

DÉDICACES

À mon feu père SOPJIO FIDÉLE et à ma mère JIOFACK CORENTINE SYLVIE.

REMERCIEMENTS

Je rends grâce à DIEU pour son amour envers nous, son soutien incommensurable, pour nous avoir accordé la santé et la force pour mener à bien ce travail. Mes remerciement les plus sincères :

- Au Dr AZANGUEZET QUIMATIO Benoit Martin pour la direction de mes travaux de Master et mon initiation avec bienveillance au monde de la recherche. Il a su avec succès me transmettre son dynamisme et rien de ce que je dirai ne pourra égaler tout ce qu'il a fait pour moi durant cette épreuve.
- À tous les membres de mon jury, pour l'attention qu'ils porteront particulièrement à ce travail;
- À tous les enseignants du département de Mathématiques-Informatique, qui ont su bien m'encadrer tout au long de ce cycle de Master.
- À Mr TSOGNONG Fidèle pour le soutien indéfectible qu'il m'a apporté.
- À ma maman DJIOFACK Corentine et mon tuteur NGINTEDEM Jérôme pour tous les sacrifices consentis tout au long de mes études.
- À Mes frères et sœurs (Chaneline, Aurélien, Bertol, Rophanie et Nadège) et tous mes oncles et tantes;
- À tous mes camarades de promotion pour leurs soutient morale et intellectuel tout au long de ce travail;
- À mes admis DONGMO Alex et TEMATEU Roslyn pour l'aide qu'ils m'ont apporté lors l'implémentation de mon approche.
- À tous ceux qui de près ou de loin ont contribué à la réalisation de ce travail.

TABLE DE MATIÈRES

Dédicaces	i
Remerciements	ii
Table de matières	iii
Liste des acronymes	viii
Liste des figures	ix
Résumé	xi
Abstract	xii
Introduction	1
Contexte	1
Problématique	3
Objectifs	4
Plan de travail	4
Chapter I ► Généralité sur la sécurité des systèmes d'information	5
I.1 - Sécurité des systèmes d'information	5
I.1.1 - Définition de la sécurité	5
I.1.2 - Confidentialité	6
I.1.3 - Intégrité	6
I.1.4 - Disponibilité	7
I.2 - Contrôle d'accès	7
I.2.1 - Définition du contrôle d'accès	7
I.2.2 - Le contrôle d'accès et les autres services	8
I.2.3 - Politique de sécurité	9
I.2.4 - Conclusion	10
Chapter II ► état de l'art sur les politiques et modèles de contrôle d'accès	11
II.1 - Introduction	11
II.2 - Politiques et modèles de contrôle d'accès discrétionnaires (DAC)	12
II.2.1 - Modèle de matrice d'accès	12
II.2.2 - Implémentation des politiques discrétionnaires	13

II.2.2.1 - Liste de contrôle d'accès	13
II.2.2.2 - Les capacités	14
II.2.3 - Vulnérabilité des politiques discrétionnaire	14
II.3 - Politiques et modèles de contrôle d'accès obligatoire (MAC)	15
II.3.1 - Vulnérabilité des politiques Obligatoire	17
II.4 - Politiques et modèles de contrôle d'accès à base de rôles (RBAC)	17
II.4.1 - Politiques et modèles de contrôle d'accès basés sur les rôles (RBAC)	17
II.4.1.1 - Composants principaux de RBAC	18
II.4.1.2 - Hiérarchie des rôles	19
II.4.1.3 - Notion de séparation des tâches	19
II.4.1.4 - Avantages du modèle RBAC	20
II.4.1.5 - Inconvénient du modèle RBAC	20
II.4.2 - Politiques et modèles de contrôle d'accès basé sur les organisations (Or-BAC)	21
II.4.2.1 - Relations existantes entre les entités du niveau concret et les entités du niveau abstrait	22
II.4.2.2 - Limites du modèle Or-BAC	24
II.5 - Politiques et modèles de contrôle d'accès basés sur les attributs (ABAC)	24
II.5.1 - Les principaux composants d'ABAC	25
II.5.2 - Architecture de fonctionnement du modèle ABAC	26
II.5.3 - Avantages du modèle ABAC	27
II.5.4 - Inconvénient du modèle ABAC	28
II.6 - Politiques et modèles de contrôle d'accès basés sur les rôles améliorés par les attributs(AERBAC)	29
II.6.1 - Composants du modèle AERBAC	29
2.6.2 - Modèle formel AERBAC	31
2.6.2.1 - Décision d'accès	31
2.6.3 - Inconvénients	33
2.7 - Politiques et modèles de contrôle d'accès basés sur les rôles et les attributs(ARBAC)	33
2.7.1 - Composants du modèle ARBAC	33
2.7.2 - Avantages du modèle ARBAC	35
2.7.3 - Inconvénients du modèle ARBAC	35
2.8 - Politiques et modèles de contrôle d'accès basés sur la hiérarchie organisationnelle (HOr-RBAC)	35
2.8.1 - Composants de base du modèle HOr-BAC	36

2.8.1.1 - Unité organisationnelle	36
2.8.1.2 - Employé métier	36
2.8.1.3 - Ressources	36
2.8.1.4 - Requête	37
2.8.2 - Mode de traitement d'une requête dans HOr-BAC	37
2.8.3 - Politique de sécurité du modèle HOr-BAC	37
2.8.3.1 - Les permissions dites de préparation	37
2.8.3.2 - Les permissions dites de validations	37
2.8.3.3 - Les permissions dites de préparation	38
2.8.4 - Architecture de fonctionnement du modèle HOr-BAC: par- pheur électronique	38
2.8.5 - Inconvénient du modèle HOr-BAC	39
2.9 - Conclusion	40

**Chapter 3 ► une approche orientée attributs pour le contrôle d'accès basé
sur la hiérarchie organisationnelle (AHor-BAC) 41**

3.1 - Introduction	41
3.2 - Motivation	41
3.3 - Concepts de base du modèle AHor-BAC	42
3.3.1 - Organisation	42
3.3.2 - Employé Métier	42
3.3.3 - Ressource	43
3.3.4 - Requête	43
3.3.5 - Unité organisationnelle et hiérarchie organisationnelle	43
3.3.5.1 - Unité opérationnelle	43
3.3.5.2 - Unité administrative	44
3.3.6 - Structure organisationnelle	44
3.3.7 - Contexte	45
3.3.8 - Mode de traitement	45
3.3.9 - Règles	45
3.4 - Présentation des relations dans AHor-BAC	45
3.4.1 - Employé métier et Unité opérationnelle	46
3.4.2 - Employé métier et Unité Administrative	46
3.4.3 - Unité Administrative et Unité Opérationnelle	46
3.4.4 - Unité Administrative et Unité Administrative	46
3.4.5 - Ressources et Vues	47
3.4.6 - Actions et Requêtes	47

3.4.7 - La relation définit	48
3.4.8 - Permissions	48
3.4.9 - Peut-suggérer et peut-traiter	50
3.4.10 - Attribution des Règles aux Permissions de bas niveau	50
3.5 - Algorithme du parapheur électronique	51
3.5.1 - La phase d'initialisation du parapheur électronique	52
3.5.2 - La phase d'émission d'une requête du parapheur électronique	53
3.5.3 - La phase de traitement d'une requête du parapheur électronique	54
3.6 - Conclusion	55
 Chapter 4 ► Proposition d'un langage permettant d'exprimer les poli- tiques d'accès basé sur la hiérarchie organisationnelle et les attributs	 57
4.1 - Introduction	57
4.2 - Langage de balisage extensible de contrôle d'accès	57
4.2.1 - Présentation de XACML	57
4.2.2 - Architecture du langage XACML	58
4.2.2.1 - Point d'administration des politiques (PAP)	58
4.2.2.2 - Point d'information politique (PIP)	58
4.2.2.3 - Point de récupération des politiques (PRP)	58
4.2.2.4 - Point de décision politique (PDP)	58
4.2.2.5 - Point d'application des politiques (PEP)	58
4.2.3 - Flux d'autorisation XACML	58
4.2.4 - Structure et syntaxe du langage XACML	60
4.2.4.1 - Cible	60
4.2.4.2 - Règle	61
4.2.4.3 - Politique	61
4.2.4.4 - Algorithme de combinaison	62
4.2.4.5 - Ensemble de règles	62
4.2.4.6 - Les conditions	62
4.2.5 - Avantages et inconvénient du langage XACML	63
4.3 - Extension du langage XACML pour l'expression des politiques basé sur hiérarchie organisationnelle et les attributs	64
4.3.1 - Insertion du parapheur électronique dans l'architecture du lan- gage XACML	64
4.3.1.1 - Le parapheur électronique	64
4.3.1.2 - Policy administration point	64
4.3.1.3 - Policy information point	64

4.3.1.4 - Polycy provider	64
4.3.2 - Modèle de flux de donnée dans notre langage	64
Conclusion générale	67
Bilan	67
Quelques perspectives	67
Bibliography	68

LISTE DES ACRONYMES

ABAC	Attribute Based Access Control.
AC	Access Control.
ACL	Access Control List.
DAC	Discretionary Access Control.
MAC	Mandatody Access Control.
HOr-BAC	Organisational hierarchy based access control.
ITSEC	Information technology security evaluation criteria.
NIST	National Institute of Standards and Technology.
OASIS	Organization for the Advancement of Structured Information Standards.
Or-BAC	Organization based access control.
PAP	Policy Administration Point.
PDP	Decision Point.
PEP	Policy Enforcement Point.
PIP	Policy Information Point.
PRP	Policy Retrieval Point.
RBAC	Role Based Access Control.

LISTE DES FIGURES

1 - Modèle de base du contrôle d'accès [12]	7
2 - Contrôle d'accès et autres services de sécurité [20]	9
3 - ACL de la matrice d'accès ci-dessus	14
4 - Attribution des permissions aux sujets à travers des rôles [23]	18
5 - RBAC standard [5]	19
6 - Hiérarchie RBAC [5]	19
7 - Modèle Or-BAC [23]	24
8 - Architecture de fonctionnement d'ABAC [10].	27
2.9 - Modèle de contrôle d'accès basé sur les rôles et amélioré par les attributs [16].	31
2.10 - Composants du parapheur électronique	39
3.11 - Affectation des valeurs d'attribut d'employé aux employé métiers . .	42
3.12 - Affectation des valeurs d'attribut de ressource aux ressources	43
3.13 - Structure organisationnelle d'une organisation	44
3.14 - Affectation des valeurs d'attributs de contexte aux contextes	45
3.15 - Diagramme de classes de la relation emploie	46
3.16 - Diagramme de classes de la relation nomme	46
3.17 - Diagramme de classes de la relation place-sous	47
3.18 - Diagramme de classes de la relation subordonne	47
3.19 - Diagramme de classes de la relation Utilise	48
3.20 - Diagramme de classes de la relation considère	48
3.21 - Diagramme de classes de la relation définit	49
3.22 - Diagramme de classes de la relation permission-opérationnelle . . .	49
3.23 - Diagramme de classes de la relation permission-Administrative . . .	50
3.24 - Diagramme de classes de la relation peut-suggérer	51
3.25 - Diagramme de classes de la relation peut-traiter	51
3.26 - Affectation des règles aux permissions de bas niveaux 1	51
3.27 - Affectation des règles aux permissions de bas niveaux 2	52
3.28 - Algorithme d'émission du parapheur électronique	54

3.29 - Algorithme de traitement du parapheur électronique	56
4.30 - Diagramme de flux de données [22]	59
4.31 - Model du langage XACML [22]	63
4.32 - Architecture de notre langage	65
4.33 - Architecture de notre langage	66

RÉSUMÉ

Le travail réalisé dans ce mémoire...

Mots clés: Contrôle d'accès, HOr-BAC, Attributs, Context-based access control, Intégration de HOr-BAC et d'ABAC.

ABSTRACT

Le modèle de contrôle d'accès basé sur les attributs est

Keywords: Access control, HOr-BAC, Attributes, Context-based access control, Integration of HOr-BAC and ABAC.

INTRODUCTION

CONTENTS

Contexte	1
Problématique	3
Objectifs	4
Plan de travail	4

Contexte

Avec le développement de l'informatique et d'Internet, les données sont de plus en plus stockées dans des serveurs distants afin de faciliter leur utilisation et de réduire les coûts du matériel nécessaire pour leur stockage. Cependant, avec internet, la sécurité des données n'est pas sans faille. Ainsi il est possible d'accéder de façon frauduleuse à des supports de stockage contenant des données venant de divers individus (personne ou entreprise). Au fil des années nous avons assisté à de nombreuses attaques telles que : les pertes de données, l'hameçonnage, le cheval de troie. Par exemple, Le 18 février 2021, le California Department of Motor Vehicles (DMV) a alerté les conducteurs californiens qu'ils avaient été victimes d'une fuite de données après que son prestataire de gestion de facturation, Automatic Funds Transfer Services, ait subi une attaque de ransomware. Dans l'optique de trouver une solution à ces problèmes, plusieurs approches telles que le contrôle d'accès ont été proposées. Le contrôle d'accès permet de déterminer les utilisateurs ou les programmes autorisés à accéder ou à modifier des données sécurisées dans le système. Traditionnellement, le contrôle d'accès (CA) est basé soit, sur l'identité d'un utilisateur demandant l'exécution d'une autorisation pour effectuer une opération (par exemple, lire) sur un objet (par exemple, un fichier), soit directement, ou encore par l'intermédiaire de types d'attribut prédéfinis tels que des rôles ou des groupes assignés à cet utilisateur. Les praticiens ont noté que cette approche

de contrôle d'accès est souvent lourde à gérer étant donné la nécessité d'associer les autorisations directement aux utilisateurs ou à leurs rôles ou groupes. En outre, les qualificatifs de l'utilisateur : identité, groupes et rôles sont souvent insuffisants pour exprimer les politiques de contrôle d'accès du monde réel. Une alternative consiste à accorder ou à refuser les requêtes des utilisateurs sur la base d'attributs arbitraires de l'utilisateur et d'attributs sélectionnés de l'objet, ainsi que de conditions d'environnement qui pourrait être reconnues globalement et plus pertinentes pour les politiques en cours. Cette approche est souvent appelée contrôle d'accès par attributs (ABAC) [24].

ABAC définit un paradigme de contrôle d'accès selon lequel les droits d'accès sont accordés aux utilisateurs grâce à l'utilisation des règles combinant les attributs [24]. Un attribut représente une information élémentaire qui caractérise un utilisateur, un objet, une action ou un environnement d'accès. Ce modèle permet une élaboration des politiques granulaires mais flexibles. En effet, la limite de l'élaboration des politiques réside essentiellement dans les attributs dont il faut tenir compte et dans les conditions que le langage informatique peut exprimer. ABAC permet au plus grand nombre de sujets d'accéder à la plus grande quantité de ressources sans obliger les administrateurs à spécifier les relations entre chaque sujet et objet. Toutefois, le fait qu'ABAC nécessite un fort besoin de provisioning et de maintenance des attributs fait de ce dernier un modèle difficile à administrer. Dans des situations d'urgence, ABAC peut refuser l'accès à un utilisateur légitime suite à l'absence d'une valeur d'attribut lors d'une décision d'accès. Un autre inconvénient d'ABAC est qu'il octroie des pouvoirs absolus à l'administrateur du système. Ainsi dans un système de gestion d'un centre hospitalier l'administrateur peut modifier le résultat d'examen de glycémie d'un patient en indiquant que cet examen est négatif pourtant il ne l'est pas, et l'infirmière qui est en charge d'administrer une perfusion à ce patient afin de baisser sa fièvre lui administre une perfusion contenant du glucose conformément au résultat de son examen. Ce qui entraîne par conséquent la mort du patient. ainsi nous remarquons que, l'administrateur d'un système peut manipuler les données du système de façon frauduleuse et en toute liberté. C'est la raison pour laquelle un nouveau modèle a vu le jour : Il s'agit du modèle HOr-BAC qui consiste à gérer le contrôle d'accès et à surveiller toutes les opérations de bases dans un système d'information.

HOr-BAC se base sur la structure organisationnelle d'une organisation afin de permettre la spécification des politiques de sécurité contextuelle relative aux permissions. Dans HOr-BAC, les permissions sont attribuées aux unités organisationnelles et les employés affectés à ces unités obtiennent les permissions qui les sont

attribuées. Ce modèle permet de gérer le contrôle d'accès et de surveiller toutes les opérations de bases dans un système d'information, empêchant ainsi la création des entités virtuelles au sein du système. HOr-BAC permet de représenter la relation hiérarchique qui existe entre les différentes unités organisationnelles d'une organisation. Par exemple dans une entreprise l'unité organisationnelle RH (Ressources Humaines) est subordonnée par l'unité organisationnelle direction générale. En plus de cela il introduit la notion de parapheur électronique qui est un processus de traitement automatique sécurisé permettant de contrôler et de valider les activités du personnel métier y compris le super-utilisateur conformément à la structure organisationnelle de l'entreprise [23].

Problématique

Bien que HOr-BAC s'appuie sur la structure organisationnelle d'une entreprise et empêche la création d'entités virtuelles dans un système d'information, cependant il a des lacunes. En effet, dans HOr-BAC, les politiques de contrôle ne peuvent être définies que sur la base de l'unité organisationnelle à laquelle appartient un employé, ce qui limite ainsi la flexibilité du contrôle d'accès. Ceci est fréquemment rencontré dans le domaine du Cloud Computing où on assiste à une explosion des unités organisationnelles et de leurs autorisations lorsqu'on essaie de définir des politiques de contrôle d'accès qui se basent sur des caractéristiques d'entités autres que celles prédéfinies dans HOr-BAC. Par conséquent il ne permet que de définir des politiques de contrôle d'accès à gros grain, c'est-à-dire des politiques qui ne sont définies que sur la base des unités organisationnelles. Par exemple dans HOr-BAC, il est difficile d'exprimer la politique de contrôle d'accès suivante : « dans un système de gestion d'un centre hospitalier l'agent d'assurance d'un patient ne peut voir que les informations relatives à la facture des soins reçus par son patient dans son dossier médicale. Il n'a pas besoin de connaître des informations relatives à son état de santé »

Ainsi, le problème qui se pose est le suivant : comment réaliser l'approche orientée attribut du modèle HOr-BAC afin qu'il permet un contrôle d'accès fin et flexible ?

Objectifs

L'objectif principal de notre travail est de proposer un modèle de contrôle d'accès permettant un contrôle d'accès à grain fin et flexible. En effet nous partons de l'hypothèse selon laquelle en combinant une spécification de politique flexible et fine granulaire et une capacité de prise de décision dynamique qui font la renommée d'ABAC avec la facilité d'administration et le contrôle des actions effectuées par un individu (administrateur ou employé) au sein d'un système d'information, ce qui fait la renommée de HOR-BAC nous obtiendrons un modèle de contrôle d'accès flexible et fine granulaire.

Comme objectifs secondaires nous avons :

- Proposer un langage de politiques qui nous permettra d'interroger les politiques de sécurité de notre modèle.
- Il doit être applicable aux données Cloud;
- Implémenter notre approche et notre langage dans un système existant.

Plan de travail

Notre travail sera structuré comme suite:

Chapitre 1: Généralité sur la sécurité des systèmes d'information. Dans ce chapitre, nous présenterons les concepts de base de la sécurité des systèmes d'information;

Chapitre 2: Etat de l'art sur les politiques et modèles de contrôles d'accès. Il est destiné à une présentation générale des travaux existants pour la résolution du problème de contrôle d'accès. Ici, nous présenterons les différentes politiques et modèles de contrôles d'accès (principes de fonctionnement, avantages, inconvénients)

Chapitre 3 : Approche orientée attributs pour le contrôle d'accès basé sur la hiérarchie organisationnelle. Dans ce chapitre, nous allons de prime abord présenter les différents concepts que nous utiliserons pour cette approche et par la suite décrire de manière précise le fonctionnement du parapheur électronique dans une structure organisationnelle.

Chapitre 4 : Proposition d'un langage permettant d'exprimer les politiques d'accès basé sur la hiérarchie organisationnelle et les attributs. Ici nous proposerons tout d'abord un langage qui nous permettra d'exprimer les politiques de sécurité de notre Modèle, puis nous présenterons une application de notre modèle et dans un système existant.

I

CHAPTER

GÉNÉRALITÉ SUR LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

TABLE DES

MATIÈRES

I.1 - Sécurité des systèmes d'information	5
I.2 - Contrôle d'accès	7

Dans ce chapitre nous fournirons la base terminologique nécessaire à la compréhension de ce travail. Nous présenterons une brève introduction à la sécurité des systèmes d'information et présenterons le concept de contrôle d'accès.

