>
            <ResourceMatch MatchId="&function;string-equal">
```

```
        <AttributeValue
            DataType="&xml;string">(H,F,nilB)</AttributeValue>
            <!-- nilB = basic role ignored -->
        <ResourceAttributeDesignator
            AttributeId="&resource;resource-OCroot"
            DataType="&xml;string"/>
      </ResourceMatch>
    </Resource>
    <Resource>
      <ResourceMatch MatchId="&function;string-equal">
        <AttributeValue
            DataType="&xml;string">hierarchical scope
        </AttributeValue>
        <ResourceAttributeDesignator
            AttributeId="&resource;resource-id"
            DataType="&xml;string"/>
      </ResourceMatch>
      <ResourceMatch MatchId="&function;string-equal">
        <AttributeValue
            DataType="&xml;string">(L,F,NA,{})</AttributeValue>
        <ResourceAttributeDesignator
            AttributeId="&resource;resource-includedStart"
            DataType="&xml;string"/>
      </ResourceMatch>
      <ResourceMatch MatchId="&function;string-equal">
        <AttributeValue
            DataType="&xml;string">(H,F,A,{F})</AttributeValue>
        <ResourceAttributeDesignator
            AttributeId="&resource;resource-excludedEnd"
            DataType="&xml;string"/>
      </ResourceMatch>
    </Resource>
  </Resources>
  <Actions>
    <Action>
      <ActionMatch MatchId="&function;string-equal">
        <AttributeValue
            DataType="&xml;string">manage</AttributeValue>
        <ActionAttributeDesignator
            AttributeId="&action;action-id"
            DataType="&xml;string"/>
      </ActionMatch>
    </Action>
  </Actions>
```

```
      </Target>
    </Rule>
  </Policy>
  <!-- Include permissions associated with the object category (H,F,A) -->
  <PolicySetIdReference>OCPS:H_F_A:category</PolicySetIdReference>
  <!-- Include permissions associated with the role (L,F,A,{F})) -->
  <PolicySetIdReference>PPS:L_F_A_F:role</PolicySetIdReference>
  <!-- Include permissions associated with the role (H,F,NA,{}) -->
  <PolicySetIdReference>PPS:H_F_NA_:role</PolicySetIdReference>
</PolicySet>
```

A.3 Inherited PPS Instances

We will not write the PPS instance associated with the role $(L, F, A, \{F\})$ because it has no direct permissions in the PRA (Permission-Role Assignments) set, as illustrated in Table 6.5. As for the PPS of the role $(H, F, NA, \emptyset)$, the following instance states that its permissions are associated with the object category (H, F, NA), and that the PPS instance associated with the junior role $(L, F, NA, \emptyset)$ should be included:

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="PPS:H_F_NA_:role"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <!-- Include permissions associated with object category (H,F,NA) -->
  <PolicySetIdReference>OCPS:H_F_NA:category</PolicySetIdReference>
  <!-- Include permissions associated with the role (L,F,NA,{}) -->
  <PolicySetIdReference>PPS:L_F_NA_:role</PolicySetIdReference>
</PolicySet>
```

The following PPS instance of the role $(L, F, NA, \emptyset)$ states that its permissions are associated with the object category (L, F, NA). The role inheritance chain stops here for $(H, F, A, \{F\})$. No need to add a reference to the junior role of $(L, F, NA, \emptyset)$, which is $(L, \mathcal{N}, NA, \emptyset)$, because it is the abstract delimiter role of the Regular Role Hierarchy (RRH), as illustrated in Figure 6.1:

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="PPS:L_F_NA_:role"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <!-- Include permissions associated with object category (L,F,NA) -->
  <PolicySetIdReference>OCPS:L_F_NA:category</PolicySetIdReference>
</PolicySet>
```

A.4 Associated OCPS Instances

The following `Object Category <PolicySet>` (OCPS) of the object category (H, F, A) specifies its associated permission (read, write and execute car alerts), and references

its junior object category $(H, F, nil_{\mathcal{B}})$ in the organizational component NS-P (Network Structure for Permissions):

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="OCPS:H_F_A:category"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <!-- Permissions specifically for the category (H,F,A) -->
  <Policy PolicyId="Permissions:specifically:for:the:H_F_A:category"
      RuleCombiningAlgId="&rule-combine;permit-overrides">
    <!-- Permission to read, write and execute car alerts -->
    <Rule RuleId="Permission:to:RWX:car:alerts"
        Effect="Permit">
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="&function;string-equal">
              <AttributeValue
                  DataType="&xml;string">car alerts</AttributeValue>
              <ResourceAttributeDesignator
                  AttributeId="&resource;resource-id"
                  DataType="&xml;string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="&function;decimal-equal">
              <AttributeValue
                  <!-- Same as Unix: 7 = RWX -->
                  DataType="&xml;decimal">7</AttributeValue>
              <ActionAttributeDesignator
                  AttributeId="&action;action-id"
                  DataType="&xml;decimal"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
  <!-- Reference to the junior category in NS-P -->
  <PolicySetIdReference>OCPS:H_F_nilB:category</PolicySetIdReference>
</PolicySet>
```