I.1. Sécurité des systèmes d'information

I.1.1. Définition de la sécurité

Le terme sécurité correspond au mot anglais « Security » et traduit la capacité du système informatique à résister à des agressions externes physiques (incendie, inondation, bombes.) ou logiques (erreurs de saisie, intrusions, piratages, logique malicieuse.). C'est généralement le sens choisi par les spécialistes de l'audit de sécurité, lorsqu'ils doivent, pour une entreprise donnée, évaluer les risques liés à l'informatique. Les ITSEC [11] définissent la sécurité comme la combinaison de trois propriétés : la confidentialité, l'intégrité et la disponibilité de l'information. L'information représente en plus des données et des programmes, les traitements effectués sur les informations. Ici, la sécurité, implique d'empêcher la réalisation d'opérations illégitimes contribuant à mettre en défaut les propriétés de confidentialité, d'intégrité et de disponibilité, mais aussi de garantir la possibilité de réaliser les opérations légitimes dans le système. Assurer la sécurité du système revient à

assurer que les propriétés retenues sont vérifiées, et garantir la non-occurrence de défaillances vis-à-vis de ces propriétés [23].

I.1.2. Confidentialité

La confidentialité est la propriété d'une information à ne pas être révélée à des utilisateurs non autorisés à la connaître. Le système informatique doit empêcher les utilisateurs de lire une information confidentielle (sauf s'ils y sont autorisés), et d'empêcher les utilisateurs autorisés à lire une information et de la divulguer à d'autres utilisateurs (sauf autorisation). Assurer la confidentialité d'un système est une tâche complexe. Il faut analyser tous les chemins qu'une information particulière peut prendre dans le système pour s'assurer qu'ils sont sécurisés. Il faut également prendre en compte les connaissances qu'un ou plusieurs utilisateurs peuvent déduire à partir des informations acquises. Il faut donc contrôler non seulement les informations présentes dans le système, mais aussi les liens logiques qui peuvent les relier entre elles ou à des informations publiques. Les attaques contre la confidentialité consistent à essayer d'obtenir des informations qui doivent être protégées selon la politique de sécurité, en dépit des moyens de protection et des règles de sécurité [23].

I.1.3. Intégrité

L'intégrité est la propriété d'une information à ne pas être altérée. Le système informatique doit [23] :

- Empêcher une modification de l'information par des utilisateurs non autorisés ou une modification incorrecte par des utilisateurs autorisés;
- Faire en sorte qu'aucun utilisateur ne puisse empêcher la modification légitime de l'information. Par exemple, empêcher la mise à jour périodique d'un compteur de temps constituerait une atteinte à l'intégrité;
- Assurer que toute modification de donnée est approuvée et que chaque programme se comporte de manière correcte (conformément aux fonctions qu'il est censé remplir, y compris dans ses interactions avec les autres processus.);
- Assurer qu'aucune information ne peut être modifiée par des intermédiaires, que cette altération soit intentionnelle (par exemple, un utilisateur intervient pour modifier une communication entre deux autres utilisateurs.) ou accidentelle (une donnée modifiée lorsqu'elle est communiquée via un support de communication non-fiable).

I.1.4. Disponibilité

La disponibilité est la propriété d'une information d'être accessible lorsqu'un utilisateur autorisé en a besoin. Le système informatique doit [23]:

- Fournir l'accès à l'information pour que les utilisateurs autorisés puissent la lire ou la modifier ;
- Faire en sorte qu'aucun utilisateur ne puisse empêcher les utilisateurs autorisés d'accéder à l'information.

Dans le souci de garantir la non-violation de ces propriétés de sécurité, plusieurs techniques de sécurité à l'instar du contrôle d'accès ont été proposées dans la section suivante nous présenterons la notion de contrôle d'accès.

I.2. Contrôle d'accès

I.2.1. Définition du contrôle d'accès

Le contrôle d'accès est une technique de sécurité qui vise à limiter ce qu'un utilisateur ou les programmes s'exécutant au nom de cet utilisateur sont autorisés à faire dans le système. Ainsi le contrôle d'accès permet d'empêcher toute activité susceptible de conduire à une violation de la sécurité. Son but est de limiter les actions ou les opérations qu'un utilisateur légitime du système informatique peut effectuer. Le modèle de base de toutes les politiques de contrôle d'accès est montré sur la figure 1:

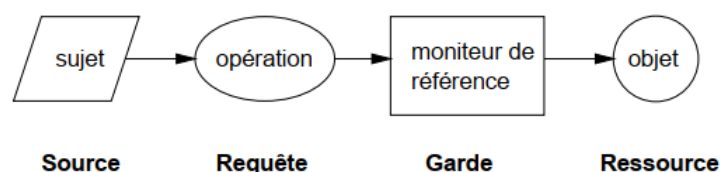


Figure 1 – *Modèle de base du contrôle d'accès* [12]

La figure montre un sujet qui souhaite faire une opération sur un objet. Le système transforme l'opération en une requête qu'il passe au moniteur de référence (qui est un médiateur incontournable dans toutes les relations entre sujets et objets. Le moniteur de référence est responsable de l'autorisation de l'accès à un objet par un sujet.) qui contrôle l'accès aux ressources. Si le sujet est autorisé à accéder à l'objet selon la politique de sécurité en vigueur, l'accès à l'objet va être accordé et l'opération peut se dérouler normalement [12].

I.2.2. Le contrôle d'accès et les autres services

Dans un système informatique, le contrôle d'accès repose sur d'autres services de sécurité et coexiste avec eux.

- **Contrôle d'accès et l'administrateur de sécurité:** Le contrôle d'accès vise à limiter l'activité des utilisateurs légitimes. Il est appliqué par le moniteur de référence qui sert d'intermédiaire à chaque tentative d'accès d'un utilisateur (ou un programme s'exécutant au nom de cet utilisateur) aux objets du système. Le moniteur de référence consulte une base de données d'autorisations afin de déterminer si un utilisateur tentant d'effectuer une opération est réellement autorisé à effectuer cette opération. Les autorisations de cette base de données sont gérées et administrées par l'administrateur de sécurité. L'administrateur définit ces autorisations sur la base de la politique de sécurité de l'organisation. Les utilisateurs peuvent également modifier une partie de cette base de données des autorisations, par exemple, pour définir les autorisations pour leurs fichiers personnels[20].
- **Le contrôle d'accès et l'authentification:** Il est important de faire une distinction entre l'authentification et le contrôle d'accès. L'établissement correct de l'identité de l'utilisateur relève de la responsabilité de l'authentification. Le contrôle d'accès suppose que l'authentification de l'utilisateur a été vérifiée avec succès avant l'application du contrôle d'accès via le moniteur de référence. L'efficacité du contrôle d'accès repose sur une identification correcte de l'utilisateur et sur l'exactitude des autorisations accordées au moniteur de référence [20].
- **contrôle d'accès et l'audit:** Il est important de comprendre que le contrôle d'accès n'est pas une solution complète pour sécuriser le système. Il doit être couplé à l'audit. L'audit permet de surveiller et d'enregistrer les activités pertinentes dans le système. Les contrôles d'audit concernent une analyse à posteriori de toutes les demandes et activités des utilisateurs dans le système. L'audit nécessite l'enregistrement de toutes les demandes et activités des utilisateurs pour une analyse ultérieure. Les contrôles d'audit sont utiles à la fois comme moyen de dissuasion et comme moyen d'analyser le comportement des utilisateurs dans l'utilisation du système afin de découvrir d'éventuelles tentatives ou violation réelles. En outre, l'audit peut être utilisé pour déterminer les éventuelles failles du système de sécurité. En fin il est essentiel pour garantir que les utilisateurs autorisés n'abusent pas de leurs privilèges. En d'autres termes, pour les tenir responsables de leurs actions [20].

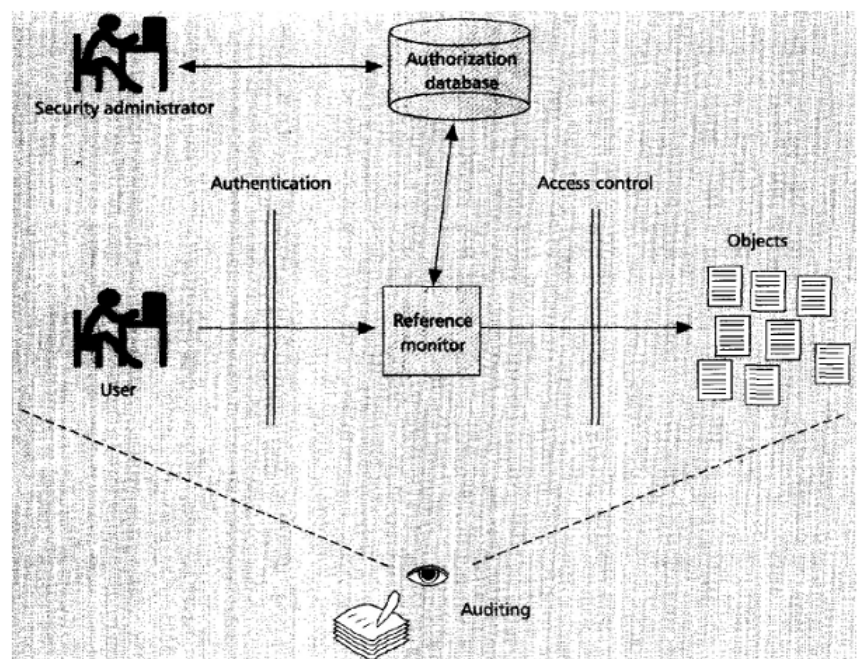


Figure 2 – Contrôle d'accès et autres services de sécurité [20]

I.2.3. Politique de sécurité

Dans un système informatique, l'autorisation a pour but de ne permettre que les actions légitimes, d'empêcher qu'un utilisateur puisse exécuter des opérations qui ne lui sont pas permises. Pour définir quelles sont les opérations autorisées et celles qui sont interdites, il faut établir une politique de sécurité ou « doctrine de sécurité » [23]. Dans les systèmes de contrôle d'accès, une distinction est généralement faite entre les politiques et les mécanismes de sécurité. Les politiques sont des directives de haut niveau qui déterminent comment les accès sont contrôlés et les décisions d'accès déterminées. Alors que les mécanismes sont des fonctions logicielles et matérielles de bas niveau qui peuvent être configurées pour mettre en œuvre une politique. Au fil des années, il a été développé par des chercheurs et praticiens de la sécurité des mécanismes de sécurité qui sont indépendants des politiques de sécurité. Les politiques de contrôle d'accès ne sont pas nécessairement exclusives. Différents politiques peuvent être combinés pour fournir un système de protection plus adapté. Lorsque les politiques sont combinées, seule l'intersection de leur accès est autorisée. Une telle combinaison de politique est relativement simple tant qu'il n'y a pas de conflits entre une politique qui affirme qu'un accès particulier doit être autorisé et une autre qui l'interdit. Ces conflits sont conciliés par des négociations à un niveau de gestion approprié [20].

I.2.4. Conclusion

En conclusion, il était question pour nous dans ce chapitre de présenter les concepts de base de notre travail.

II

CHAPTER

ÉTAT DE L'ART SUR LES POLITIQUES ET MODÈLES DE CONTRÔLE D'ACCÈS

SOMMAIRE

II.1 - Introduction	11
II.2 - Politiques et modèles de contrôle d'accès discrétionnaires (DAC) . .	12
II.3 - Politiques et modèles de contrôle d'accès obligatoire (MAC)	15
II.4 - Politiques et modèles de contrôle d'accès à base de rôles (RBAC) . .	17
II.5 - Politiques et modèles de contrôle d'accès basés sur les attributs (ABAC)	24
II.6 - Politiques et modèles de contrôle d'accès basés sur les rôles améliorés par les attributs(AERBAC)	29
2.7 - Politiques et modèles de contrôle d'accès basés sur les rôles et les attributs(ARBAC)	33
2.8 - Politiques et modèles de contrôle d'accès basés sur la hiérarchie or- ganisationnelle (HOr-RBAC)	35
2.9 - Conclusion	40

II.1. Introduction

La sécurité des systèmes d'information a pour but de garantir trois propriétés fondamentales appelées objectifs de sécurité : la confidentialité, l'intégrité et la disponibilité. La confidentialité assure que seules les personnes autorisées peuvent accéder à une information. L'intégrité, quant à elle, vise à empêcher toute altération ou destruction d'une information par des personnes malveillantes. La disponibilité garantit que l'information est accessible aux personnes autorisées quand elles en ont besoin. De ce fait, plusieurs méthodes de sécurité telles que le contrôle d'accès ont été proposées. Le contrôle d'accès a pour but de limiter les actions ou les

opérations qu'un utilisateur légitime du système informatique peut effectuer [20]. Faisant l'objet de nombreuses recherches plusieurs approches de contrôle d'accès plus connues sous le nom de modèle de contrôle d'accès ont été implémentées. Parmi ces modèles, nous pouvons citer : DAC (Discretionary Access Control), MAC (Mandatory Access Control), RBAC (Role Based Access Control), Or-BAC (Organization Based Access Control), ABAC (Attribute Based Access Control) et HOr-BAC (Hierarchy Organization Based Access Control).

Dans ce chapitre, il sera question pour nous de présenter les différents modèles de contrôle d'accès ainsi que leurs avantages et leurs inconvénients.

II.2. Politiques et modèles de contrôle d'accès discrétionnaires (DAC)

Les politiques de protection discrétionnaires régissent l'accès des utilisateurs à l'information sur la base de l'identité de l'utilisateur et des autorisations qui spécifient pour chaque utilisateur (ou groupe d'utilisateurs) et chaque objet du système, les modes d'accès auxquels l'utilisateur est autorisé à effectuer sur l'objet [20]. Chaque demande de l'utilisateur pour accéder à un objet est vérifiée par rapport aux autorisations spécifiées. S'il existe une autorisation statuant que l'utilisateur peut accéder à l'objet dans le mode spécifique, l'accès est accordé, sinon il est refusé. Le guide NCSC (A Guide To Understanding Discretionary Access Control in Trusted Systems) [13], indique que la base du DAC réside sur le fait qu'un utilisateur individuel, ou un programme fonctionnant au nom de cet utilisateur est autorisé à spécifier explicitement les types d'accès que les autres utilisateurs (ou les programmes s'exécutant en leurs noms) peuvent avoir sur les informations qui sont sous le contrôle de l'utilisateur.

Les politiques de contrôle d'accès se basent sur le modèle de matrice d'accès pour la définition des autorisations qu'ont les utilisateurs sur les objets. La flexibilité des politiques discrétionnaires les rend adaptés à une variété de systèmes et d'application. Pour ces raisons, elles ont été largement utilisées dans une variété d'implémentations, en particulier dans les environnements commerciaux et industriels.

II.2.1. Modèle de matrice d'accès

La matrice d'accès est un modèle conceptuel qui spécifie les droits que chaque sujet possède pour chaque objet. Elle définit trois types d'entités [17] :

- Les objets protégés : ce sont les entités ou ressources auxquelles on veut accéder;
- Sujet : ce sont des entités actives qui accèdent aux objets;
- Les droits d'accès : qui associent les sujets aux objets protégés, ceci en spécifiant les opérations que les sujets peuvent effectuer sur les objets.

Dans cette matrice, les lignes représentent les sujets, les colonnes représentent les objets et les cellules représentent des droits qu'ont les sujets sur les objets. La tâche du contrôle d'accès est d'assurer que seules les opérations autorisées par la matrice d'accès sont exécutées. Ceci est réalisé au moyen d'un moniteur de référence qui est responsable de la médiation de toutes les opérations tentées par les sujets sur les objets. De façon générale, le propriétaire d'un fichier est autorisé à accorder aux autres utilisateurs l'accès ou non au fichier [20]. Le tableau 1 montre un exemple de matrice d'accès. Dans cette matrice, nous voyons que Bob détient des droits de lecture (R) et d'écriture (W) sur le fichier 1 ; des droits de propriété (Own), de lecture et d'écriture sur le fichier 3 et le droit d'exécution sur le fichier 4 et n'a aucun droit sur le fichier 2.

Table I – Exemple de matrice d'accès.

	fichier 1	fichier 2	fichier 3	fichier 4
Bod	R, W		Own, R, W	X
Alice	R	Own, R, W		R
John	Own, R, W		R, W	

II.2.2. Implémentation des politiques discrétionnaires

Dans de grands systèmes, la matrice d'accès est énorme en taille et la plupart de ses cellules vides, ce qui rend ainsi son implémentation rare. De ce fait en pratique, il est développé plusieurs approches de la matrice d'accès.

II.2.2.1. Liste de contrôle d'accès

Une approche populaire de l'implémentation de la matrice d'accès est l'utilisation des listes de contrôle d'accès. Ici, chaque objet est associé à une liste de contrôle d'accès (ACL) qui indique pour chaque sujet du système les accès que le sujet est autorisé à exécuter sur l'objet. Cette approche correspond à un stockage de la matrice par colonnes [20]. La figure 3 ci-dessous illustre les ACL de la matrice d'accès ci-dessus. En consultant l'ACL d'un objet, il est facile de déterminer quels modes d'accès sont actuellement autorisés pour cet objet. Il est également facile de supprimer tous les accès à un objet en remplaçant l'ACL existante par une ACL

vide. Par contre, il est difficile de déterminer tous les accès d'un sujet dans un système basé sur les ACL. Car cela nécessite un examen de la liste de contrôle d'accès de chaque objet du système pour faire le contrôle d'accès d'un sujet. De manière similaire, si tous les accès d'un sujet ont besoin d'être révoqués, toutes les ACL vont être visitées une par une.

De nombreux systèmes permettent aux noms de groupe d'apparaître dans les listes de contrôle d'accès. Les systèmes d'exploitation tels que l'UNIX et VMS implémentent une forme abrégée d'ACL dans laquelle un petit nombre de noms de groupe, souvent un ou deux, peuvent apparaître dans l'ACL [20].

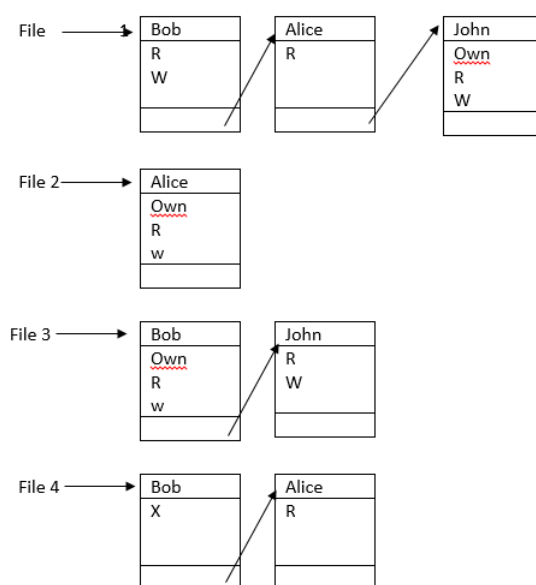


Figure 3 – ACL de la matrice d'accès ci-dessus

II.2.2.2. Les capacités

Ici, chaque sujet est associé à une liste de capacité qui indique, pour chaque objet du système quels accès le sujet est autorisé d'exécuter sur l'objet. Cette approche correspond au stockage de la matrice d'accès par lignes. La liste des capacités permet facilement de voir tous les accès qu'un sujet est autorisé à effectuer, ceci par un simple examen de la liste des capacités. Toutefois, déterminer tous les sujets qui peuvent accéder à un objet revient à un examen de la liste des capacités de chaque sujet. Cette approche, contrairement aux ACL n'a pas connue de succès.

II.2.3. Vulnérabilité des politiques discrétionnaire

Les politiques discrétionnaires ne font pas de distinction entre les utilisateurs et les sujets. De façon générale, les utilisateurs sont des entités passives pour lesquelles des autorisations peuvent être spécifiées et qui peuvent se connecter au

système. Une fois connectés au système, les utilisateurs créent des processus (sujets) qui s'exécutent en leur nom et, par conséquent, soumettent des demandes au système. Elles évaluent ainsi toutes les demandes des processus exécutés au nom d'un utilisateur par rapport aux autorisations de l'utilisateur. Cependant une analyse plus précise pour le problème de contrôle d'accès montre une utilité de séparer les utilisateurs des sujets. Un autre inconvénient des politiques discrétionnaires est qu'elles ne fournissent pas une réelle assurance sur les flux d'information dans le système. Par conséquent, il est facile de contourner les restrictions d'accès énoncés dans les autorisations. Par exemple, un utilisateur qui a le droit de lire sur un fichier peut le transmettre à d'autres utilisateurs non autorisés pour le lire sans que le propriétaire du fichier ne soit au courant. Afin de palier à cet inconvénient, les politiques obligatoires ont vu le jour.

II.3. Politiques et modèles de contrôle d'accès obligatoire (MAC)

Contrairement aux politiques discrétionnaires, les politiques obligatoires centralisent l'autorité d'administration des droits à un seul utilisateur généralement appelé super-utilisateur [15]. Les politiques obligatoires régissent l'accès sur la base de la classification des sujets et des objets dans le système. Les objets sont des entités passives stockant des informations. Les sujets sont des entités actives qui demandent l'accès aux objets. Il est bon à savoir ici que les politiques obligatoires font une distinction entre les utilisateurs et les sujets. Les utilisateurs sont des êtres humains qui peuvent accéder au système, tandis que les sujets sont des processus (programmes en cours d'exécution) opérant pour le compte des utilisateurs. Cette distinction permet à la politique de contrôler les accès indirects (fuites ou modification) provoqués par l'exécution des processus [18]. Ce modèle associe à chaque sujet et à chaque objet du système un niveau de sécurité. Le niveau de sécurité associé à un objet reflète la sensibilité de l'information contenue dans l'objet, c'est-à-dire le dommage potentiel qui pourrait résulter de la divulgation non autorisée de ces informations. Le niveau de sécurité associé à un utilisateur, également appelé autorisation, reflète la confiance de l'utilisateur dans le fait de ne pas divulguer d'informations sensibles à des utilisateurs non autorisés à les voir. De façon simple, un niveau de sécurité est un élément d'un ensemble hiérarchique ordonné. Les niveaux de sécurité les plus connus sont : très secret (TS), secret (S), confidentiel (C), non classifié (U). Chaque niveau de sécurité se domine lui-même

et tous les autres en dessous de lui dans cette hiérarchie, exemple : TS>S>C>U [20].