The following `Object Category <PolicySet>` (OCPS) of the category (H, F, NA) specifies its two associated permissions (read and write photos and read private files), and

references its junior object category $(H, F, nil_{\mathcal{B}})$ in the organizational component NS-P (Network Structure for Permissions):

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="OCPS:H_F_NA:category"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <!-- Permissions specifically for the category (H,F,NA) -->
  <Policy PolicyId="Permissions:specifically:for:the:H_F_NA:category"
      RuleCombiningAlgId="&rule-combine;permit-overrides">
    <!-- Permission to read and write photos -->
    <Rule RuleId="Permission:to:RW:photos"
        Effect="Permit">
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="&function;string-equal">
              <AttributeValue
                  DataType="&xml;string">photos</AttributeValue>
              <ResourceAttributeDesignator
                  AttributeId="&resource;resource-id"
                  DataType="&xml;string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="&function;decimal-equal">
              <AttributeValue
                  <!-- Same as Unix: 6 = RW -->
                  DataType="&xml;decimal">6</AttributeValue>
              <ActionAttributeDesignator
                  AttributeId="&action;action-id"
                  DataType="&xml;decimal"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
    <!-- Permission to read private files -->
    <Rule RuleId="Permission:to:R:private:files"
        Effect="Permit">
      <Target>
        <Resources>
          <Resource>
```

```
            <ResourceMatch MatchId="&function;string-equal">
              <AttributeValue
                  DataType="&xml;string">private files</AttributeValue>
              <ResourceAttributeDesignator
                  AttributeId="&resource;resource-id"
                  DataType="&xml;string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="&function;decimal-equal">
              <AttributeValue
                  <!-- Same as Unix: 4 = R -->
                  DataType="&xml;decimal">4</AttributeValue>
              <ActionAttributeDesignator
                  AttributeId="&action;action-id"
                  DataType="&xml;decimal"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
  <!-- Reference to the junior category in NS-P -->
  <PolicySetIdReference>OCPS:H_F_nilB:category</PolicySetIdReference>
</PolicySet>
```

The following `Object Category <PolicySet>` (OCPS) of the category (L, F, NA) specifies its two associated permissions (read family documents and execute Internet connection), and references its junior object category $(L, F, nil_{\mathcal{B}})$ in the organizational component NS-P (Network Structure for Permissions):

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    PolicySetId="OCPS:L_F_NA:category"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <!-- Permissions specifically for the category (L,F,NA) -->
  <Policy PolicyId="Permissions:specifically:for:the:L_F_NA:category"
      RuleCombiningAlgId="&rule-combine;permit-overrides">
    <!-- Permission to read family documents -->
    <Rule RuleId="Permission:to:R:family:documents"
        Effect="Permit">
      <Target>
        <Resources>
```

```
        <Resource>
          <ResourceMatch MatchId="&function;string-equal">
            <AttributeValue
                DataType="&xml;string">family documents</AttributeValue>
            <ResourceAttributeDesignator
                AttributeId="&resource;resource-id"
                DataType="&xml;string"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="&function;decimal-equal">
            <AttributeValue
                <!-- Same as Unix: 4 = R -->
                DataType="&xml;decimal">4</AttributeValue>
            <ActionAttributeDesignator
                AttributeId="&action;action-id"
                DataType="&xml;decimal"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
  <!-- Permission to execute Internet connection -->
  <Rule RuleId="Permission:to:x:Internet:connection"
      Effect="Permit">
    <Target>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="&function;string-equal">
            <AttributeValue DataType="&xml;string">
                Internet connection</AttributeValue>
            <ResourceAttributeDesignator
                AttributeId="&resource;resource-id"
                DataType="&xml;string"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="&function;decimal-equal">
            <AttributeValue
                <!-- Same as Unix: 1 = X -->
```

```
                DataType="&xml;decimal">1</AttributeValue>
            <ActionAttributeDesignator
                AttributeId="&action;action-id"
                DataType="&xml;decimal"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
 </Policy>
 <!-- Reference to the junior category in NS-P -->
 <PolicySetIdReference>OCPS:L_F_nilB:category</PolicySetIdReference>
</PolicySet>
```

The previous XACML sample instances altogether state that a node having the role $(H, F, A, \{F\})$ is an authority node that can manage SRBAC components with respect to the organizational scope represented by the set $\{(H, F, nil_{\mathcal{K}}), (H, F, nil_{\mathcal{B}})\}$ and the hierarchical scope represented by the set $\{[(L, F, NA, \emptyset), (H, F, A, \{F\})[\}$, and that it has the direct permission to read, write and execute car alerts. It inherits the direct set of permissions of the role $(H, F, NA, \emptyset)$ allowing to read and write photos and to read private files. It also inherits from the role $(H, F, NA, \emptyset)$ the direct set of permissions of its junior role $(L, F, NA, \emptyset)$ allowing to read family documents and to execute the Internet connection. This result corresponds to Table 6.6, in which we can see all the permissions that can be assigned to a node by activating a certain role, except for the administrative permissions.

Bibliography

[1] Ana project: Autonomic network architecture. `http://www.ana-project.org/`.

[2] International comittee for informational technology standards. `http://www.incits.org/`.

[3] National institute of standards and technology: Role based access control and role based security. `http://csrc.nist.gov/groups/SNS/rbac/`.

[4] Oasis extensible access control markup language (xacml). `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`.

[5] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, 2003.

[6] A. Adnane, R. T. de Sousa,Jr., C. Bidan, and L. Mé. Autonomic trust reasoning enables misbehavior detection in olsr. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, 2008.

[7] M. Aljnidi. Modèles et protocoles de sécurité pour un réseau domestique. Master's thesis, TELECOM ParisTech: Ecole Nationale Supérieure des Télécommunications, 2005.

[8] M. Aljnidi. Sécurité des réseaux mobiles autonomes. In *Actes du Premier workshop GET sur les réseaux spontanés*, 2006.

[9] M. Aljnidi and J. Leneutre. Autonomic security for home networks. In *Proceedings of the First International Workshop on Self-Organizing Systems*, 2006.

[10] M. Aljnidi and J. Leneutre. A security policy system for mobile autonomic networks. In *Proceedings of the First International Conference on Autonomic Computing and Communication Systems*, 2007.

[11] M. Aljnidi and J. Leneutre. Towards an autonomic security system for mobile ad hoc networks. In *Proceedings of the Third International Symposium on Information Assurance and Security*, 2007.

[12] M. Aljnidi and J. Leneutre. Security solutions in mobile autonomic networks. In *Proceedings of the Third International Conference on Information and Communication Technologies: From Theory to Applications*, 2008.

[13] M. Aljnidi and J. Leneutre. Asrbac: A security administration model for mobile autonomic networks (mautonets). In *Proceedings of the Second International Workshop on Autonomous and Spontaneous Security*, 2009.

[14] R. Ananthanarayanan, M. Mohania, and A. Gupta. Management of conflicting obligations in self-protecting policy-based systems. In *Proceedings of the Second International Conference on Automatic Computing*, 2005.

[15] A. Anderson. *Core and Hierarchical Role Based Access Control (RBAC) Profile of XACML v2.0.* OASIS, 2005.

[16] R. J. Anthony. A policy-definition language and prototype implementation library for policy-based autonomic systems. In *Proceedings of the IEEE International Conference on Autonomic Computing*, 2006.

[17] J. Bacon, K. Moody, and W. Yao. A model of oasis role-based access control and its support for active security. *ACM Transactions on Information and System Security*, 2002.

[18] S. Barker, M. J. Sergot, and D. Wijesekera. Status-based access control. *ACM Transactions on Information and System Security*, 2008.

[19] J. Barkley, K. Beznosov, and J. Uppal. Supporting relationships in access control using role based access control. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control*, 1999.