L'accès à un objet par un sujet n'est accordé que si une certaine relation est satisfaite entre les niveaux de sécurité associés aux deux en particulier, les deux principes suivants doivent être respectés [20] :

- **Read down** : l'autorisation d'un sujet doit dominer le niveau de sécurité de l'objet en cours de lecture;
- **Write up** : l'autorisation d'un sujet doit être dominée par le niveau de sécurité de l'objet en cours d'écriture.

La satisfaction de ces principes empêche les informations connues dans les objets de haut niveau (c'est-à-dire plus sensibles) de circuler vers les objets de niveaux inférieurs. Dans un tel système, les informations ne peuvent circuler que vers le haut ou au sein de la même classe de sécurité. Il est important de contrôler à la fois les opérations de lecture et d'écriture, car les deux peuvent être utilisées de manière inappropriée pour divulguer des informations. Les politiques basées sur ces deux principes permettent de fait la différence entre les utilisateurs et les sujets. En effet à travers le Read down les sujets sont empêchés de lire les informations contenues dans les objets de classe supérieure ; et à travers le principe d'écriture ascendante, elles empêchent les logiciels malveillants de divulguer des secrets vers le bas : de la classe supérieure à la classe inférieure. Ici, on fait confiance aux utilisateurs pour ne pas divulguer de telles informations, mais les programmes qu'ils exécutent ne méritent pas le même degré de confiance. Cette règle empêche également les utilisateurs de divulguer des informations du haut vers le bas.

Le contrôle d'accès obligatoire peut également être appliqué pour la protection de l'intégrité des informations. Ici, le niveau d'intégrité associé à un objet reflète le degré de confiance que l'on peut accorder aux informations stockées dans l'objet et des dommages potentiels qui pourraient résulter d'une modification non autorisée de l'information. Le niveau d'intégrité associé à un utilisateur reflète la capacité de l'utilisateur à insérer, modifier ou supprimer des données et des programmes à ce niveau. Des principes similaires à ceux, énoncés pour le secret doivent être respectés comme suite [20]. :

- **Read up** : le niveau d'intégrité d'un sujet doit être dominé par le niveau d'intégrité de l'objet en cours de lecture;
- **Write down** : le niveau d'intégrité d'un objet doit dominer celui de l'utilisateur en cours d'écriture.

Le respect de ces principes garantit l'intégrité en empêchant les informations stockées dans les objets de niveaux inférieurs de circuler vers les objets de niveaux supérieurs. Le contrôle de flux d'information n'est qu'un aspect de la réalisation de l'intégrité.

Les politiques mandataires sont implémentées dans les systèmes d'exploitation tels que Vista et Linux et elles sont également utilisées dans les bases de données.

II.3.1. Vulnérabilité des politiques Obligatoire

Bien que les politiques obligatoires offrent une protection contre les fautes d'informations indirectes, elles ne garantissent pas le secret complet de l'information. En effet, les politiques obligatoires de secret ne contrôlent que les canaux manifestent d'information (c'est-à-dire les canaux légitimes). Elles restent toujours vulnérables aux canaux secrets. Il s'agit des canaux qui ne sont pas destinés à la communication normale, mais qui peuvent néanmoins être exploités pour déduire des informations. Les politiques mandataires sont lentes à administrer et sont également rigides dans leur fonctionnement. Les politiques obligatoires ne sont pas adaptés à des organisations de grande taille.

II.4. Politiques et modèles de contrôle d'accès à base de rôles (RBAC)

II.4.1. Politiques et modèles de contrôle d'accès basés sur les rôles (RBAC)

Les politiques discrétionnaires et obligatoires sont difficiles à administrer pour les organisations de grande taille. En effet, dans ces politiques, il faut à chaque fois enregistrer tous les utilisateurs et définir les droits d'accès aux objets du système pour chaque sujet. Les politiques basées sur les rôles (RBAC) viennent résoudre ce problème en accordant les droits d'accès aux rôles plutôt qu'aux utilisateurs individuellement [19]. Les politiques basées sur les rôles régulent l'accès des utilisateurs aux informations sur la base des activités qu'ils exécutent dans le système. Ces politiques nécessitent l'identification des rôles dans le système. Un rôle peut être défini comme un ensemble d'actions et de responsabilités associées à une activité professionnelle particulière. Ensuite, au lieu de spécifier tous les accès que chaque utilisateur est autorisé à exécuter, les autorisations d'accès aux objets sont spécifiées pour les rôles. Les utilisateurs reçoivent des autorisations pour adopter des rôles. Une étude récente du NIST [7] confirme que les rôles sont une approche

utiles pour de nombreuses organisations commerciales et gouvernementales. Avec RBAC, les administrateurs du système créent des rôles en fonction des fonctions exercées dans une entreprise ou une organisation, donnent les permissions (autorisations) d'accès à ces rôles, puis affectent les utilisateurs aux rôles en fonctions de leurs responsabilités et qualifications professionnelles spécifiques [23].



Figure 4 – Attribution des permissions aux sujets à travers des rôles [23]

II.4.1.1. Composants principaux de RBAC

RBAC est généralement composé des quatre éléments suivants [5] :

- **User ou Utilisateur:** un utilisateur est défini comme un être humain, une machine, un réseau, un processus ou un agent autonome intelligent. Cette définition couvre différents domaines de sécurité et des domaines d'application, par exemple : les bases de données;
- **Rôle:** un rôle est généralement une fonction qui dans le contexte d'une organisation avec une sémantique associée concernant son autorité et sa responsabilité;
- **Permission:** c'est un mode d'accès qui peut être exercé sur des objets d'un système. Les objets et les modes d'accès dépendent du domaine. Par exemple, dans le cas d'une base de données, l'ensemble d'objets comprend des tables, des colonnes et des lignes, et l'ensemble des modes d'accès comprennent des opérations d'insertion, de suppression et de mise à jour;
- **Session:** c'est une instance particulière d'une connexion d'un utilisateur au système et définit le sous-système de rôles activés. A chaque instant, différentes sessions pour le même utilisateur peuvent être activées. Lorsque l'utilisateur se connecte au système, il établit une session et pendant celle-ci, il peut demander d'activer un sous-ensemble du rôle qu'il est autorisé à jouer. L'utilisateur obtient toutes les permissions associées aux rôles qu'il a activés dans sa session.

Au fil des années, différentes variantes du modèle RBAC ont été mises sur pieds afin d'introduire de nouveaux concepts tels que la hiérarchie des rôles et la séparation des tâches.

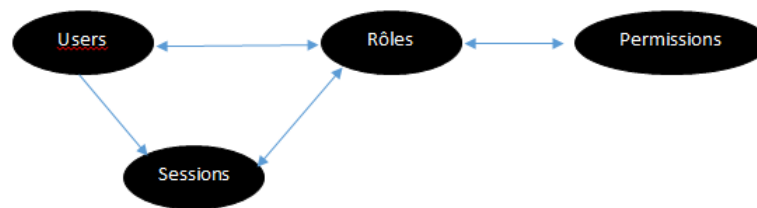


Figure 5 – RBAC standard [5]

II.4.1.2. Hiérarchie des rôles

La hiérarchie des rôles représente une amélioration importante du RBAC standard, car elle constitue un moyen naturel de structurer les rôles pour refléter la structure de cette organisation. A cette fin, une relation d'ordre partiel sur les rôles est introduite, appelée hiérarchie des rôles [23]. La hiérarchie des rôles, définie comme $RHC \text{ Rôles} \times \text{Rôles}$, identifie l'ensemble des paires de rôles $\langle r_i, r_j \rangle$ telles que le rôle r_i hérite du rôle r_j . Dans la littérature, il existe deux interprétations différentes de l'héritage :

- La hiérarchie générale : elle permet de supporter le concept de hiérarchie multiple qui fournit la capacité d'héritage des permissions de deux rôles ou plus ;
- La hiérarchie limitée : elle se base sur les mêmes principes que la hiérarchie générale, cependant elle ne supporte pas l'héritage multiple.

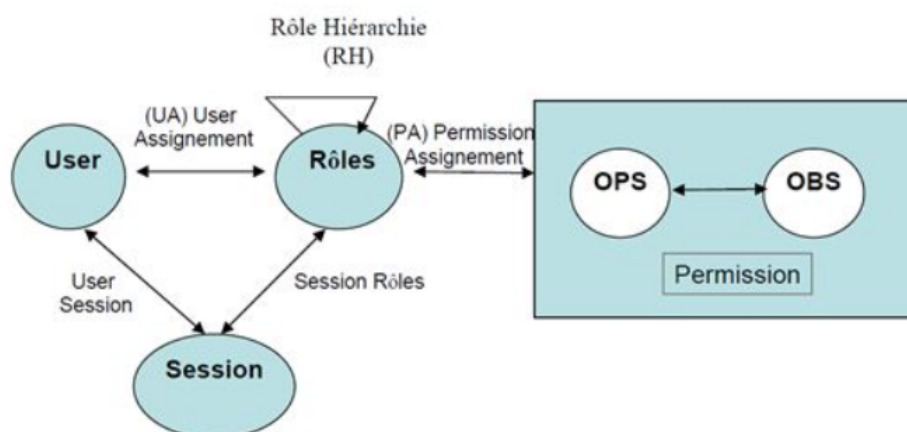


Figure 6 – Hiérarchie RBAC [5]

II.4.1.3. Notion de séparation des tâches

La notion de séparation de tâches a été ajoutée dans le modèle RBAC. Celle-ci stipule qu'aucun utilisateur ne possède assez de privilèges pour abuser seul du

système, afin d'assurer que les fraudes et les erreurs majeures ne peuvent avoir lieu que par une collaboration préméditée de plusieurs utilisateurs. Au sein des organisations, ceci se traduit par le fait d'empêcher l'utilisateur de jouer des rôles conflictuels ou aussi de restreindre le nombre de rôles associés aux utilisateurs d'une façon statique ou dynamique. A cet effet, ce modèle distingue deux types de séparation de tâches [19] : la séparation statique des devoirs (SSD) vise à empêcher le conflit d'intérêts qui survient lorsque les rôles en conflits sont affectés à un même utilisateur et la séparation dynamique des devoirs (DSD) qui impose le contrôle dynamiquement par une contrainte sur les rôles lors de l'activation d'une session donnée.

II.4.1.4. Avantages du modèle RBAC

L'approche d'un contrôle d'accès basée sur les rôles présente des avantages incontestables sur différents plans [23]:

- **Gestion des autorisations:** les politiques basées sur les rôles bénéficient d'une indépendance logique dans la spécification des autorisations des utilisateurs en divisant cette tâche en deux parties. L'une attribue les utilisateurs aux rôles et l'autre attribue les droits d'accès aux objets aux rôles. Cela simplifie grandement la gestion de la sécurité. Par exemple, supposons qu'un utilisateur change de responsabilités, disons en raison d'une promotion;
- **Hiérarchie des rôles:** dans de nombreuses applications, il existe une hiérarchie naturelle des rôles, fondée sur les principes familiers de généralisation et de spécification. Exemple : les rôles ingénieurs logiciels et ingénieurs matériels sont des spécifications du rôle d'ingénieur. Un utilisateur affecté à l'un de ces rôles héritera également des privilèges et autorisations attribués aux rôles plus généraux d'ingénieur ; les rôles hiérarchiques simplifient encore la gestion des autorisations;
- ce modèle de contrôle d'accès est adapté pour les organisations qui ont un niveau de turn-over élevé.

Le modèle RBAC a été implémenté dans plusieurs systèmes parmi lesquels le système Oracle, Dresdner Bank une banque européenne internationale comportant 50 659 employés et 1459 branches dans le monde dont la branche principale est située en Allemagne.

II.4.1.5. Inconvénient du modèle RBAC

RBAC a été critiqué dans [2] pour les raisons suivantes :

- Le concept de hiérarchie de rôle est quelque peu ambigu. En général, la hiérarchie des rôles ne correspond pas tout à fait à la hiérarchie organisationnelle. Par exemple, le directeur de l'hôpital a un rôle administratif supérieur au rôle de médecin. Pour autant, un directeur de l'hôpital n'est pas nécessairement un médecin, ainsi, il n'est pas faisable d'accorder au directeur les permissions du médecin;
- L'impossibilité d'exprimer des règles contextuelles relatives aux permissions, aux interdictions, aux obligations et aux recommandations et notamment des permissions qui dépendent du contexte. Par conséquent, il serait difficile de spécifier qu'un médecin n'a la permission d'accéder au dossier médical d'un patient que si ce dernier est son patient. ;
- D'après [21] et [16] RBAC souffre d'un problème d'explosion de rôle et de permission ce qui le rend moins flexible.
- RBAC ne permet pas d'effectuer un contrôle à grain fin des ressources . cela est dû au fait qu'il n'accorde ou refuse l'accès aux ressources protégées du système que sur la base de l'attribut Rôle de l'utilisateur.

II.4.2. Politiques et modèles de contrôle d'accès basé sur les organisations (Or-BAC)

Aucun des modèles cités plus haut n'est pleinement satisfaisant pour modéliser des politiques de sécurité qui ne se limitent pas qu'aux permissions statiques, mais incluent également des règles contextuelles liées aux permissions, interdictions, recommandations et obligations. D'où la nécessité d'Or-BAC, qui est un modèle de contrôle d'accès basé sur l'organisation [14], permettant la spécification des politiques de sécurité contextuelles relatives aux permissions, interdictions, recommandation et aux obligations. Une organisation représente l'ensemble des rôles, des activités et des vues qui représentent les abstractions respectives des utilisateurs, des opérations et des objets par rapport à une organisation donnée. Par exemple, un utilisateur est lié à un ensemble de rôles pour une organisation donnée. Il peut être affecté à d'autres rôles pour une autre organisation. Or-BAC définit des relations ternaires entre les organisations, les sujets et les rôles : un sujet joue un rôle dans une organisation [15]. Ce qui veut dire que l'utilisateur ayant plusieurs rôles peut activer tous les rôles, soit un sous-ensemble de ses rôles dans n'importe quelle équipe à laquelle il participe. Dans la pratique, même si un util-

isateur possède plusieurs rôles, il n'a pas forcément le droit de les jouer dans toutes les équipes auxquelles il appartient [14].

II.4.2.1. Relations existantes entre les entités du niveau concret et les entités du niveau abstrait

- **Sujets et rôles** dans ce modèle, un sujet est considéré comme étant une entité active du système, c'est-à-dire un utilisateur, soit une organisation. Les exemples de sujet comprennent donc des utilisateurs tels que Jean, Marie, Pierre Ou des organisations telles que « le service comptable de la clinique privée langue-Doc, ... ». L'entité rôle est utilisée pour structurer le lien entre les sujets et les organisations. Dans le domaine de la santé, les rôles « cardiologue », « infirmière », « médecin » seront joués par des utilisateurs tandis que les rôles « service des urgences », « équipe de secours », « unité de soins intensifs » seront joués par des organisations. Puisque les sujets jouent des rôles dans les organisations, il est donné une relation qui relie ces entités ensemble. La relation liant le sujet, le rôle et l'organisation est la relation *habilitéée*. Par exemple, la relation habilitéée (FS-UDS, Paul, coordonnateur) signifie « l'organisation FS-UDS habilite le sujet Paul dans le rôle coordonnateur » [23]. La définition des rôles, l'affectation des rôles aux sujets et l'héritage des permissions à travers la hiérarchie ont pour objectif de structurer l'ensemble des sujets d'une organisation et de simplifier ainsi la gestion de la politique de sécurité;
- **Objets et vues:** les objets Or-BAC représentent des entités passives (fichiers, dossiers, administratifs, dossiers d'inscription, fichiers de note). Une vue correspondant à un ensemble d'objets qui satisferont une propriété commune (comme dans une base de données relationnelles) par exemple dans un système de gestion de fichiers, la vue « dossiers administratifs » correspond aux dossiers administratifs des patients alors que la vue « dossier médical » correspond aux dossiers médicaux des patients... Par conséquent, une relation appelée *utilise* est définie pour relier les organisations, les objets et les vues : si org est une organisation, o est un objet et v est une vue, alors (org, o, v) signifie qu'org utilise l'objet o dans la vue v [23];
- **Activités et actions:** les politiques de sécurité spécifient les accès autorisés aux entités inactives par les entités actives et régulent les actions effectuées dans le même système. Dans OR-BAC, l'entité action représente les opérations qui peuvent être effectuées par les sujets sur des objets. Elle représente les actions élémentaires telles que « lire », « écrire », « envoyer », etc. Les

activités correspondent aux actions qui ont le même objectif, par exemple « consulter », « modifier » ... Là encore l'objectif est de permettre à des organisations de structurer différemment les mêmes activités. La relation considérée liant les actions et les activités est utilisée pour associer trois entités organisation, action, activité [14];

- **Activités et actions:** les politiques de sécurité spécifient les accès autorisés aux entités inactives par les entités actives et régulent les actions effectuées dans le même système. Dans OR-BAC, l'entité action représente les opérations qui peuvent être effectuées par les sujets sur des objets. Elle représente les actions élémentaires telles que « lire », « écrire », « envoyer », etc. Les activités correspondent aux actions qui ont le même objectif, par exemple « consulter », « modifier » ... Là encore l'objectif est de permettre à des organisations de structurer différemment les mêmes activités. La relation considérée liant les actions et les activités est utilisée pour associer trois entités organisation, action, activité [14];
- **contexte** les contextes représentent ici les circonstances concrètes dans lesquelles les organisations accordent aux rôles les autorisations d'effectuer des activités sur les vues [14]. L'introduction des contextes est donc faite avec l'entité contexte qui est reliée aux entités organisation, sujet, objet et action par la relation définie. Si org est une organisation, s un sujet, o un objet, a une action et c un contexte, alors la relation définie (org, s, a, o, c) signifie qu'au sein de l'organisation org, le contexte c est vrai entre le sujet s, l'action a et l'objet o. Les conditions requises pour qu'un contexte donné soit lié, au sein d'une organisation donnée, à des sujets, des objets et des actions seront formellement spécifiées par des règles logiques.

Après avoir défini tous ces concepts, la notion de permissions est utilisée dans le but de pouvoir joindre des organisations, de vues, des activités et ceci dans un contexte donné[14].

Les politiques basées sur les organisations sont implémentées grâce à MOTORBAC qui est un outil permettant de spécifier les politiques de sécurité dynamique d'Or-BAC. En effet, chaque règle des politiques Or-BAC est associée à une condition contextuelle par exemple une règle pour être activé en fonction de l'heure ou de la position spatiale du sujet à laquelle elle s'applique. Il utilise l'interface de programmation (API) Or-BAC qui permet facilement d'interfacer le

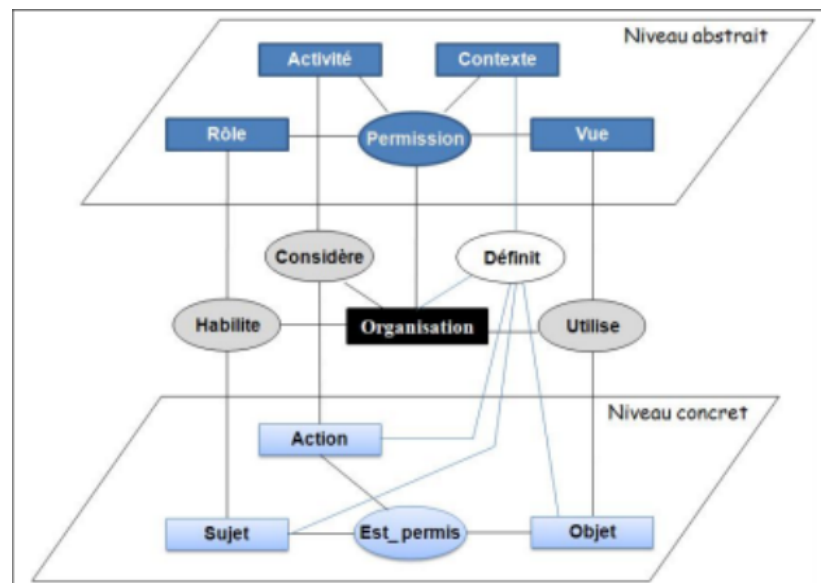


Figure 7 – Modèle Or-BAC [23]

module d'évaluation des contextes avec le monde extérieur de manière à interfacer une politique avec notre système d'information.

II.4.2.2. Limites du modèle Or-BAC

Bien que les politiques basées sur les organisations offrent la possibilité d'exprimer les règles contextuelles relatives aux permissions, aux interdictions, aux obligations et aux recommandations, elles ne permettent pas d'exprimer les règles spécifiques au super-utilisateur dans un SI. En d'autres termes, elles octroient tout le pouvoir au DBA (dataBase Administrator) d'une organisation en lui faisant une totale confiance. Ce qui n'est pas très normal, car ce dernier, parfois, use de ce pouvoir pour faire du n'importe quoi avec le système. De plus, tous comme dans les modèles basé sur les rôles, l'hiérarchie des rôle ne correspond pas à l'hiérarchie organisationnelle dans les organisations mais plutôt à la notion d'héritage des rôles.

II.5. Politiques et modèles de contrôle d'accès basés sur les attributs (ABAC)

L'un des principaux inconvénients des politiques basées sur l'identité ou les rôles, est que lorsqu'une exigence de contrôle d'accès est modifiée il est difficile d'identifier tous les endroits où l'implémentation de ces différentes politiques doit être mise à jour. Le contrôle d'accès basé sur les attributs (ABAC) est mis sur pied pour palier à ce problème. NIST SP 800-162, Guide to Attribute Based Access Control (ABAC) Definition and Considerations [10], définir ABAC comme une

méthodologie de contrôle d'accès logique dans laquelle l'autorisation d'effectuer un ensemble d'opérations est déterminée en évaluant les attributs associés au sujet, à l'objet, aux opérations demandées, et dans certains cas, aux conditions de l'environnement par rapport à la politique, aux règles ou aux relations qui décrivent les opérations admissibles pour un ensemble donné d'attributs. En général, ABAC évite que les capacités (paires opération/objet) soient directement assignées aux demandeurs ou à leurs rôles ou groupes avant que la demande ne soit faite [10]. Au lieu de cela, lorsqu'un sujet demande l'accès, le moteur ABAC peut prendre une décision de contrôle d'accès basée sur les attributs assignés du demandeur, les attributs assignés de l'objet, les conditions de l'environnement, et un ensemble de politiques qui sont spécifiées en termes de ces attributs et conditions. Selon cet arrangement, les politiques peuvent être créées et gérées sans faire référence directe à des utilisateurs et des objets potentiellement nombreux, et les utilisateurs et les objets peuvent être approvisionnés sans faire référence à la politique [24].

II.5.1. Les principaux composants d'ABAC

Les principaux composants du contrôle d'accès basé sur les attributs sont [10] :

- **Attributs:** les attributs sont des caractéristiques du sujet, de l'objet ou des conditions d'environnement, les attributs contiennent des informations données par une paire nom, valeur;
- **Sujet:** un sujet est un utilisateur humain ou Une entité non personnelle (NPE), tel qu'un dispositif qui émet des demandes d'accès pour effectuer des opérations sur des objets. Les sujets se voient attribuer un ou plusieurs attributs;
- **Objet:** un objet est une ressource du système dont l'accès est géré par le système ABAC, comme des dispositifs, des fichiers, des enregistrements, des tables, des processus, des programmes, des réseaux ou des domaines contenant ou recevant des informations. Il peut s'agir de la ressource ou de l'entité demandée, ainsi que de tout ce sur quoi une opération peut être effectuée par un sujet, y compris les données, les applications, les services, les dispositifs et les réseaux;
- **Opération:** c'est l'exécution d'une fonction à la demande d'un sujet sur un objet. Les opérations comprennent la lecture, l'écriture, l'édition, la suppression, la copie, l'exécution et la modification;

- **Politique:** c'est la représentation des règles ou des relations qui permettent de déterminer si un accès demandé doit être autorisé, compte tenu des valeurs des attributs du sujet, de l'objet et éventuellement des conditions d'environnement. Ces règles peuvent être exprimées par de nombreuses formes de langage informatique telles que :
 - Une combinaison booléenne d'attributs et de conditions qui satisfont l'autorisation d'une opération spécifique;
 - Un ensemble de relations associant les attributs du sujet et de l'objet et les opérations autorisées.
- **Conditions environnementales:** contexte opérationnel ou situationnel dans lequel les demandes d'accès se produisent. Les conditions d'environnement sont des caractéristiques environnementales détectables. Les caractéristiques de l'environnement sont indépendantes du sujet ou de l'objet, et peuvent inclure l'heure actuelle, le jour de la semaine, l'emplacement d'un utilisateur ou le niveau de menace actuel.

II.5.2. Architecture de fonctionnement du modèle ABAC

L'architecture de fonctionnement du modèle de contrôle d'accès basé sur les attributs est illustrée par la figure suivante :

- **User request:** est l'utilisateur qui émet une demande;
- **Decision:** est le résultat qui en ressort du traitement de la demande de l'utilisateur. Elle peut être l'accès accordé ou refusé;
- **Server Resource:** est le serveur des applications et des ressources ciblées par la demande de l'utilisateur;
- **Subject Resource:** est le serveur des applications et des sources externes
- **PAP (Policy Administration Point):** il est l'endroit où les politiques de contrôle d'accès sont éditées;
- **PEP (Policy Enforcement Point):** il est responsable de la protection des applications et des données auxquelles on souhaite appliquer ABAC. Il inspecte la demande et génère une demande d'autorisation à partir de celle-ci qu'il envoie au PDP;

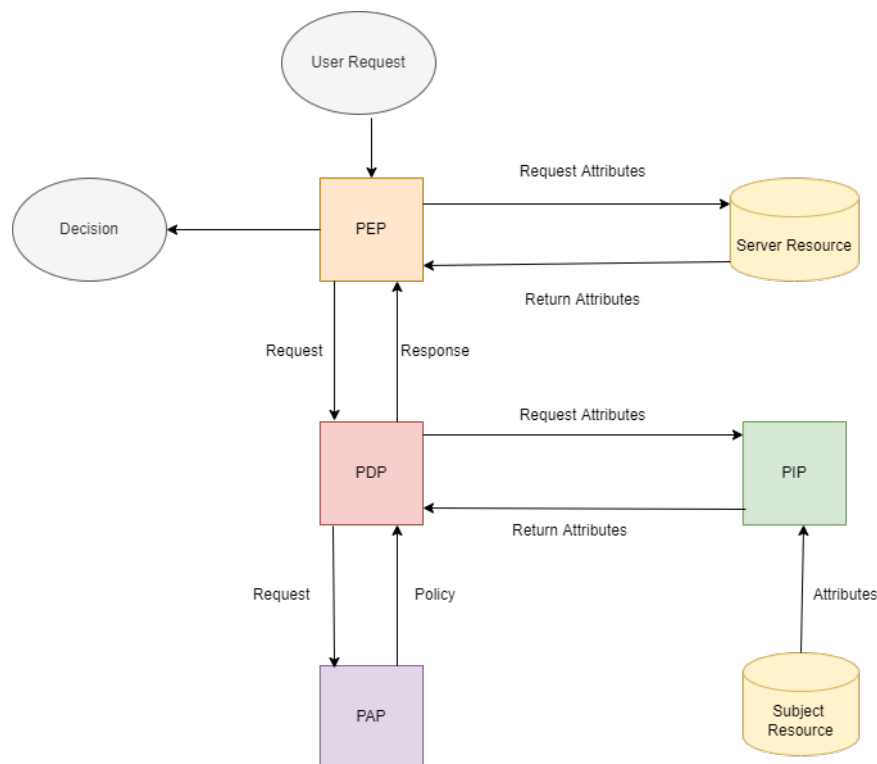


Figure 8 – Architecture de fonctionnement d'ABAC [10].

- **PDP (Policy Decision Point):** il est le cerveau de l'architecture. C'est l'élément qui évalue les demandes entrantes par rapport aux politiques avec lesquelles il a été configuré. Le PDP renvoie une décision d'autorisation/refus au PEP. Il peut également utiliser des PIP pour récupérer les méta-données manquantes;
- **PIP (Policy Information Point):** est le point où le PDP se connecte aux sources externes d'attributs comme LDAP ou une base de données. L'idée est que lors de l'évaluation d'une requête contre une politique, le PDP a besoin d'information (attributs) supplémentaire pour obtenir une décision.

II.5.3. Avantages du modèle ABAC

comme avantage d'ABAC nous avons:

- Les systèmes ABAC sont capables d'appliquer à la fois les concepts de contrôles d'accès discrétionnaire (DAC), obligatoire (MAC) et basé sur les rôles. [8], [10];

- Il est très souple dans un grand système ouvert où le nombre d'utilisateurs potentiels est très élevé et où la plupart des utilisateurs ne seront pas connus auparavant [3];
- Il n'utilise pas les attributs pour définir directement la permission entre les sujets et les objets mais il utilise les attributs comme la base de l'autorisation [3];
- Le stockage central des attributs des utilisateurs permet l'interopérabilité et le partage entre plusieurs fournisseurs de services qui peuvent utiliser ces données [3];
- Il donne les accords globaux des attributs afin que les attributs fournis dans un domaine puissent être transmis à l'autre domaine pendant l'interaction entre domaines [3];
- L'infrastructure est mise en place pour assurer la sécurité des informations et des attributs des utilisateurs, ainsi que pour effectuer les tâches d'autorisation et d'authentification.

II.5.4. Inconvénient du modèle ABAC

Bien que ce modèle est doté de nombreux avantages il fait face à des problèmes tels que:

- Lorsqu'elles sont déployées à l'échelle d'une entreprise dans le but d'accroître le partage d'informations entre diverses organisations, les mises en œuvre d'ABAC peuvent devenir complexes et s'appuyer sur l'existence d'une infrastructure de gestion des attributs, de politiques applicables par les machines et d'un ensemble de fonctions qui prennent en charge les décisions d'accès et l'application des politiques [8];
- Parfois, les attributs possédés par l'utilisateur ne correspondent pas nécessairement à ceux utilisés par le fournisseur de services d'un système basé sur le Web [3];
- L'hétérogénéité des informations sur les utilisateurs accroît la complexité, qui ne peut être résolue que par une base de données centralisée contenant tous les attributs dans le même format [3];
- l'un des inconvénients du modèle ABAC est qu'il ne permet pas de spécifier les règles spécifiques au contrôle de l'administrateur du système d'information

ce qui lui donne la possibilité de créer d'entités virtuelles et ceci autant de fois qu'il le souhaite

- Dans des situations d'urgence, ABAC peut refusé l'accès à une utilisateur légitime suite à l'absence d'une valeur d'attribut lors d'une décision d'accès. ce qui ne respecte pas le fondement même du contrôle d'accès

II.6. Politiques et modèles de contrôle d'accès basés sur les rôles améliorés par les attributs(AERBAC)

Afin de palier au problème d'explosions de rôles et de permissions que soulève RBAC et au problème d'analyse des politiques de revision ou de modification des permissions qui est une tache difficile dans ABAC; Qasim Mahmood Rajpoot et Al [16] proposent un modèle de contrôle d'accès basé sur les rôles et amélioré par les attributs (AERBAC). AERBAC fournit un mécanisme de contrôle d'accès à grain fin qui non seulement prend en compte les informations contextuelles lors de la prise de décision de contrôle d'accès, mais convient également aux applications où l'accès aux ressources est contrôlé en exploitant le contenu des ressources dans la politique. Ce modèle conserve la flexibilité offerte par ABAC, tout en conservant les avantages de RBAC, à savoir une administration, une analyse des politiques et une révision des permissions plus facile. En plus de cela, il présente les caractéristiques clé suivantes

- Il permet de prendre des décisions de contrôle d'accès en fonction du contexte en associant des conditions aux permissions qui sont utilisées pour vérifier si les informations contextuelles requises sont présentes au moment de la prise de décision;
- Il offre un système d'autorisation basé sur le contenu tout en gardant l'approche orientée rôle, afin de conserver les avantages offerts par RBAC. Ceci grâce à la spécification des permissions à l'aide des attributs des objets plutôt qu'en utilisant uniquement leur identifiant.

II.6.1. Composants du modèle AERBAC

Ici les entités utilisateurs, rôles, objets, et opérations ont la même sémantique que dans RBAC. Les utilisateurs et les objets sont également associés à des attributs. L'attribut environnement est également incorporé pour rendre compte de la

situation dans laquelle l'accès doit être autorisé . En plus de ceux-là, il intègre des composants tels que [16] :

- **Les attributs:** ils capturent les propriétés d'entités spécifiques (par exemple, l'utilisateur), AERBAC définit une fonction d'attributs pour chaque attribut du système qui est chargé de renvoyer la valeur de cet attribut. Chaque attribut est représenté par une plage d'ensemble fini des valeurs automatiques. Par exemple, la plage de l'attribut branche est un ensemble de valeurs de noms de branches semi-pertinents pour le domaine d'application. Un attribut peut être statique, c'est-à-dire que ses valeurs changent rarement, par exemple : la désignation, le département, le type. Soit dynamique, c'est-à-dire que ses valeurs peuvent changer fréquemment et de manière imprévisible, de sorte qu'elles peuvent très bien changer pendant la durée de vie d'une session. Exemple : officier commandant, le lieu, l'occurrence d'un indice.
- **Permissions et conditions:** contrairement aux approches traditionnelles de RBAC, les permissions dans AERBAC font référence aux objets indirectement, en utilisant leurs attributs. Une permission fait référence à un ensemble d'objets partageant des attributs communs, par exemple : le type ou la branche, en utilisant une seule permission, contrairement aux permissions séparées pour chaque objet unique. Dans AERBAC, une permission est constituée d'une expression d'objet et d'une opération autorisée sur l'ensemble d'objets désignés par l'expression. Les expressions d'objet sont formées en utilisant les attributs des objets. Chaque permission est associée à une ou plusieurs conditions, qui doivent être évaluées comme étant vraies pour que l'utilisateur puisse exercer cette permission;
- **Session:** une session contient une liste de permissions associées aux rôles activés par l'utilisateur. Comme décrit précédemment, les permissions sont différentes des permissions RBAC standard en termes de référence aux objets en utilisant leurs attributs et en étant liées aux conditions qui sont évaluées chaque fois qu'une permission doit être exercée;
- **Demande d'accès:** Dans AERBAC, la demande de l'utilisateur peut également être basée sur les attributs des objets. Par exemple, dans une application d'imagerie médicale, un utilisateur peut vouloir voir toutes les images contenant des caractéristiques spécifiées, par exemple, les objets avec type=tumeur, domaine= hopital-nw. Pour qu'une demande de l'utilisateur soit acceptée, il doit exister une expression d'objet dans la session de l'utilisateur

qui désigne les objets demandés, et la condition liée à cette expression d'objets doit être évaluée comme étant vraie.

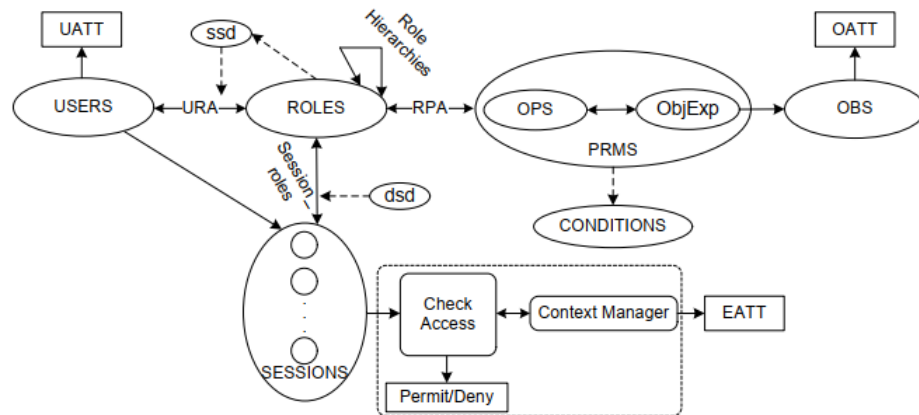


Figure 2.9 – Modèle de contrôle d'accès basé sur les rôles et amélioré par les attributs [16].

2.6.2. Modèle formel AERBAC

AERBAC est un modèle formel qui incorpore les attributs de l'utilisateur, de l'objet, et de l'environnement dans RBAC d'une manière orientée rôle. La logique de premier ordre est utilisée ici pour faire des descriptions formelles et la convention selon laquelle toutes les variables non liées sont universellement quantifiées et données comme Range (att) est suivie. UATT, OATT et EATT sont respectivement des ensembles de fonctions d'attribut pour les utilisateurs, les objets et l'environnement. Chaque fonction d'attributs renvoie soit un ensemble de valeur, soit une valeur atomique, déterminé en fonction du type de l'attribut (c'est-à-dire attType) [16]. La relation d'attribution de rôles et de permissions (RPA) capture les permissions qui sont attribuées à un rôle lorsqu'un ensemble donné de conditions est rempli. Un ensemble de permissions peut changer pour un rôle si les conditions varient entre les requêtes.

2.6.2.1. Décision d'accès

Dans ce modèle, une demande d'accès de l'utilisateur peut spécifier explicitement un objet, en lisant son identifiant, soit designer implicitement un ensemble d'objet en utilisant les attributs des objets. Si une demande de l'utilisateur ne porte pas sur un objet spécifique, mais plutôt sur un ensemble d'objets, le système doit prendre en compte les critères donnés pour renvoyer les objets demandés. Lorsqu'un utilisateur soumet une demande d'accès, celle-ci doit être évaluée par rapport à la politique. AERBAC distingue ainsi deux formes d'évaluation des demandes selon leur spécification [16].

- **Demande basée sur l'identifiant:** Ici, l'utilisateur spécifie l'identifiant de l'objet auquel il veut accéder. L'évaluation de ce type de demande est très simple. Dans ce cas, la fonction CheckAccess renvoie true si et seulement s'il existe une permission P dans les permissions disponibles au cours d'une session donnée qui contient une expression d'objet qui vaut vrai pour obj, une opération accordée dans la session et la condition correspondant vaut true;

- **Demande basée sur les attributs:** Ici, l'utilisateur peut spécifier les attributs de l'objet dans sa demande, plutôt qu'un identifiant unique de l'objet. Le fait de spécifier les attributs de l'objet dans la demande implique que l'utilisateur souhaite accéder à tous les objets qui ont les valeurs d'attribut spécifiées. Il a donc été examiné deux possibilités de formuler et de traiter de telles demandes.
 - **Requêtes de ressources:** Dans cette approche, la demande de l'utilisateur contient une expression similaire aux expressions des objets. La fonction `checkAccess` reçoit en entrée la demande d'accès et renvoie en sortie les objets autorisés à l'utilisateur si la demande est acceptée, sinon la demande est refusée. Lors de l'évaluation d'une requête, l'expression donnée est convertie en une requête et les objets résultants sont récupérés dans la base de données des ressources. L'étape suivante consiste à trouver les expressions d'objet applicables en faisant une comparaison entre l'opération demandée par l'utilisateur et celles mentionnées dans l'ensemble de permissions existantes dans la session de l'utilisateur. Une fois les expressions d'objets sélectionnées, elles sont évaluées une par une pour chaque objet renvoyé par la requête. Si une expression d'objet et sa condition correspondante sont évaluées comme vraie pour un objet, l'objet est ajouté à la liste des objets autorisés à accorder à l'utilisateur. Enfin, l'utilisateur se voit accorder l'accès à tous les objets pour lesquels une expression d'objet et sa condition correspondante sont vraies. Comme les expressions d'objet doivent être évaluées pour chaque objet retourné, cette approche peut s'avérer coûteuse dans le cas où plusieurs objets seraient retournés par la requête formée sur la base de la demande de l'utilisateur;
 - **Les valeurs des attributs:** une autre stratégie consiste à évaluer la demande de l'utilisateur par rapport aux expressions d'objets avant de récupérer les objets réels dans la base de données des ressources. Dans cette approche, plutôt que de fournir une expression, l'utilisateur spécifie sa demande d'accès en indiquant les valeurs d'attribut des objets souhaités. La fonction `checkAccess` reçoit en entrée la demande `Req` de l'utilisateur et renvoie les objets désignés par les valeurs d'attributs d'objets données dans `Req`, si la demande est acceptée, sinon la demande est refusée. Pour traiter la demande de l'utilisateur, on identifie toutes les expressions d'objet existant dans la session de l'utilisateur qui utilisent les attributs mentionnés dans la demande de l'utilisateur et l'opération spécifiée dans cette permission correspond à l'opération demandée. Les expressions d'objet qui incluent un attribut non spécifié par la demande de l'utilisateur ne sont pas pertinentes. Ensuite, pour chaque expression d'objet pré-sélectionnée, les fonctions d'attribut dans l'expression d'objet reçoivent les valeurs d'attribut fournies par l'utilisateur. Dès qu'une expression d'objet et sa condition correspondante reviennent à la réalité, la demande de l'utilisateur est acceptée et les autres expressions d'objet sont ignorées. Lorsqu'une expression retourne vraie, il est formé une requête basée sur les valeurs d'attributs d'objets spécifiées dans la demande de l'utilisateur et l'utilisateur a accès à tous les objets retournés par la requête.

Ici ne sont évaluées que les expressions d'objet qui utilisent des attributs

d'objet spécifiés dans la requête de l'utilisateur.