[20] R. Barrett, P. P. Maglio, E. Kandogan, and J. Bailey. Usable autonomic computing systems: The administrator's perspective. In *Proceedings of the First International Conference on Autonomic Computing*, 2004.

[21] E. Bertino, P. A. Bonatti, and E. Ferrari. Trbac: A temporal role-based access control model. *ACM Transactions on Information and System Security*, 2001.

[22] R. Bhatti, B. Shafiq, E. Bertino, A. Ghafoor, and J. B. D. Joshi. X-gtrbac admin: A decentralized administration model for enterprise-wide access control. *ACM Transactions on Information and System Security*, 2005.

[23] V. Cahill, E. Gray, J.-M. Seigneur, C. D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. M. Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2003.

[24] S. Capkun, L. Buttyan, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2003.

[25] L. Capra, W. Emmerich, and C. Mascolo. Carisma: Context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering*, 2003.

[26] S. Chang, Q. Chen, and M. Hsu. Managing security policy in a large distributed web services environment. In *Proceedings of the 27th Annual International Conference on Computer Software and Applications*, 2003.

[27] D. M. Chess, C. C. Palmer, and S. R. White. Security in an autonomic computing environment. *IBM Systems Journal*, 2003.

[28] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, 2001.

[29] J. Crampton and G. Loizou. Administrative scope and role hierarchy operations. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, 2002.

[30] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Transactions on Information and System Security*, 2003.

[31] F. Cuppens, N. Cuppens-Boulahia, and C. Coma. Multi-granular licences to decentralize security administration. In *Proceedings of the First International Workshop on Reliability, Availability and Security*, 2007.

[32] F. Cuppens and A. Miege. Administration model for or-bac. In *On The Move to Meaningful Internet Systems 2003: OTM 2003Workshops*. Springer, 2003.

[33] F. Cuppens and C. Saurel. Specifying a security policy: A case study. In *Proceedings of the 9th IEEE Workshop on Computer Security Foundations*, 1996.

[34] N. Damianou, A. Bandara, M. Sloman, and E. Lupu. A survey of policy specification approaches, 2002.

[35] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, 2001.

[36] R. Darimont and A. v. Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *Proceedings of the 4th ACM SIGSOFT Symposium on Foundations of Software Engineering*, 1996.

[37] M. A. C. Dekker, J. Crampton, and S. Etalle. Rbac administration in distributed systems. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, 2008.

[38] S. Dobson. Putting meaning into the network: Some semantic issues for the design of autonomic communications systems. In *Proceedings of the 1st International Workshop on Autonomic Communication*, 2005.

[39] S. Dobson. Achieving an acceptable design model for autonomic systems. In *Proceedings of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems*, 2007.

[40] S. Dobson, L. Coyle, and P. Nixon. Hybridising events and knowledge as a basis for building autonomic systems. *Journal of Trusted and Autonomic Computing*, 2006.

[41] S. Dobson, S. Denazis, A. Fernandez, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems*, 2006.

[42] H. M. Faheem. A multiagent-based approach for managing security policy. In *Proceedings of the Second IFIP International Conference on Wireless and Optical Communications Networks*, 2005.

[43] L. M. Feeney, B. Ahlgren, and A. Westerlund. Spontaneous networking: An application-oriented approach to ad hoc networking. *Communications Magazine, IEEE*, 2001.

[44] A. Garg, R. Battiti, and G. Costanzi. Dynamic self-management of autonomic systems: The reputation, quality and credibility (rqc) scheme. In *Proceedings of the 1st IFIP TC6 WG6.6 International Workshop on Autonomic Communication*, 2004.

[45] E. Gelenbe. Towards autonomic networks. In *Proceedings of the 2006 International Symposium on Applications and the Internet*, 2006.

[46] E. Gelenbe. Steps toward self-aware networks. *Communications of the ACM*, 2009.

[47] M. B. Ghorbel-Talbi, F. Cuppens, N. Cuppens-Boulahia, and A. Bouhoula. Managing delegation in access control models. In *Proceedings of the International Conference on Advanced Computing and Communications*, 2007.

[48] J. Glasgow, G. Macewen, and P. Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 1992.

[49] J. Y. Halpern and V. Weissman. Using first-order logic to reason about policies. *ACM Transactions on Information and System Security*, 2008.