Cette approche permet de prendre une décision d'accès en évaluant uniquement les expressions d'objet, sans avoir à extraire les objets de la base de données des ressources. C'est important, car de nombreuses demandes peuvent être refusées à ce stade sans avoir à récupérer les objets et à évaluer les conditions.

2.6.3. Inconvénients

D'après [21], AERBAC ne résout pas totalement le problème d'explosion des rôles et des permissions que pose RBAC et modifie pas la même occasion la structure de base de ce dernier empêchant ainsi une migration des politiques déjà spécifiées à l'aide d'RBAC. Tout comme ses prédécesseurs, ce modèle octroie tout le pouvoir au DBA d'une organisation en lui faisant une totale confiance. Favorisant ainsi un usage malveillant du système.

2.7. Politiques et modèles de contrôle d'accès basés sur les rôles et les attributs(ARBAC)

C'est dans le but de résoudre le problème d'explosions de rôles et des permissions que soulève RBAC et celui de la difficulté d'analyser les politiques de sécurité que soulève ABAC que Singh et al, ont proposé ARBAC. ARBAC est un modèle de contrôle d'accès qui définit les politiques de sécurité sur la base des rôles et des attributs. Il introduit au dessus du modèle RBAC le concept de "règle" est basée sur les attributs qui permettent d'activer ou de désactiver les rôles d'un utilisateurs et les permissions associées à ces rôles. Ceci dans le but de permettre une spécification des politiques multidimensionnelles à grain fin [21].

2.7.1. Composants du modèle ARBAC

Ce modèle a pour composant:

- **User ou Utilisateur:** un utilisateur est une entité qui provoque le flow d'information et est capturée par la relation *User*. Les utilisateurs peuvent avoir un ou plusieurs attributs, et chaque attribut peut avoir une ou plusieurs valeurs d'attribut qui sont capturés dans les relations *User-attribute* et *User-attribute-values* respectivement [21];
- **Objet:** dans une organisation, les objets contiennent ou reçoivent des informations et sont capturés dans la relation *Objet*. Les objets peuvent avoir un ou plusieurs attributs, et chaque attribut peut avoir une ou plusieurs valeurs d'attribut qui sont capturés dans les relations *Object-attribute* et *Object-attribute-values* respectivement [21];
- **Environnement:** l'environnement tel que le temps, l'emplacement est un facteur supplémentaire, indépendant de l'utilisateur et de l'objet, qui peut restreindre d'avancé

tage la disponibilité des objets pour les utilisateurs. il peut avoir un ou plusieurs attributs et chaque attribut peut avoir une ou plusieurs valeurs qui sont saisie dans les les relation *environment-attribut* et *environment-attribut-values* respectivement [21];

- **Rôles et hiérarchie des rôles:** les rôles définissent les fonctions professionnelles que les utilisateurs supposent exécuter dans une organisation. Une rôle dans ARBAC, est contraint par un ensemble de règles qui peuvent être composées d'attributs d'utilisateur et d'attributs de l'environnement. Ainsi les rôles seraient disponibles que pour les utilisateurs qui possèdent les attributs nécessaires et satisfont au moins une des règles associées à ces rôle. la hiérarchie des rôles définit une relation d'ordre partiel sur les rôles qui permet l'héritage des rôles [21];
- **Permission:** une permission définit l'autorisation sous la forme d'un droit (Par exemple, lecture, écriture, etc.) sur un objet qui habilite un utilisateur à effectuer une tâche par le biais d'un rôle;
- **Attribution des rôles aux utilisateurs:** ici un utilisateur peut avoir un ou plusieurs rôles, et un rôle peut être attribué à plusieurs utilisateurs;
- **Affectation des permissions aux rôles:** Dans ARBAC, un rôle peut avoir une ou plusieurs permissions, et une permissions peut être assignée à plusieurs rôles;
- **Session:** dans RBAC, une session capture un sous-ensemble de rôles actifs de l'ensemble des rôles attribués à un utilisateur. un utilisateur peut activer une ou plusieurs sessions, mais chaque session serait attribuée à un seul utilisateur. Dans ARBAC, l'activation d'un sous-ensemble de rôles d'utilisateur dans des sessions dépendant de la satisfaction des règles à ces rôles [21];
- **Contraintes:** l'ARBAC proposé peut également spécifier différents types de contraintes qui permettent de capturer les rôles ou les permissions critiques d'une organisation, le lien entre l'utilisateur et un rôle ou celui entre un rôle et une permission, les valeurs d'attribut communes à un ensemble d'utilisateurs ou d'objets et en fin la relation cardinalité qui permet de voir le nombre d'occurrences de chaque relation;
- **Règle:** ARBAC introduit la notion de règle afin d'activer ou de désactiver les rôles et les permissions. Dans l'ARBAC, les règles qui restreignent la disponibilité des rôles pour les utilisateurs peuvent être composées des valeurs d'attributs d'utilisateur et de valeurs d'attributs d'environnement . De même, les règles qui restreignent la disponibilité des permissions peuvent être composées de valeurs d'attribut d'utilisateur, de valeurs d'attribut d'objet, et de valeurs d'attributs d'environnement [21];
- **Attribution des règles aux rôles:** RuR dans ARBAC, capture l'association des règles aux rôles. Une règle peut être assignée à plusieurs rôles, et un rôle, peut avoir plusieurs règles. L'association d'un ensemble de règles basées sur les attributs à chaque rôle permet de réduire l'espace des règles applicables pour un rôle, et de minimiser le temps d'évaluation de la disponibilité des rôles pour un utilisateur [21];

- **Attribution des règles aux permissions:** RuP dans ARBAC, capture l'association des règles avec les permissions. Une règle peut être assignée à plusieurs permissions, et une permissions, peut avoir plusieurs règles. L'association directe d'un ensemble de règles basées sur les attributs aux permissions permet de réduire l'espace des règles applicables pour une permission, et de minimiser le temps d'évaluation de la disponibilité de la permission pour un rôle [21].

2.7.2. Avantages du modèle ARBAC

Comme avantages de ce modèle nous avons:

- le modèle ARBAC permet une administration plus facile des politiques;
- il permet de limiter la disponibilité des rôles et des permissions
- ce modèle permet une spécification de politiques flexibles et sensibles au contexte. C'est-à-dire que grâce aux concepts de règle, d'attribut et d'environnement qu'introduit ARBAC nous permettent d'exprimer des politiques d'accès basées sur des attributs et ceci dans un contexte bien précis
- ce modèle permet de minimiser considérablement les effets de migration. En effet, cette approche permet aux entreprises et applications utilisant le modèle RBAC de migré leurs politiques dans ARBAC sans toutes fois les modifier.

2.7.3. Inconvénients du modèle ARBAC

Comme inconvénients nous avons:

- ARBAC ne permet pas l'analyse automatique de la sécurité des politiques;
- ce modèle ne permet pas de contrôles les faits et gestes du super-utilisateur.

2.8. Politiques et modèles de contrôle d'accès basés sur la hiérarchie organisationnelle (HOr-RBAC)

Les politiques basées sur la hiérarchie organisationnelle sont venues palier au problème des politiques précédentes. Elles permettent d'exprimer les règles de sécurité telles que [23] :

- les règles qui spécifient le contrôle du super-utilisateur;
- les règles qui spécifient la hiérarchie entre les unités administrative et opérationnelles;
- les règles spécifiques pour empêcher l'ajout des entités virtuelles dans le système d'information d'une organisation.

•

HOr-BAC s'appuie sur le modèle Or-BAC en y implémentant des concepts propres à lui-même. Ces politiques tirent leur originalité du fait qu'elles sont purement construites autour de la structure organisationnelle d'une organisation, cela signifie que dans le modèle HOr-BAC, il est possible pour nous de trouver tous les composants principaux de la structure organisationnelle d'une organisation réelle.

2.8.1. Composants de base du modèle HOr-BAC

2.8.1.1. Unité organisationnelle

C'est un regroupement des unités administratives et opérationnelles. Elle est définie dans notre modèle comme une entité ayant un rôle administratif ou opérationnel et remplace ainsi l'entité rôle vu dans le modèle Or-BAC. Ce concept nous permet de mieux représenter la relation hiérarchie qu'il existe entre ces deux unités [23].

- **Unité organisationnelle:** elle représente l'ensemble des employés métiers ayant les mêmes formations, les mêmes rôles et une fonction spécifique dans une organisation. Elle ne prend aucune décision sur le fonctionnement de l'organisation et ne fait qu'obéit aux décisions qui lui ont été données par l'unité administrative qui la subordonne. Exemple : comptabilité, enseignant;
- **Unité administrative:** elle représente l'ensemble des unités décisionnelles de l'organisation. Cette entité permet de représenter les fonctions de contrôle, de supervision, et de validation des requêtes émises dans le SI. Elle peut être placée sur une unité opérationnelle ou sur une autre unité administrative. Exemple : département, rectorat et décanat dans une université. Les modèles précédents basés sur les rôles n'ont pas fait une différence entre un rôle opérationnel et un rôle administratif. Or, il existe bel et bien une différence entre ces rôles. En considérant par exemple les rôles, directeur et infirmier dans une organisation hospitalière, ils ne sont pas de même nature, car l'infirmier travaille sous le contrôle du directeur ; Ainsi, le rôle infirmier est une unité opérationnelle et le rôle directeur est une unité administrative.

2.8.1.2. Employé métier

Il représente une personne physiquement identifiable ayant un rôle actif dans l'organisation. C'est la seule entité réellement active dans l'organisation. Le concept d'employé métier remplace le concept de sujet vu dans le modèle Or-BAC. L'intérêt d'une telle approche réside sur le fait qu'étant donné que l'utilisateur est une personne physique, il pourra être facilement contrôlé ; Car s'il existe des sujets virtuels, le super-utilisateur pourra les ajouter autant qu'il le souhaite et les habilitier à jouer les rôles de son choix.

2.8.1.3. Ressources

L'entité ressource est utilisée pour organiser l'ensemble des données de l'organisation et exprimer les entités passives. Cette entité peut être la note d'un étudiant, les dossiers d'inscription, les dossiers de changement de grade des enseignants pour une organisation universitaire [23].

2.8.1.4. Requête

Elle est définie comme étant une demande faite par un Employé métier de l'organisation. Une requête doit avoir les informations suivantes : le nom de l'émetteur et la ressource. Il existe deux types de requêtes à savoir les requêtes à exécution directe (réelle) et indirecte (différée). Lorsqu'elle est indirecte, elle nécessite automatiquement la validation du supérieur hiérarchique. Par contre une requête à exécution directe n'a pas besoin de validation du supérieur hiérarchique.

2.8.2. Mode de traitement d'une requête dans HOr-BAC

Il consiste à définir certains attributs qui seront pris en compte lors de l'émission et du traitement d'une requête. Étant donné que l'un des buts du modèle HOr-BAC est le contrôle des opérations de base d'un DBA dans une organisation de manière interactive, nous nous intéresserons sur les attributs suivants :

- **Effet:** consiste à spécifier si la requête émise doit attendre l'accord de la hiérarchie ou pas;
- **Attente:** spécifie le temps d'attente après lequel l'émetteur obtient l'accord tacite de sa requête;
- **Nombre:** spécifie le nombre de fois que l'émetteur a le droit d'exécuter une requête;
- **Mode:** spécifie le mode de traitement de la requête qui peut être validé, rejeté ou transmis.

2.8.3. Politique de sécurité du modèle HOr-BAC

Une politique de sécurité régleme les accès au système à travers des permissions. Une permission dans notre modèle matérialise le fait qu'une organisation autorise une unité organisationnelle de traiter ou d'émettre une requête donnée dans une vue donnée. Le modèle HOr-BAC dans le but de matérialiser la hiérarchie entre les différentes unités organisationnelles implémente en son sein deux types de permission afin de différencier les actions effectuées par chaque unité. Ainsi, on distingue donc :

2.8.3.1. Les permissions dites de préparation

Encore appelée permissions opérationnelles, elles permettent aux unités opérationnelles de préparer (initier, émettre ou soumettre) des requêtes à leur hiérarchie. Cela laisse sous-entendre qu'aucune unité opérationnelle n'a le droit de traiter une requête venant d'une autre unité organisationnelle, car sa seule fonction est d'exécuter. Elle est matérialisée par la relation permissions-opérationnelles qui relie les entités organisation, unité opérationnelle, requête et vue.

2.8.3.2. Les permissions dites de validations

Encore appelées permissions administratives, elles permettent aux unités administratives de traiter (contrôler, rejeter, valider et/ou transmettre) les requêtes provenant des unités opérationnelles. Elle est matérialisée par la relation permissions-administratives qui relie les entités organisation, unité opérationnelle, unité administrative, requête et vue.

2.8.3.3. Les permissions dites de préparation

Sachant qu'une personne est appelée à jouer au sein d'une organisation un certain nombre de rôles, y compris ceux qu'elle joue habituellement, le modèle HOr-BAC nous donne la possibilité à travers l'entité contexte de définir une situation ou des circonstances dans lesquelles les organisations accordent des permissions à des rôles pour réaliser des requêtes sur des vues. Ainsi à nos permissions citées plus haut nous ajoutons l'entité contexte pour spécifier à quel moment, situation et localisation les unités opérationnelles et administratives sont respectivement appelées à émettre ou à traiter une requête donnée dans une vue donnée. Ces permissions ne permettent qu'à une organisation donnée de spécifier les permissions accordées aux unités organisationnelles suivant un contexte précis. Mais, elles ne permettent pas de décrire des actions concrètes que réalisent les employés sur les ressources. Ainsi, nous avons implémenté le concept de contrôle d'accès de bas niveau à travers les relations suivantes:

- **Peut-suggérer:** cette relation permet à un employé d'obtenir la permission de suggérer l'application d'une action sur une ressource donnée;
- **Peut-traiter:** cette relation matérialise le fait que le supérieur hiérarchique d'un employé a l'autorisation de traiter les suggestions d'application d'une action donnée sur une ressource donnée.

2.8.4. Architecture de fonctionnement du modèle HOr-BAC: parapheur électronique

Contrairement aux autres modèles, le modèle HOr-BAC implémente en son sein le concept de parapheur électronique. Il s'agit ici d'un processus de contrôle de l'émission et du traitement des requêtes dans un système d'information. Il permet de contrôler les actions des utilisateurs du système y compris celles du super-utilisateur ou DBA. Ici, nous supposons que la quasi-totalité des actions dans une organisation se fait sur la demande et chaque demande obtient une validation pour une exécution effective dans le système sinon, la demande est rejetée. Son fonctionnement est illustré par le schéma ci-dessous :

- **Employee Request:** c'est l'employé qui émet la requête;
- **Employee Supervisor:** c'est le supérieur hiérarchique de l'employé qui émet la requête. Il a pour rôle de valider, refuser, traiter ou transmettre la requête de ce dernier;
- **Security store:** il représente la base de données qui stocke les politiques de sécurité de l'entreprise, les informations sur la structure organisationnelle, les informations sur les employés, les comptes utilisateurs. Toutes ces informations sont enregistrées avant le fonctionnement du parapheur;
- **Initialization:** c'est l'ensemble des procédures permettant d'enregistrer les données dans le security store;
- **Context information:** c'est le contexte (temporel, géographique, lié au sujet ou à la ressource) dans lequel la requête est traitée;

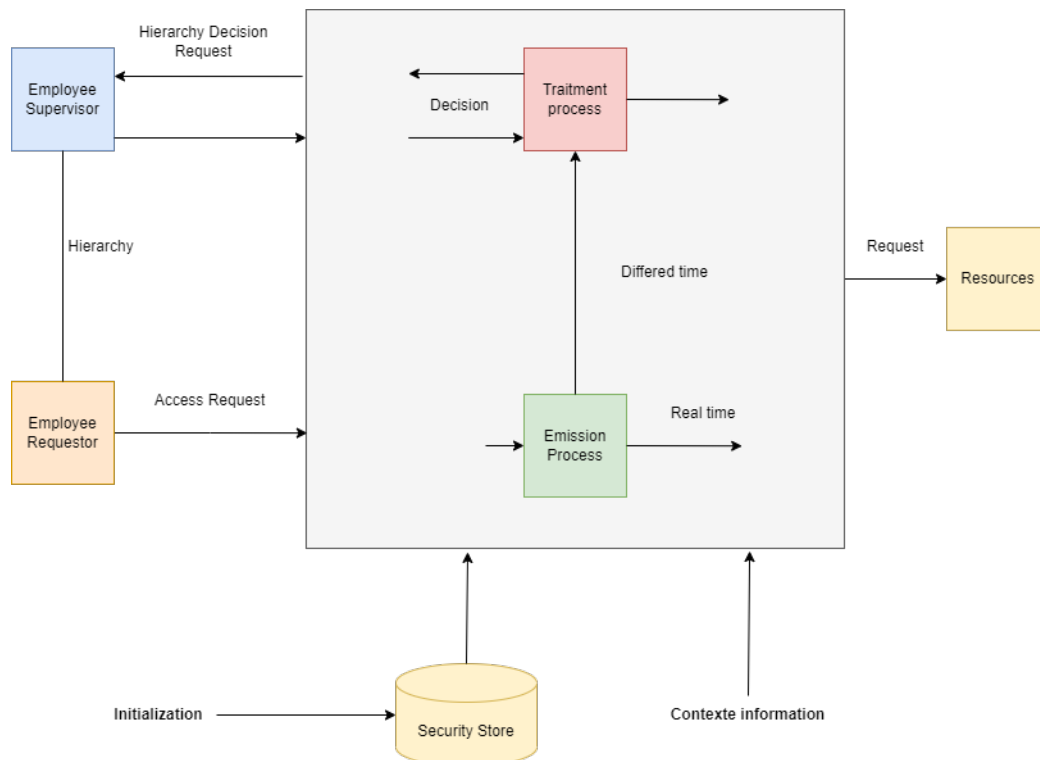


Figure 2.10 – Composants du parapheur électronique

- **Emission process** : c'est l'algorithme qui permet l'émission d'une requête. Selon le type de la requête, il effectue le traitement donné. S'il s'agit d'une requête immédiate, il l'exécute de façon automatique et enregistre son résultat dans la BD avant d'accorder ou pas l'accès à la ressource à ce dernier. Si la requête est différée, il l'enregistre puis la transmet au traitement process. ;
- **Traitement process**: c'est l'algorithme qui s'occupe du traitement des requêtes différées. Quand il reçoit une requête, il envoie une alerte au supérieur hiérarchique de l'employé qui a émis la requête pour qu'il la valide, refuse, traite ou la transmette. Une fois qu'il obtient la décision du supérieur hiérarchique, il accorde ou pas l'accès à la ressource;
- **Resource**: c'est la ressource à laquelle un employé souhaite accéder.
-

2.8.5. Inconvénient du modèle HOr-BAC

Dans HOr-BAC, les politiques de contrôle ne peuvent être définies que sur la base de l'unité organisationnelle à laquelle appartient un employé, ce qui limite ainsi la flexibilité du contrôle d'accès. Par conséquent il ne permet que de définir des politiques de contrôle d'accès à gros grain, c'est-à-dire des politiques qui ne sont définies que sur la base des unités organisationnelles.

2.9. Conclusion

En somme, nous avons présenté dans ce chapitre l'état de l'art sur les politiques et modèles de contrôles d'accès. Nous avons vu que bien que les politiques et modèles de contrôles d'accès discrétionnaires et obligatoires ont été implantés dans les systèmes d'exploitation Windows et Linux, leur principal inconvénient est que les permissions sont directement affectées aux utilisateurs. Les politiques et modèles à bases des rôles ont donc vu le jour pour palier à ce problème. Le premier de ces modèles a été le modèle RBAC qui a introduit la notion de rôle dont les permissions sont affectées aux rôles et les rôles affectés aux utilisateurs. Son inconvénient majeur a été l'héritage des rôles. De plus, ce modèle n'exprime pas les permissions et les interdictions. Le modèle Or-BAC est donc venu résoudre ces problèmes en introduisant le concept d'organisation. Cependant, ces différents modèles permettent d'exprimer les politiques de sécurité de façon statique. Ainsi ABAC a vu le jour. ABAC attribue l'accès aux ressources sur la base des attributs de différentes entités, il permet une spécification flexible des politiques de sécurité et un contrôle à grain fin des ressources du système. Toutefois, ABAC est difficile d'administration. De ce fait, plusieurs auteurs ont proposé différentes modèles issues de la fusion des modèles RBAC et ABAC afin de combler leurs lacunes tel que le problème d'explosion de rôles et de permissions donc souffre RBAC. l'un de ces modèles fut le modèle de contrôle d'accès basé sur les rôles améliorés par les attributs (AERBAC) qui vient résoudre ce problème en permettant une flexible des politiques de sécurité basé sur les rôles et attribut. Toutefois d'après [21], AERBAC ne résous pas totalement le problème d'explosion des rôles et des permissions que pose RBAC et modifie pas la même occasion la structure de base de ce dernier. Ainsi ils proposent un nouvelle modèle qui place au dessus de la structure de base de RBAC le concept de règle basé sur les attributs d'utilisateur, d'objet et de contexte qui permet d'activer ou de désactiver les rôles et les permissions. Ce modèle permet également un contrôle à grain fin des objets du système. Mais aucun de ces modèles ne contrôle les opérations effectuées par le super-utilisateur au sein du système d'information. Nous avons ainsi vu que c'est le modèle HOr-BAC qui a effectué ce contrôle en intégrant au modèle Or-BAC le concept de parapheur électronique. L'inconvénient majeur du modèle HOr-BAC est que les politiques de contrôle d'accès ne peuvent être définies que sur la base de l'unité organisationnelle à laquelle appartient un employé, ce qui limite ainsi la flexibilité du contrôle d'accès.

3

CHAPTER UNE APPROCHE ORIENTÉE ATTRIBUTS POUR LE CONTRÔLE D'ACCÈS BASÉ SUR LA HIÉRARCHIE ORGANISATIONNELLE (AHOOr-BAC)

SOMMAIRE

3.1 - Introduction	41
3.2 - Motivation	41
3.3 - Concepts de base du modèle AHOOr-BAC	42
3.4 - Présentation des relations dans AHOOr-BAC	45
3.5 - Algorithme du parapheur électronique	51
3.6 - Conclusion	55

3.1. Introduction

Dans le chapitre précédent, nous avons présenté les différents modèles de contrôles d'accès y compris le modèle HOOr-BAC. Le but de notre travail étant la proposition d'un modèle de contrôle d'accès basé sur la hiérarchie organisationnelle et les attributs. Nous proposerons une nouvelle couche fonctionnelle au-dessus du modèle HOOr-BAC afin de permettre une spécification flexible et sensible au contexte des politiques. Nous allons dans ce chapitre, présenter notre Modèle. Nous commencerons dans un premier temps par présenter les concepts qui constitueront notre modèle. Ensuite nous présenterons les différentes relations qui existent entre ces différents concepts. Puis, nous continuerons par décrire les différentes phases du parapheur électronique. En fin nous présenterons un cas pratique de notre approche.

3.2. Motivation

De tous les politiques et modèles de sécurité qui existent dans la littérature (et que nous avons décrits dans le chapitre précédent), seul les politiques basées sur la hiérarchie

organisationnelle permettent de spécifier les règles qui d'une part, assurent le contrôle des opérations effectuées par le super-utilisateur au sein du système d'information, puis représentent la relation hiérarchique qui existe entre les unités de traitement administratives et opérationnelles et en fin empêchent la création d'entités virtuelles dans le système d'information de l'organisation. Cependant, HOr-BAC ce modèle ne permet pas les points suivants:

- une spécification flexible des politiques de sécurité;
- un contrôle à grain fin des ressources protégées du système.

La section suivante présente un ensemble de concepts permettant de spécifier un modèle de sécurité capable de prendre en compte ces différents points.

3.3. Concepts de base du modèle AHor-BAC

3.3.1. Organisation

Une organisation est un ensemble d'individus, regroupés au sein d'une structure régulée, ayant un système de communication pour faciliter la circulation de l'information, dans le but de répondre à des besoins et d'atteindre des objectifs déterminés. Elle est représentée dans notre modèle par l'entité *Organisation*

3.3.2. Employé Métier

Il représente une personne physiquement identifiable ayant un rôle actif dans l'organisation. C'est la seule entité réellement active dans l'organisation. Tout comme dans HOr-BAC, ce concept permet d'empêcher la création des entités virtuelles dans le système d'information par le super-utilisateur. Car, si l'utilisateur est une personne physique, il pourra être facilement contrôlé. Les employés métier peuvent avoir un ou plusieurs attributs, et chaque attribut peut avoir une ou plusieurs valeurs d'attributs qui sont représentés par les entités *AttributE* et *valeurAttributE* respectivement. Dans notre modèle, on attribue une ou plusieurs valeurs d'attributs d'employés à chaque Employé métier. Ce qui est représenté par la relation suivante:

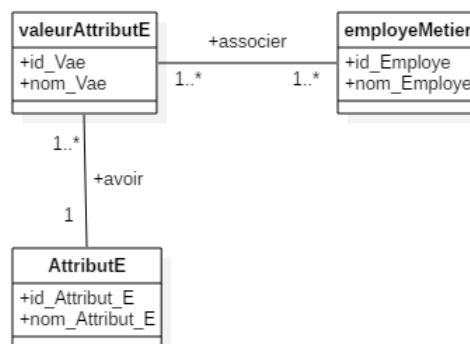


Figure 3.11 – Affectation des valeurs d'attribut d'employé aux employés métiers

3.3.3. Ressource

l'entité *ressource* est utilisée pour organiser l'ensemble des données et informations de l'organisation et exprime les entités passives du système. Par exemple, le dossier médical d'un patient, les dossiers d'inscription. les ressources peuvent avoir un ou plusieurs attributs, et chaque attribut peut avoir une ou plusieurs valeurs d'attributs de ressource qui sont représentés par les entités *AttributR* et *valeurAttributR* respectivement. Notre modèle, attribue une ou plusieurs valeurs d'attribut de ressource à chaque ressource. cela est matérialisé par la relation suivante:

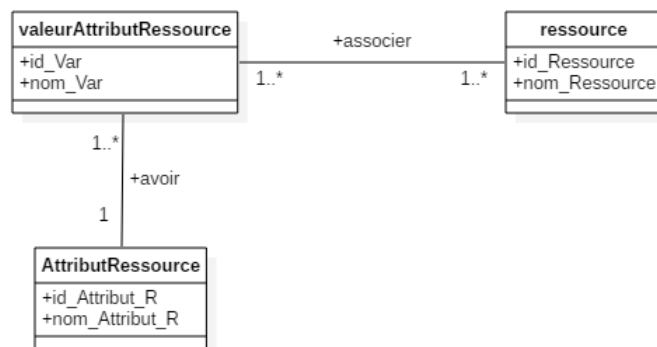


Figure 3.12 – Affectation des valeurs d'attribut de ressource aux ressources

3.3.4. Requête

Tout comme HOr-BAC, une requête est définie dans notre modèle comme une demande fait par un employé métier dans le système. Une requête doit avoir les informations suivantes: le nom de l'émetteur, la ressource, l'action et le nom su destinataire. Il existe deux type de requêtes à savoir :

- la requête à exécution indirecte (différée): il s'agit ici, d'une requête qui nécessite automatiquement la validation du supérieur hiérarchique de l'employé qui initié la requête.
- la requête à exécution indirecte (réelle): il s'agit ici, d'une requête qui n'a pas besoin d'être traité par le supérieur hiérarchique de l'employé qui émet la requête.

3.3.5. Unité organisationnelle et hiérarchie organisationnelle

Dans HOr-BAC, une unité organisationnelle est définie comme étant le regroupement des unités administratives et opérationnelles. Ce qui implique qu'une unité organisationnelle peut jouer soit un rôle administrative soit un rôle opérationnel.

3.3.5.1. Unité opérationnelle

Elle représente l'ensemble des employés métiers ayant les mêmes formations, les mêmes rôles et une fonction spécifique dans une organisation. Elle ne prend aucune décision sur le fonctionnement de l'organisation et ne fait qu'obéit aux décisions qui lui ont été données par l'unité administrative qui la subordonne. Exemple : comptabilité, enseignant.

3.3.5.2. Unité administrative

Elle représente l'ensemble des unités décisionnelles de l'organisation. Cette entité permet de représenter les fonctions de contrôle, de supervision, et de validation des requêtes émises dans le SI. Elle peut être placée sur une unité opérationnelle ou sur une autre unité administrative. Exemple : département, rectorat et décanat dans une université.

Grâce à la structure organisationnelle d'une organisation, il est possible de dégager l'ordre hiérarchique entre deux unités organisationnelles.

3.3.6. Structure organisationnelle

La structure organisationnelle est définie comme étant la représentation schématique des liens hiérarchiques et fonctionnels d'une entreprise ou d'une organisation. Elle permet de préciser les niveaux de responsabilités et le mode de communication interne à l'organisation. Pour pouvoir contrôler les différentes opérations des utilisateurs du système d'information dans une organisation, nous allons ressortir la structure organisationnelle hiérarchique de cette dernière.

En effet, nous représentons cette structure hiérarchique par un arbre comportant les nœuds et les feuilles signifiant respectivement les unités administratives et les unités opérationnelles de l'organisation. Une unité opérationnelle se représente sous la forme $UO_{1,P}$ où 1 représente le premier niveau et p représente la p-ième unité opérationnelle. Par ailleurs, une unité administrative est représentée sous la forme $UA_{n,p}$ où n est le niveau hiérarchique et p la p-ième unité administrative. La figure 3.13 présente une structure organisationnelle à 5 niveaux hiérarchiques.

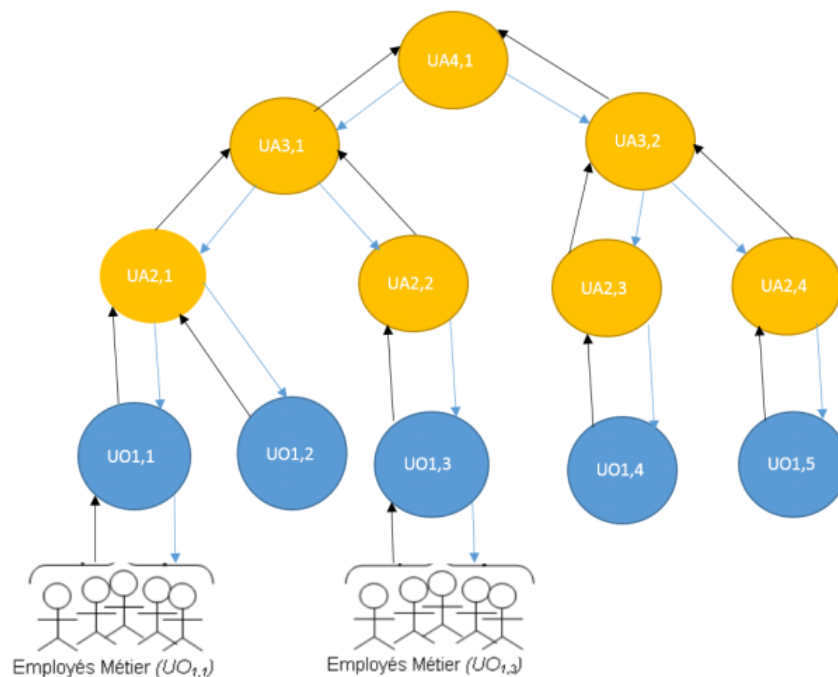


Figure 3.13 – Structure organisationnelle d'une organisation

3.3.7. Contexte

Dans notre modèle, le *contexte* définit une situation ou des circonstances dans lesquelles les organisations accordent des permissions à des rôles pour réaliser des requêtes sur des vues. Il est indépendant des employés métiers et des ressources. Tout comme l'employé métier et la ressource, un contexte peut avoir un ou plusieurs attributs. Ces attributs permettent d'exprimer des contraintes relatives aux unités organisationnelles, aux ressources, et aux employés métiers. Chaque attribut peut avoir une ou plusieurs valeurs d'attribut de contexte qui sont représentés par les entités *AttributC* et *valeurAttributC* respectivement. Dans notre modèle, on associe une ou plusieurs valeurs d'attribut de contexte à chaque contexte. Ce qui est capturé par la relation de la figure 3.14

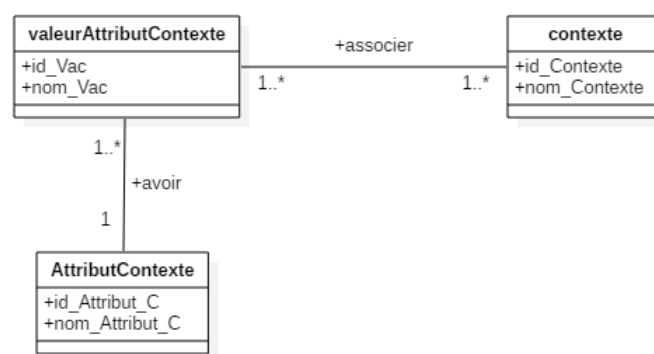


Figure 3.14 – Affectation des valeurs d'attributs de contexte aux contextes

3.3.8. Mode de traitement

Tout comme dans HOr-BAC, l'entité *Mode de traitement* dans notre modèle permet de matérialiser l'état d'urgence de traitement d'une requête. En effet, le changement d'état ressource doit être faite après validation ou non du supérieur hiérarchique de l'employé qui a émit la requête demandant accès à cette ressource.

3.3.9. Règles

Afin de permettre le contrôle des politiques de sécurité en fonction des attributs, nous introduisons dans notre modèle l'entité *Règle*. Une règle permet d'activer ou de désactiver les permissions de bas niveau. C'est-à-dire liées aux actions concrètes que réalise les employés métiers sur des ressources.

Dans AHOr-BAC, les règles qui restreignent la disponibilité des permissions pour les employés métiers peuvent être composées des valeurs d'attributs d'employé, des valeurs d'attribut de contexte et/ou des valeurs d'attribut de ressource.

3.4. Présentation des relations dans AHOr-BAC

3.4.1. Employé métier et Unité opérationnelle

La relation *Emploie* permet de matérialiser le fait qu'une organisation emploie un employé métier dans une unité opérationnelle. Cette relation est représentée à la figure 3.15.

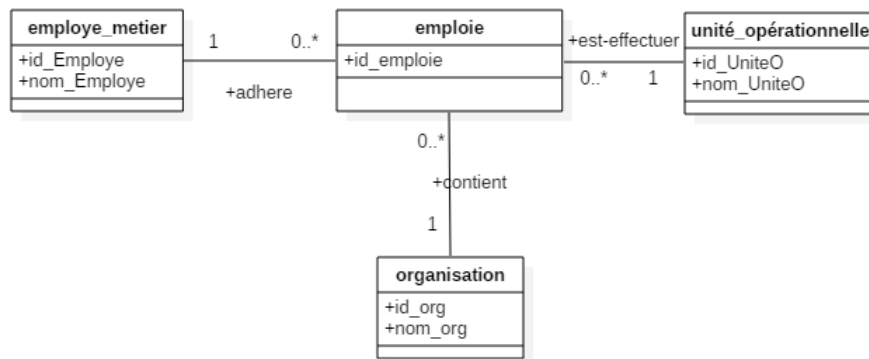


Figure 3.15 – Diagramme de classes de la relation *emploie*

3.4.2. Employé métier et Unité Administrative

La relation *Nomme* permet de matérialiser le fait que dans une organisation une unité administrative est dirigée par un employé métier. Cette relation est représentée à la figure 3.16.

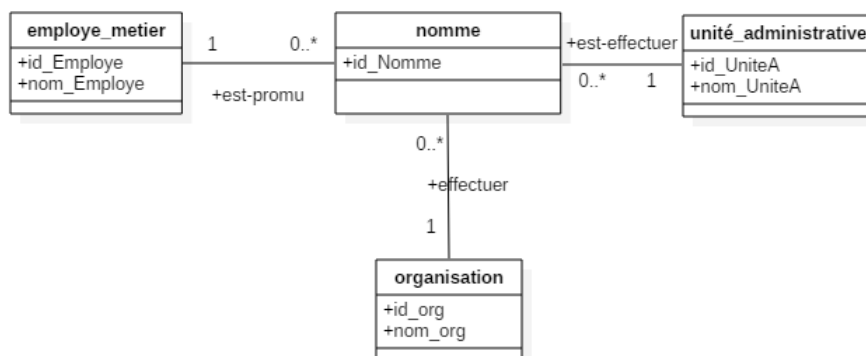


Figure 3.16 – Diagramme de classes de la relation *nomme*

3.4.3. Unité Administrative et Unité Opérationnelle

La relation *Place-sous* représente le fait que dans une organisation les rôles opérationnels sont subordonnés aux rôles administratifs. Car l'unité opérationnelle a une fonction d'exécution, alors que l'unité administrative a une fonction de contrôle.

3.4.4. Unité Administrative et Unité Administrative

La relation entre les unités administratives est matérialisée dans notre modèle par la relation *Subordonne*. En effet, dans la hiérarchie de l'organisation, en dehors du conseil

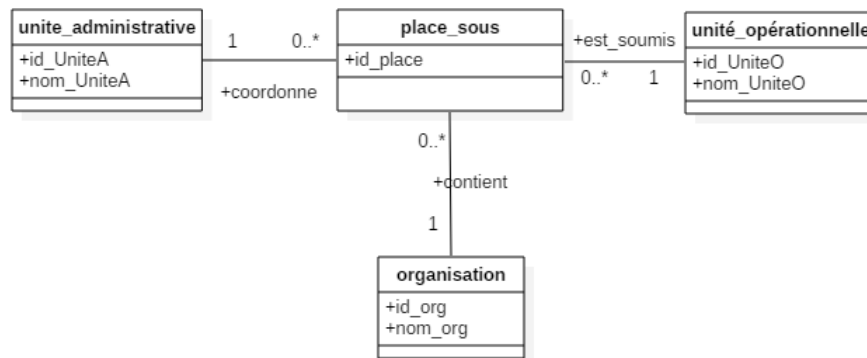


Figure 3.17 – Diagramme de classes de la relation place-sous

d'administration qui est une unité spéciale, toutes les unités administratives sont subordonnées à une autre qui assure le contrôle de ses activités. Pour cette relation il ne peut y avoir dans une organisation une unité administrative subordonnée à elle-même. Cette relation est représentée à la figure 3.18.

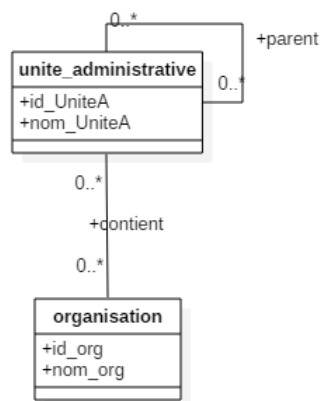


Figure 3.18 – Diagramme de classes de la relation subordonne

3.4.5. Ressources et Vues

La relation entre les unités administratives est matérialisée dans notre modèle par la relation *Utilise*. En effet, une même ressource peut être considérée de différentes manières dans une organisation en fonction des unités organisationnelles à partir desquelles cette ressource est vue. Une vue est un regroupement de ressources sur lesquelles on applique les mêmes politiques de sécurité. Suivant les organisations une même vue peut être définie différemment. Ainsi *utilise* permet de matérialiser le fait qu'une organisation utilise une ressource dans une vue.

3.4.6. Actions et Requêtes

La relation entre les unités administratives est matérialisée dans notre modèle par la relation *Considère*. En effet, les politiques de sécurité spécifient les accès aux ressources par les unités organisationnelles et régulent les actions opérées sur le système. Dans notre modèle, l'entité Action englobe principalement les actions informatiques comme "lire", "écrire", "envoyer". Et l'entité Requête englobe principalement les demandes

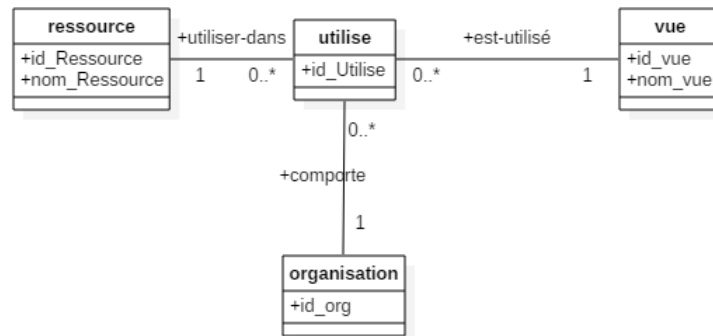


Figure 3.19 – Diagramme de classes de la relation Utilise

comme "consulter", "modifier", "transmettre", "ajouter", "supprimer". Considère permet de représenter le fait qu'une organisation considère une action comme faisant partie d'une requête.

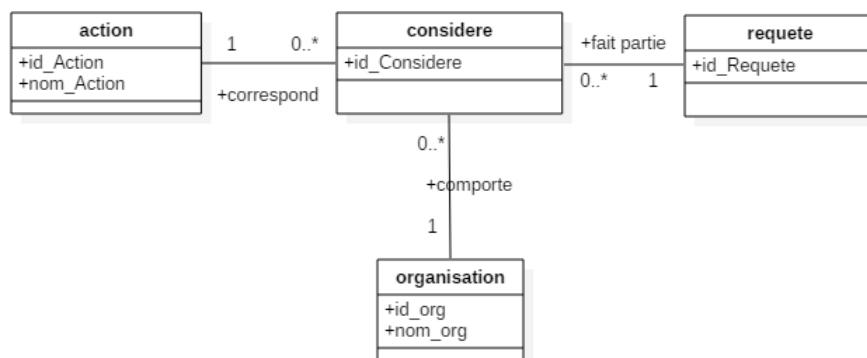


Figure 3.20 – Diagramme de classes de la relation considère

3.4.7. La relation définit

La relation *définit* permet de vérifier si un employé métier a le droit d'appliquer une requête sur une ressource et dans un contexte prédéfinie par l'organisation. Cette relation est représentée à la figure 3.21.