[50] S. Hariri, B. Khargharia, H. Chen, J. Yang, Y. Zhang, M. Parashar, and H. Liu. The autonomic computing paradigm. *Cluster Computing*, 2006.

[51] P. Horn. Autonomic computing: Ibm's perspective on the state of information technology. Technical report, IBM Research, 2001.

[52] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, 1997.

[53] Y. Karabulut. Implementation of an agent-oriented trust management infrastructure based on a hybrid pki model. In *Proceedings of the 1st International Conference on Trust Management*, 2003.

[54] S. L. Keoh, E. Lupu, and M. Sloman. Peace: A policy-based establishment of adhoc communities. In *Proceedings of the 20th Annual Computer Security Applications Conference*, 2004.

[55] J. O. Kephart. Research challenges of autonomic computing. In *Proceedings of the 27th International Conference on Software Engineering*, 2005.

[56] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 2003.

[57] M. Koch and K. Pauls. Engineering self-protection for autonomous systems. In *Proceedings of the 9th International Conference on Fundamental Approaches to Software Engineering*, 2006.

[58] H. Koshutanski and F. Massacci. E pluribus unum: Deduction, abduction and induction, the reasonning services for access control in autonomic communication. In *Proceedings of the 1st IFIP TC6 WG6.6 International Workshop on Autonomic Communication*, 2004.

[59] H. Koshutanski and F. Massacci. Interactive access control for web services. In *Proceedings of the 19th IFIP Information Security Conference*, 2004.

[60] M. Lacoste, A. Saxena, T. Jarboui, U. Lucking, B. Steinke, J. Pulou, J. Polakovic, and S. Buljore. Towards autonomic security in beyond 3g infrastructures - end-to-end reconfigurability ii white paper, 2007.

[61] B. Lampson. Protection. In *Proceedings of the 5th Princeton Symposium on Information Science and Systems*, 1971.

[62] J. Ligatti, L. Bauer, and D. Walker. Run-time enforcement of nonsafety policies. *ACM Transactions on Information and System Security*, 2009.

[63] M. P. Locatelli and G. Vizzari. Awareness in collaborative ubiquitous environments: The multilayered multi-agent situated system approach. *ACM Transactions on Autonomous and Adaptive Systems*, 2007.

[64] Y. Lu, L. Zhang, Y. Liu, and J. Sun. A distributed domain administration of rbac model in collaborative environments. In *Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design*, 2006.

[65] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-securing ad hoc wireless networks. In *Proceedings of the 7th IEEE Symposium on Computers and Communications*, 2002.

[66] P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Bertino. Xacml policy integration algorithms. *ACM Transactions on Information and System Security*, 2008.

[67] T. Messerges, J. Curkier, T. Kevenaar, L. Puhl, R. Struik, and E. Callaway. A security design for a general purpose, self-organizing, multi-hop ad-hoc wireless network. In *Proceedings of the First ACM Workshop on Security of Ad-hoc and Sensor Networks*, 2003.

[68] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the IFIP 6th Joint Working Conference on Communications and Multimedia Security*, 2002.

[69] A. Miege. *Definition of a Formal Framework for Specifying Security Policies. The Or-BAC Model and Extensions*. PhD thesis, TELECOM ParisTech: Ecole Nationale Supérieure des Télécommunications, 2005.

[70] D. L. Mills. The autokey security architecture, protocol and algorithms. Technical report, University of Delaware, 2006.

[71] S. Oh and S. Park. An improved administration method on role-based access control in the enterprise environment. *Journal of Information Science and Engineering*, 2001.

[72] S. Oh and R. Sandhu. A model for role administration using organization structure. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, 2002.

[73] S. Oh, R. Sandhu, and X. Zhang. An effective role administration model using organization structure. *ACM Transactions on Information and System Security*, 2006.

[74] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security*, 2000.

[75] J. Park and J. Chung. Design of sps model using mobile agent system. In *Proceedings of the IEEE 37th Annual International Carnahan Conference on Security Technology*, 2003.

[76] J. Park and R. Sandhu. The uconabc usage control model. *ACM Transactions on Information and System Security*, 2004.

[77] A. A. Pirzada and C. McDonald. Trust establishment in pure ad-hoc networks. *Wireless Personal Communications*, 2006.

[78] S. Poslad. Specifying protocols for multi-agent systems interaction. *ACM Transactions on Autonomous and Adaptive Systems*, 2007.

[79] N. Prigent, C. Bidan, J.-P. Andreaux, and O. Heen. Secure long term communities in ad hoc networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[80] M. A. Razzaque, S. Dobson, and P. Nixon. Cross-layer architectures for autonomic communications. *Journal of Network and Systems Management*, 2006.

[81] A. J. Rocke and R. F. Demara. Confidant: Collaborative object notification framework for insider defense using autonomous network transactions. *Autonomous Agents and Multi-Agent Systems*, 2006.

[82] P. V. Roy. Self management and the future of software design, 2006.

[83] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 2009.

[84] R. Sandhu, V. Bhamidipati, and Q. Munawer. The arbac97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 1999.

[85] R. Sandhu, D. Ferraiolo, and R. Kuhn. The nist model for role-based access control: Towards a unified standard. In *Proceedings of the Fifth ACM Workshop on Role-Based Access Control*, 2000.

[86] R. Sandhu and Q. Munawer. The arbac99 model for administration of roles. In *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999.

[87] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by trusted computing and pei models. In *Proceedings of the ACM Symposium on Information, Computer and Communication Security*, 2006.

[88] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 1993.

[89] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 1996.

[90] S. Schmid, M. Sifalakis, and D. Hutchison. Towards autonomic networks. In *Proceedings of the First International IFIP TC6 Conference on Autonomic Networking*, 2006.

[91] K. E. Seamons, T. Chan, E. Child, M. Halcrow, A. Hess, J. Holt, J. Jacobson, R. Jarvis, A. Patty, B. Smith, T. Sundelin, and L. Yu. Trustbuilder: Negotiating trust in dynamic coalitions. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2003.

[92] M. Seredynski, P. Bouvry, and M. A. Klopotek. Evolution of cooperation in ad hoc networks under game theoretic model. In *Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access*, 2006.

[93] B. Shafiq, J. Joshi, E. Bertino, and A. Ghafoor. Secure interoperation in a multidomain environment employing rbac policies. *IEEE Trans. Knowl. Data Eng.*, 2005.

[94] M. Shehab, E. Bertino, and A. Ghafoor. Secure collaboration in mediator-free environments. In *Proceedings of the 12th ACM Conference on Computer and Communication Security*, 2005.

[95] D. K. Smetters and R. E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. In *Proceedings of the 2002 Workshop on New Security Paradigms*, 2002.

[96] M. Smirnov. Autonomic communication: Research agenda for a new communication paradigm, 2004.

[97] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ubiquitous computing. *Computer*, 2002.

[98] R. K. Thomas. Team-based access control (tmac): A primitive for applying role-based access controls in collaborative environments. In *Proceedings of the Second ACM Workshop on Role-Based Access Control*, 1997.

[99] R. K. Thomas and R. S. Sandhu. Task-based authorization controls (tbac): A family of models for active and enterprise-oriented autorization management. In *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI: Status and Prospects*, 1997.

[100] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong. Access control in collaborative systems. *ACM Computing Surveys*, 2005.

[101] S. S. Yau, Y. Yao, Z. Chen, and L. Zhu. An adaptable security framework for service-based systems. In *Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, 2005.

[102] F. Zambonelli, M.-P. Gleizes, M. Mamei, and R. Tolksdorf. Spray computers: Explorations in self-organization. *Pervasive and Mobile Computing*, 2005.

[103] G. Zhang and M. Parashar. Dynamic context-aware access control for grid applications. In *Proceedings of the 4th International Workshop on Grid Computing*, 2003.

[104] X. Zhang, M. Nakae, M. J. Covington, and R. Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Transactions on Information and System Security*, 2008.

[105] X. Zhang, F. Parisi-Presicci, and R. Sandhu. Formal model and policy specification of usage control. *ACM Transactions on Information and System Security*, 2005.

[106] Z. Zhang, X. Zhang, and R. Sandhu. Robac: Scalable role and organization based access control models. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2006.

[107] Z. Zhang, X. Zhang, and R. Sandhu. Towards a scalable role and organization based access control model with decentralized security administration. In *Handbook of Research on Social and Organizational Liabilities in Information Security*. Information Science Reference (an imprint of IGI Global), 2009.

[108] H. Zhou and S. N. Foley. A collaborative approach to autonomic security protocols. In *Proceedings of the 2004 Workshop on New Security Paradigms*, 2004.