3.4.8. Permissions

Une permission matérialise le fait qu'une organisation autorise une unité organisationnelle de traiter ou d'émettre une requête donnée dans une vue donnée selon un contexte précis et un mode de traitement bien définie. De ce fait, nous distinguons deux types de permissions qui sont:

- **Les permissions dites de préparation:** encore appelées permissions opérationnelles. Elles permettent aux unités opérationnelles de préparer (initier, émettre ou soumettre.) des requêtes à leurs hiérarchie. Elles sont matérialisées par la relation permission-opérationnelle qui relie les entités organisation, unité opérationnelle, requête, vue, contexte et mode de traitement. Cette relation est représentée à la figure 3.22;

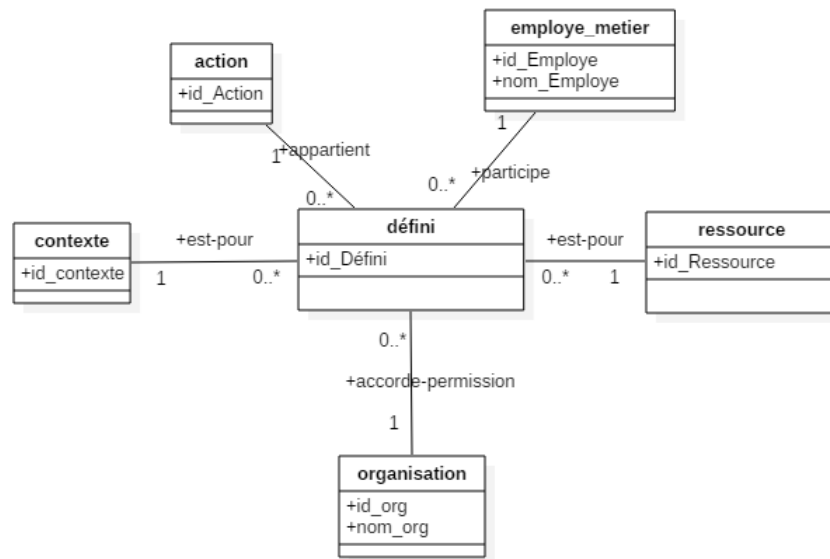


Figure 3.21 – Diagramme de classes de la relation défini

- **Les permissions dites de validation:** encore appelées permissions administratives, elles permettent aux unités administratives de traiter (contrôler, rejeter, valider et/ou transmettre) les requêtes provenant dans unités opérationnelles. Elles sont matérialisées par la relation permission-administrative qui permet de relier les entités organisation, unité opérationnelle, unité administrative, requête, vue, contexte et mode de traitement. Cette relation est représentée à la figure 3.23.

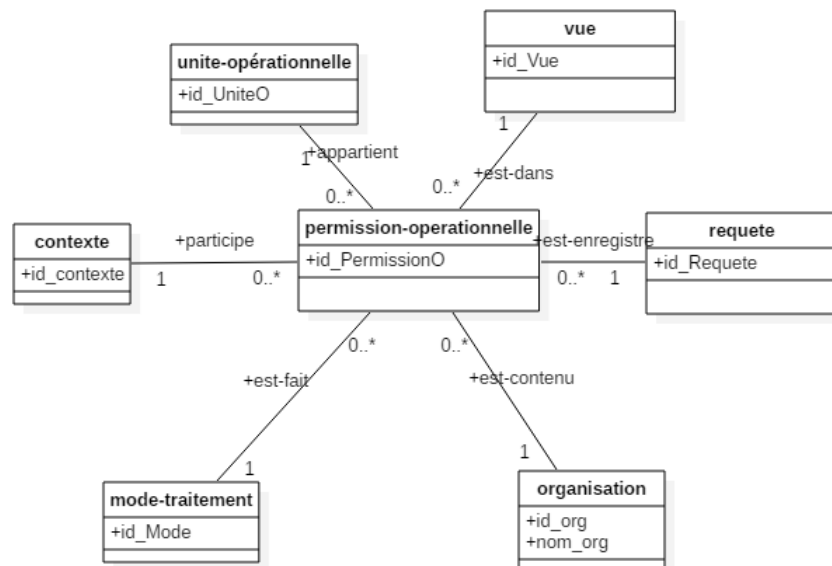


Figure 3.22 – Diagramme de classes de la relation permission-opérationnelle

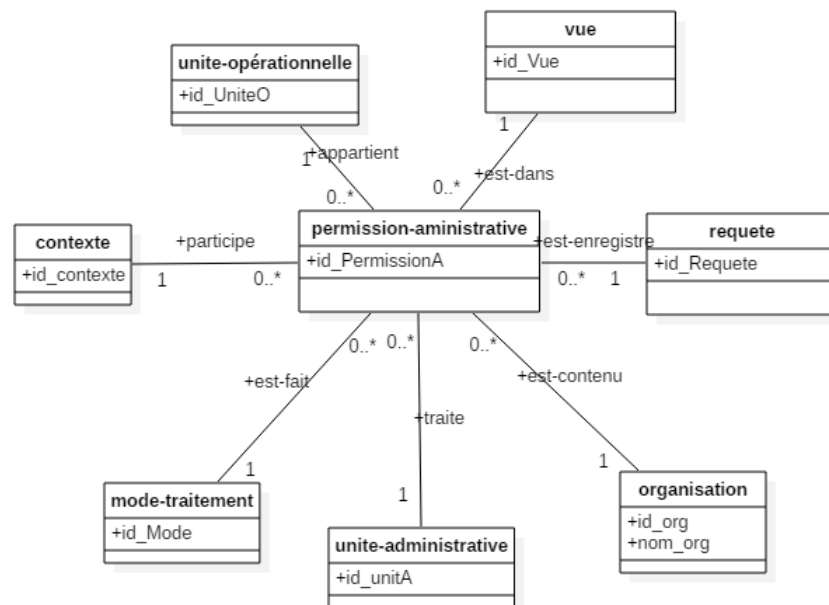


Figure 3.23 – Diagramme de classes de la relation permission-Administrative

3.4.9. Peut-suggérer et peut-traiter

Les permissions vues plus haut ne permettent qu'à une organisation donnée de spécifier les permissions accordées aux unités organisationnelles suivant un contexte précis. Mais ne permettent pas de décrire des actions concrètes que réalisent les employés sur les ressources. Ainsi nous avons implémentés le concept de contrôle d'accès de bas niveau à travers les relations suivantes :

- **Peut-suggérer** : cette relation permet à un employé d'obtenir la permission de suggérer l'application d'une action sur une ressource donnée. Cette relation est représentée à la figure 3.24;
- **Peut-traiter** : cette relation matérialise le fait que le supérieur hiérarchique d'un employé a l'autorisation de traiter les suggestions d'application d'une action donnée sur une ressource donnée. Cette relation est représentée à la figure 3.25.

3.4.10. Attribution des Règles aux Permissions de bas niveau

Dans notre modèle, une règle peut être assignée à plusieurs permissions, et une permission peut avoir plusieurs règles. Le fait que, notre modèle, associe directement un ensemble de règles basées sur les attributs aux permissions permet de réduire l'espace des règles applicables pour une permission, de minimiser le temps d'évolution de la disponibilité des permissions pour un employé métier. Car au lieu d'évaluer directement une autorisation on évolue l'ensemble des règles applicables pour cette autorisation et s'il y a une règle vraie alors la permission est accordée à l'employé. Ainsi notre approche peut restreindre les permissions.

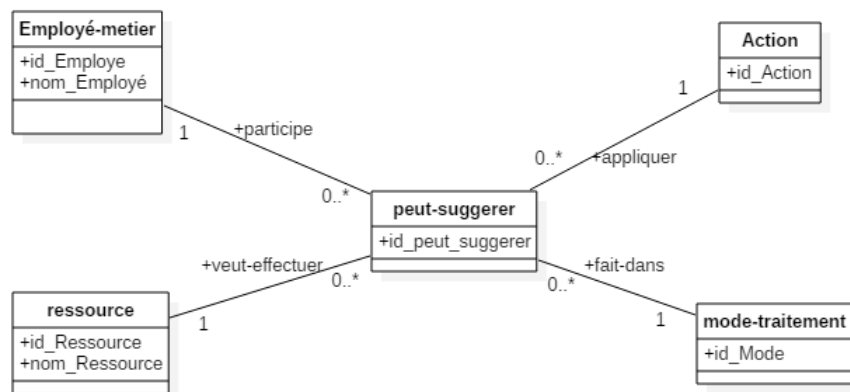


Figure 3.24 – Diagramme de classes de la relation peut-suggerer

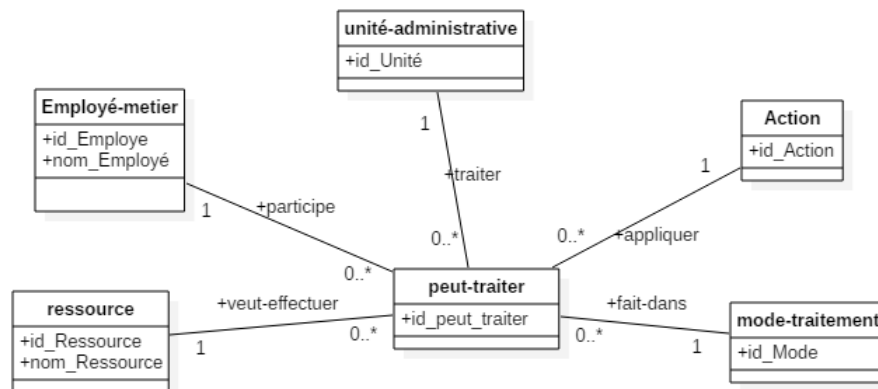


Figure 3.25 – Diagramme de classes de la relation peut-traiter

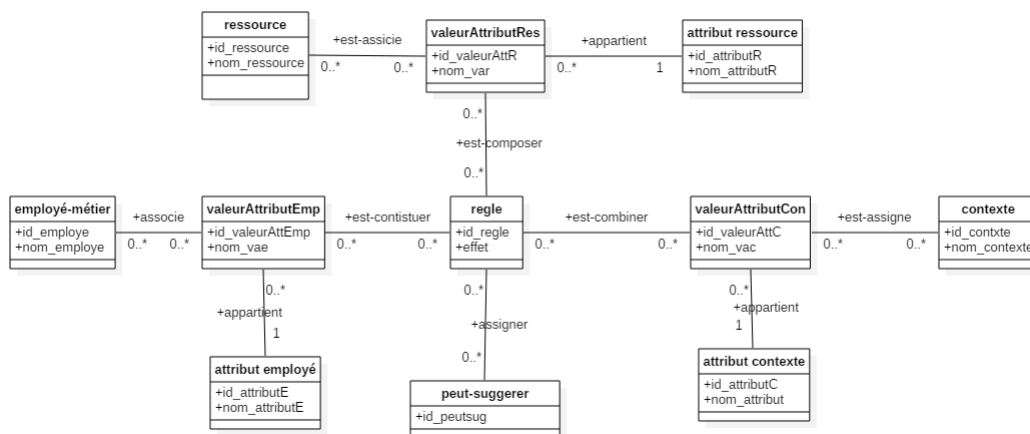


Figure 3.26 – Affectation des règles aux permissions de bas niveaux 1

3.5. Algorithme du parapheur électronique

Tout comme HOr-BAC, nous introduisons dans notre modèle la notion de parapheur électronique. Il s'agit d'un processus de traitement automatique sécurisé basé sur le mod-

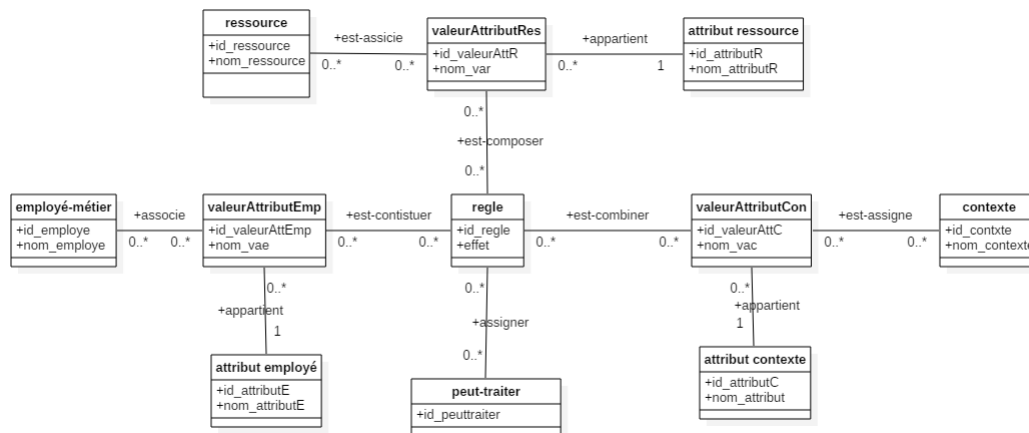


Figure 3.27 – Affection des règles aux permissions de bas niveaux 2

èle HOr-BAC. Il consiste à supposer que la quasi-totalité des actions dans une organisation se fait sur la demande et chaque demande obtient une validation pour une exécution effective dans le système sinon, la demande est rejetée. Ce processus est mis en œuvres en trois phases.

3.5.1. La phase d'initialisation du parapheur électronique

Cette phase permet la création des différentes entités du modèle HOr-BAC y compris les différentes relations qui existent entre elles. Elle est réalisée à travers six processus que sont :

- *Créer-org* Cette procédure crée la structure organisationnelle dans un arbre et retourne la racine qui est la plus grande unité administrative de l'organisation ;
- *Créer-emp* Cette procédure prend la liste du personnel et affecte chacun à une unité organisationnelle en utilisant la relation *Emploie* et *Nomme*;
- *Créer-vue* Elle prend la liste des ressources et crée les différentes vues de l'organisation, en utilisant la relation *Utilise*;
- *Créer-requête* Elle prend la liste des actions et crée les différentes requêtes de l'organisation, en utilisant la relation *Considère*;
- *Créer-mode-traitement* Cette procédure crée les différents modes de traitement utilisés dans l'organisation;
- *Créer-permission* Elle associe à chaque unité organisationnelle les Permissions opérationnelles ou les Permissions administratives suivant les cas, et donne les permissions opérationnelles aux employés métier et nomme les administrateurs. Contrairement à HOr-BAC, dans notre modèle nous définissons à ce niveau un nouveau processus qui nous permettra d'associer un ensemble de règles basées sur les attributs afin de permettre un contrôle de bas niveau et à grain fin des ressources.
- *Créer-règle-permission-bas-niveau* Elle associe à chaque permission de bas niveau un ensemble de règles en utilisant la relation *attribuer*.

3.5.2. La phase d'émission d'une requête du parapheur électronique

Cette phase est déclenché lorsqu'un employé qui veut effectuer une action sur une ressource protégée du système émet par le biais de son unité opérationnelle une requête demandant l'accès à une vue du système. Tout comme dans HOR-BAC, cette phase du parapheur électronique, dans notre modèle permet d'effectuer un contrôle de bas niveau des ressources auxquelles on souhaite y accéder. C'est-à-dire qu'on aimerait savoir si un employé métier peut ou non effectuer une action sur une ressource ou tout simplement signaler à son supérieur hiérarchique qu'il aimerait effectuer une action données sur les ressources protégées du système. La particularité de notre modèle réside dans le fait que, cette phase ne se base pas simplement sur les permissions opérationnelles pour accorder l'accès aux ressources du système, mais aussi sur des règles qui permettent un contrôle à grain fin des ressources. Ces règles comme nous l'avons spécifier plus haut sont une combinaison des valeurs d'attribut d'employé métier, des valeurs d'attribut de ressource et/ou des valeurs d'attribut de contexte.

Cette phase est capturée par un algorithme qui prend en entrée l'employé métier qui émet une requête, la requête, l'action, la vue, le contexte, le mode de traitement de la requête et un fichier contenant l'ensemble des règles d'accès. Lors de l'émission d'une requête, notre algorithme se charge de vérifier l'identité de l'émetteur de la requête. Après confirmation de l'identité de ce dernier, notre algorithme récupère l'unité opérationnelle qui émet la requête, vérifie si cette unité à une permission opérationnelle qui lui permet de réaliser la requête sur une vue donnée dans un contexte précis et selon un mode de traite de la requête défini. Si l'unité opérationnelle ne possède pas de permission alors l'accès à la ressource est refusé. Sinon, on récupère l'employé et on vérifie s'il est employé dans cette unité opérationnelle par l'organisation; si oui, on récupère la ressource de la demande dans la politique de sécurité et vérifie si celle-ci est utilisée comme vue dans l'organisation. Si cette vérification n'est pas correcte, la requête n'aboutit pas ; sinon ce processus récupère l'action de la demande dans la table action et contrôle si elle est considérée comme la requête q au sein de l'organisation. Dans le cas où il y a échec du contrôle, la demande est non valide ; dans le cas contraire, le processus continue ses vérifications en récupérant le contexte de la demande dans la politique de sécurité située dans la base de données de l'organisation et vérifie si l'émetteur a la permission de proposer une application de l'action a sur la ressource r dans le contexte c au sein de l'organisation. La demande échoue si celui-ci n'a pas cette permission et le processus d'émission continue ses vérifications. Si cette permission lui est définie, l'algorithme récupère le fichier XML qui contient les règles basée sur les attributs d'employé métier, de ressource et de contexte qui est stocké dans la base de donnée et vérifie si au plus une règle permet d'activer la permission qui donne le droit à l'employé d'effectuer une action donnée sur une ressource. Si aucune règle n'est rempli, la demande est annulée. Sinon l'algorithme récupère le mode de traitement dans la base de données. A cet effet, il existe deux modes de traitement d'une requête à savoir : les modes de traitement temps réel et temps diffère.

- Lorsque le mode de traitement est temps réel, la demande de l'émetteur n'est pas contrôlée par une unité administrative. Par conséquent le processus d'émission du parapheur électronique enregistre cette requête dans la base de données de l'organisation. Ainsi la requête préalablement émise par l'émetteur est finalement validée;

- Lorsque le mode de traitement est temps différé, la demande émise par l'émetteur doit être automatiquement validée par au moins une unité administrative de la structure organisationnelle. Alors le processus donne la possibilité à l'employé de faire une suggestion à son supérieur hiérarchique. puis la requête sera sauvegardée dans la base de donnée et une alerte sera envoyée a l'un des supérieur hiérarchique de ce dernier.

```

Emission(e:employé, a :action, r :ressource, q :requête, v :vue, c :contexte, m :mode traitement,
f :fichier XML){
  Var uo : unité opérationnelle;
  Début
    Uo<-unité(q); {uo est l'unité émettrice de la requête}
    Si permission_opérationnelle(org, uo, q, v, c, m) et
      Si emploie(org, e, uo) et
        Si utilise(org, r, v) et
          Si considère(org, a, q) et
            Si définit(org, e, a, r, c) et
              Si regle_Acces(f, e, r, c, a) alors
                Si m = immédiat alors
                  Résultat<- exécuter(a, r);
                  Sauvegarder(résultat);
                Sinon {m=différé}
                  Peut-suggérer(e, a, r, m);
                  Sauvegarder(q);
                  Alerte(q, disponible)
              Fsi
            Fsi
          Fsi
        Fsi
      Fsi
    Fsi
  Fin
}

```

Figure 3.28 – Algorithme d'émission du parapheur électronique

3.5.3. La phase de traitement d'une requête du parapheur électronique

Cette phase est déclenché lorsque le supérieur hiérarchique d'un employé métier reçoit une alerte du système lui indiquant qu'une requête vient d'être soumis par ce dernier et qu'elle attend d'être traiter par lui afin d'accorder ou pas l'accès aux ressources protégées du système. Tout comme dans HOR-BAC, cette phase du parapheur électronique, dans notre modèle permet d'effectuer un contrôle de bas niveau des ressources auxquelles on souhaite y accéder. C'est-à-dire qu'on aimerait savoir si le supérieur hiérarchique d'un employé peut ou ne peut pas autoriser à ce dernier d'effectuer une action sur une ressource ou tout simplement transmettre cette requête à une autorité au dessus de la sienne pour qu'elle valide ou refuse l'exécution de la requête par l'employé. La particularité de notre modèle réside dans le fait que, cette phase ne se base pas simplement sur les permissions administratives pour accorder l'accès aux ressources du système, mais aussi sur des règles basées sur des attributs qui permettent un contrôle à grain fin des ressources.

Cette phase est capturée par un algorithme qui prend en entrée l'employé métier qui est chargé de traiter la requête, la requête, l'action, la vue, le contexte, le mode de traitement de la requête et un fichier contenant l'ensemble des règles d'accès. Lors du traite-

ment d'une requête, notre algorithme se charge de vérifier l'identité de l'employé qui est chargé du traitement de la requête. Après confirmation de l'identité de ce dernier, notre algorithme récupère tout d'abord l'unité administrative de cet employé puis l'unité opérationnelle qui émet la requête, et vérifie si cette unité a une permission administrative qui lui permet d'autoriser à l'unité organisationnelle de l'employé qui a émis la requête de réaliser la requête sur une vue donnée dans un contexte précis et selon un mode de traitement de la requête défini. Si l'unité administrative ne possède pas de permission alors la demande est rejetée. Sinon, le processus de vérification peut continuer. Alors on récupère le supérieur hiérarchique chargé du traitement de la requête et l'émetteur de la requête et on vérifie si dans l'organisation ce dernier est le chef de l'employé qui a émis la requête. Si tel n'est pas le cas, il y a échec de la requête; sinon le processus vérifie si l'employé (supérieur hiérarchique) est nommé en tant qu'une unité administrative dans l'organisation. Dès que cette vérification est invalide, il y a échec de la requête ; dans le cas contraire, ce processus continue son exécution en récupérant la ressource de la requête dans la base de données et contrôle si cette ressource est utilisée comme vue au sein de l'organisation ; si tel est le cas, le processus récupère l'action dans la base de données et teste si cette dernière est considérée comme la requête dans l'organisation. Si tel n'est pas le cas la demande échoue.

Dans le cas contraire, il récupère le contexte de l'unité administrative autorisée à valider la requête dans la table contexte et vérifie si dans cette organisation, cette unité est autorisée à valider la requête demandant d'effectuer l'action *a* sur la ressource *r*, dans le contexte émise par l'émetteur de la requête. Si c'est le cas, l'algorithme récupère le fichier XML qui contient les règles basées sur les attributs d'employé métier, de ressource et de contexte qui est stocké dans la base de données et vérifie si au plus une règle permet d'activer la permission qui donne le droit à l'employé de valider ou de refuser l'exécution de la requête. Si aucune règle n'est remplie, la demande est annulée. Dans le cas contraire, il teste si la hauteur du destinataire de la requête ou la hauteur du supérieur hiérarchique qui traite la demande est égale à la profondeur de l'émetteur de la requête et vérifie si le mode de traitement du supérieur hiérarchique est en temps réel, la vérification étant correcte, ce dernier donne son avis qui peut être une validation ou un refus. Dans le cas où il y a acceptation, le processus effectue l'action demandée sur la ressource dans la base de données. Dans le cas où cet avis est un refus, il y a échec de la demande. Si son mode de traitement est temps différé, le processus recherche le prochain supérieur hiérarchique qui doit donner son accord sur la demande et recommence son exécution.

3.6. Conclusion

En somme, dans ce chapitre, il était question pour nous de proposer un modèle de contrôle d'accès basé sur la hiérarchie organisationnelle et les attributs. Notre modèle permet un contrôle fin des ressources protégées du système, permet de spécifier des politiques flexibles et sensibles au contexte et permet également une migration facile des politiques basées sur la hiérarchie organisationnelle. Dans le chapitre qui suit, il sera question pour nous de proposer un langage permettant de spécifier les politiques de notre modèle de contrôle d'accès.

```

Traitement(e:employé, a:action, r:ressource, q:requête, v:vue, c:contexte, m:mode traitement,
f:fichier XML)
{
    Var uo: unité opérationnelle; ua: unité administrative;
Début
    ua<-unité(e); {ua est l'unité administrée par e}
    uo<-unité(q); {uo est l'unité émettrice de la requête}
    Si permission_administrative(org, ua, q, uo, v, c, m) et
        Si chef(org, e, émetteur(e)) et
            Si utilise(org, r, v) et
                Si considère(org, a, q) et
                    Si définit(org, e, a, r, c) et
                        Si regle_Acces(f, e, r, c, a) alors
                            Si (hauteur(destinataire(q)=profondeur(émetteur(q)))(m=immédiat)) alors
                                Peut_traiter(e, émetteur(q), a, r, m)
                                Résultat<-exécuter(q, r);
                                Sauvegarder(résultat);
                            Sinon {m=différé}
                                transmission(émetteur(q), a, r, q, v, c);
                                Alerte(q, disponible)
                            Fsi
                        Fsi
                    Fsi
                Fsi
            Fsi
        Fsi
    Fin
}

```

Figure 3.29 – Algorithme de traitement du parapheur électronique

4

CHAPTER

PROPOSITION D'UN LANGAGE PERMETTANT D'EXPRIMER LES POLITIQUES D'ACCÈS BASÉ SUR LA HIÉRARCHIE ORGANISATIONNELLE ET LES ATTRIBUTS

CONTENU

4.1 - Introduction	57
4.2 - Langage de balisage extensible de contrôle d'accès	57
4.3 - Extension du langage XACML pour l'expression des politiques basé sur hiérarchie organisationnelle et les attributs	64

4.1. Introduction

Dans ce chapitre, il sera question pour nous de proposer une extension du langage XACML afin d'exprimer de façon fine granulaire des politiques de notre modèle. Notre chapitre est structuré comme suite: nous présenterons d'une part le langage XACML et ces concepts de base et d'autre part nous proposerons une extension de ce langage afin d'exprimer des règles d'accès basées non seulement sur des attributs mais aussi sur la hiérarchie organisationnelle d'une organisation.

4.2. Langage de balisage extensible de contrôle d'accès

4.2.1. Présentation de XACML

XACML (eXtensible Access Control Markup Language) est une spécification normalisée par l'OASIS [22, 4, 9]. Il s'agit d'un langage XML dédié au contrôle d'accès. Il inclut un langage protocolaire de type requêtes/réponses pour la prise de décision et un

langage pour la définition de politiques de contrôle d'accès [1].

4.2.2. Architecture du langage XACML

L'architecture XACML est composée de cinq modules logiciels clés qui fonctionnent à l'unisson pour appliquer une autorisation d'exécution normalisée à chaque point de demande d'accès.

4.2.2.1. Point d'administration des politiques (PAP)

Le point d'administration des politiques est le point de rédaction des politiques. Une fois qu'un utilisateur a écrit ou modifié/mis à jour une politique en langage clair, le PAP la convertit automatiquement en code XAML lisible par machine et basé sur des normes pour l'administration et l'application par le système.

4.2.2.2. Point d'information politique (PIP)

Le point d'information de politique est un système puissant qui appelle les différents répertoires d'attributs et services tiers au moment de l'exécution afin que le point de décision de politique établisse si la demande répond aux spécifications d'une politique. Ces soi-disant valeurs d'attribut comprenant la ressource, la source, l'environnement, etc.

4.2.2.3. Point de récupération des politiques (PRP)

Le point de récupération de politique est le point de stockage des politiques d'autorisation d'accès XACML. Il s'agit le plus souvent d'un système de fichiers ou d'une base de données.

4.2.2.4. Point de décision politique (PDP)

Le point de décision de politique évalue la demande, en fonction de ce qui est écrit dans une politique, et prend une décision – généralement Autoriser ou Refuser l'accès. Le PDP XACML informe alors le PEP de la décision.

4.2.2.5. Point d'application des politiques (PEP)

Le point d'application de la politique reçoit la demande d'accès et applique la décision d'autorisation ou de refus du PDP XACML au moment de l'exécution.

4.2.3. Flux d'autorisation XACML

Les principaux acteurs du domaine XACML sont présentés dans le diagramme de flux de données de la figure 4.31.

Le modèle fonctionne selon les étapes suivantes [22].

1. Les PAP rédigent des politiques et des ensembles de politiques et les mettent à la disposition du PDP. Ces politiques ou ensembles de politiques représentent la politique complète pour une cible spécifiée;
2. Le demandeur d'accès envoie une demande d'accès au PEP;

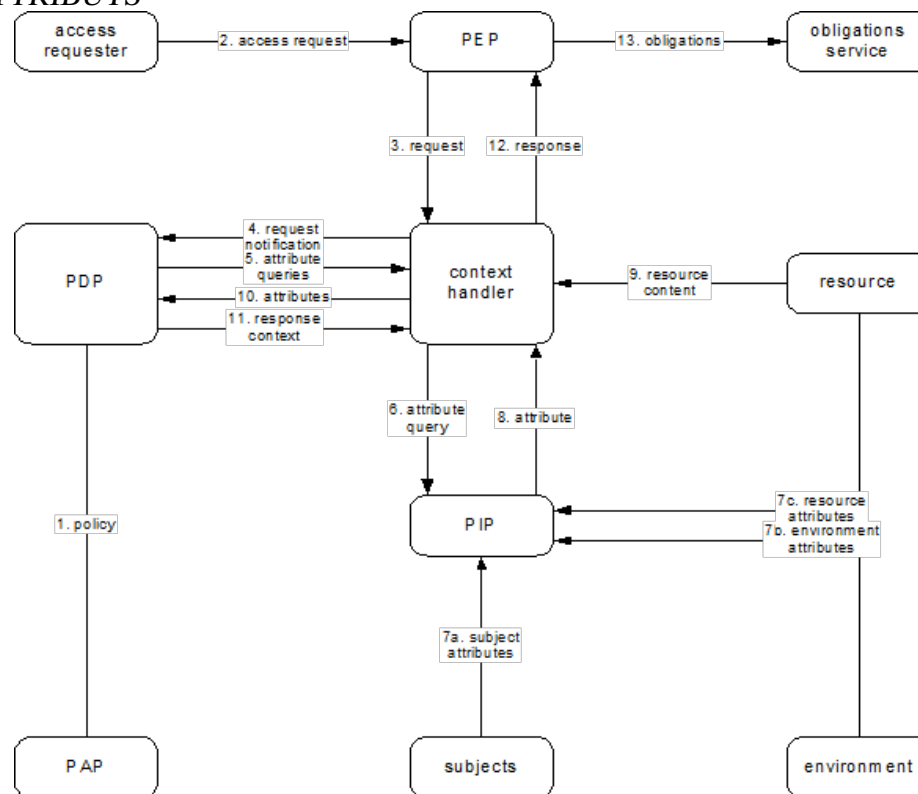


Figure 4.30 – Diagramme de flux de données [22]

3. Le PEP envoie la demande d'accès au gestionnaire de contexte « context handler » dans son format de demande natif, incluant éventuellement les attributs des sujets, ressource, action, environnement et d'autres catégories;
4. Le gestionnaire de contexte construit un contexte de demande XACML, ajoute éventuellement des attributs et l'envoie au PDP;
5. Le PDP demande tous les attributs sujet, ressource, action, environnement et autres catégories (non représentés) supplémentaires au gestionnaire de contexte;
6. Le gestionnaire de contexte demande les attributs d'un PIP;
7. Le PIP obtient les attributs demandés;
8. Le PIP renvoie les attributs demandés au gestionnaire de contexte;
9. Facultativement, le gestionnaire de contexte inclut la ressource dans le contexte;
10. Le gestionnaire de contexte envoie les attributs demandés et (éventuellement) la ressource au PDP. Le PDP évalue la politique;
11. Le PDP renvoie le contexte de réponse (y compris la décision d'autorisation) au gestionnaire de contexte;
12. Le gestionnaire de contexte traduit le contexte de réponse au format de réponse natif du PEP. Le gestionnaire de contexte renvoie la réponse au PEP;

13. Le PEP remplit les obligations;
14. (Non représenté) Si l'accès est autorisé, alors le PEP autorise l'accès à la ressource; sinon, il refuse l'accès.

4.2.4. Structure et syntaxe du langage XACML

Le langage de politique XACML est composé d'un certain nombre d'éléments clés qui permettent d'implémenter une autorisation précise sur différents modèles de déploiement, c'est-à-dire des environnements Cloud, sur site et hébergés.

4.2.4.1. Cible

XACML a introduit la notion cible « Target » afin d'identifier les règles et les politiques qui concernent une requête. La cible est composée d'attributs qui décrivent le sujet, les ressources, les actions et l'environnement:

- **Le sujet** décrit les attributs de l'utilisateur qui a fait une demande d'accès;
- **La ressource** décrit les attributs de l'objet auquel l'accès est demandé;
- **L'action** représente les attributs qui décrivent les mesures que le sujet veut prendre sur la ressource demandée;
- **L'environnement** concerne les attributs détenant des informations sur le contexte.

Chacun de ces composants est déterminé par des propriétés. Un sujet peut être défini par un identificateur, un groupe auquel il appartient, un rôle etc. Une ressource peut être caractérisée par un identificateur, des propriétés et un type. La même chose pour l'action qui peut être définie par un identificateur, le nom d'action à effectuer. Par exemple, nous considérons la requête suivante:

Un étudiant identifié par « user-id » qui appartient au groupe « A » veut accéder à un document public en mode écriture. Pour cette requête nous pouvons distinguer:

- Le sujet: user-id ;
- La ressource: document public;
- L'action: écriture.

Les propriétés des sujets, ressources et actions sont appelées attributs. Chaque attribut possède une valeur. Une cible permet au PDP XACML de vérifier quelle stratégie ou quelles règles s'appliquent à une certaine demande. Les instructions de cible agissent comme des définisseurs pour les attributs pertinents de la règle, de la stratégie ou de l'ensemble de stratégies.

4.2.4.2. Règle

Une règle de contrôle d'accès est définie avec le langage XACML comme étant un ensemble de prédicats qui répondent aux questions suivantes [6]:

- Quels sont les sujets concernés?
- Quelles sont les ressources demandées?
- Quelles sont les actions demandées?
- Quelle est la décision à renvoyer?

Une règle XACML est composée d'un effet, d'une cible et d'une condition:

- effet: détermine la décision de la règle. C'est soit « Permit » soit « Deny » ;
- cible: permet de déterminer si la règle correspond à la requête ou non;
- condition: décrite par un prédicat sur les attributs de la règle.

4.2.4.3. Politique

Une politique est exprimée par une cible, un ensemble de règles et un algorithme de combinaison [6]. Pour identifier les politiques appropriées à l'évaluation de la requête, il faut d'abord comparer la cible de la règle avec la cible de la politique, et par la suite, vérifier les conditions des règles de la politique afin de déterminer la décision « permit » ou « deny ». Il est possible que les règles d'une politique retournent des décisions différentes par rapport à une requête donnée. L'algorithme de combinaison des règles permet de spécifier comment déterminer la décision de la politique. Les décisions possibles sont [22]:

- **Permit:** l'accès est autorisé;
- **Deny:** l'accès à la ressource est refusé;
- **Indeterminate:** il n'est pas possible d'appliquer la politique à la requête parce qu'un élément (sujet, ressource, etc) est inconnu ou parce que la construction de la politique ne permet pas d'aboutir à une décision (erreur) ;
- **NotApplicable:** il n'est pas possible d'appliquer la politique à la requête car eUe ne contient aucune règle qui s'applique à la requête.

Un ensemble de Politiques (Policyset) est une agrégation de plusieurs politiques ou des ensembles de politiques. Policyset contient aussi un algorithme de combinaison pour combiner la décision de politiques.

4.2.4.4. Algorithme de combinaison

Nous rappelons qu'un algorithme de combinaison permet de calculer la décision d'une politique et d'un ensemble de politiques à partir des décisions de leurs agrégats. XACML 2.0 offre quatre algorithmes de combinaison prédéfinis [4] :

- Permit-overrides: s'il y a une règle évaluée avec effet « Permit » alors la décision de combinaison donne également un effet « permit » ;
- Deny-overrides : s'il y a une règle évaluée avec effet « Deny » alors la décision de combinaison donne un effet « Deny » ;
- First-applicable : avec l'algorithme First-applicable, l'ordre d'évaluation des règles est important. La politique prend l'effet de la première règle qui s'applique (on ignore la règle non applicable);
- Only-one-applicable : si plusieurs règles sont applicables, la décision « Indeterminate » est retournée, sinon la décision de la politique est celle de la règle applicable.

Une extension de ces algorithmes est ajoutée dans la version XACML 3.0, par exemple:

- Ordered-deny-overrides utilise le même principe que l'algorithme « Deny-override », sauf que l'ordre dans lequel les règles sont évaluées est le même que l'ordre dans lequel elles sont décrites dans la politique. La même chose pour l'algorithme « Ordered-permit-overrides »;
- Deny-unless-permit est destiné pour les cas où une décision « Permit » doit avoir la priorité par rapport à la décision « Deny » et qu'aucune règle ne retourne la décision « Indeterminate ou NotApplicable ». Cet algorithme est particulièrement utile dans une structure politique et qu'un PDP retournera toujours une décision « permit ou deny » ;
- est conçu pour les cas où une décision « Deny » devrait avoir priorité sur une décision « Permit ».

4.2.4.5. Ensemble de règles

Un jeu de règles est un groupe de règles pouvant se trouver à divers emplacements. Les ensembles de politiques comprennent des politiques, un algorithme de combinaison de politiques, des obligations facultatives et un avis.

4.2.4.6. Les conditions

Les conditions font partie d'une règle et peuvent comparer les valeurs d'attributs, pour évaluer si un attribut est "vrai", "faux" ou "indéterminé". Dans l'exemple XACML ci-dessous, vous pouvez voir le rôle d'une condition lors de la vérification si le nom d'utilisateur d'un sujet est le même que l'attribut de propriétaire d'une ressource.

4.3. Extension du langage XACML pour l'expression des politiques basé sur hiérarchie organisationnelle et les attributs

4.3.1. Insertion du parapheur électronique dans l'architecture du langage XACML

4.3.1.1. Le parapheur électronique

Dans notre langage le parapheur électronique est composé des composants suivantes:

- **PEP:** Ce composant est chargé de l'interception des requêtes des utilisateurs et de les envoyer au PDP pour une validation, puis il est chargé d'appliquer les décisions d'accès sur les ressources et d'envoyer;
- **PDP:** Le point de décision de politique évalue la demande, en fonction de ce qui est écrit dans une politique, et prend une décision. Puis informe alors le PEP de la décision.
- **Le module de Traitement Hiérarchique (Treatment Hierarchical):** Il est chargé du contrôle hiérarchie de toutes les requêtes entrantes du système et ceci après validation de ces requêtes par le PDP

4.3.1.2. Policy administration point

C'est le point de rédaction des politiques de contrôle d'accès il peut être un éditeur de politiques d'accès

4.3.1.3. Policy information point

Il a pour rôle d'extraire les informations supplémentaires qui ne sont pas présentes dans la demande d'accès. Le PIP peut lui-même chercher les informations dans des sources externes. Ces sources externes peuvent être une base de données.

4.3.1.4. Policy provider

Il est le point de stockage de politiques d'autorisation d'accès. Il peut être une base de données, un fichier XML ou encore un fichier JSON

4.3.2. Modèle de flux de données dans notre langage

notre langage fonctionne comme suite:

1. Le PAP crée les politiques d'accès et les stocke dans le Policy provider;
2. Le Policy provider met les politiques d'autorisation d'accès à la disposition du PDP;
3. L'utilisateur effectue une demande dans le système;

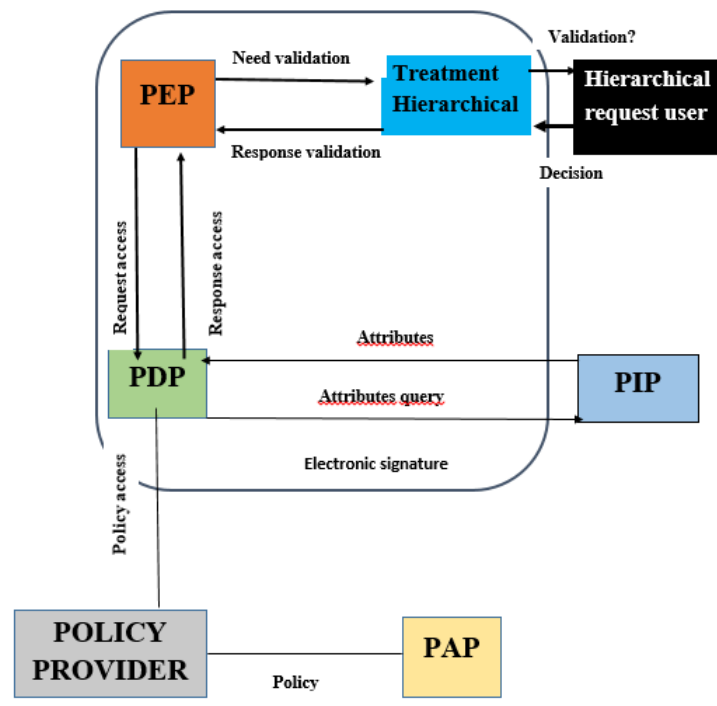


Figure 4.32 – Architecture de notre langage

4. Le PEP récupérer la demande d'accès de l'utilisateur la transforme en requête et l'envoie au PDP;
5. Le PDP demande tous les attributs sujet, ressource, action, environnement et autres catégories (non représentés) supplémentaires au PIP;
6. Le PIP obtient les attributs demandés;
7. Le PIP envoie les attributs demandés au PDP;
8. Le PDP évalue la politique;
9. Le PDP renvoie le contexte de réponse (y compris la décision d'accès, la requête et l'obligation qui contient généralement l'unité organisationnelle qui est sensé effectuer un contrôle hiérarchique sur la requête afin de donner la décision d'accès final) au PEP;
10. Le PEP enregistre la requête dans la base de données, évalue l'obligation que contient la réponse
11. Si l'obligation révèle que la requête ne nécessite pas une validation hiérarchique alors, le PEP applique la décision d'accès qui peut être accordée ou refusée;
12. Sinon le PDP envoie la requête au module de traitement hiérarchique;
13. Le module de traitement hiérarchique évalue la requête et renvoie la décision d'accès au PEP

14. Le PEP notifier l'utilisateur et applique la décision d'accès sur la politique si celle-ci est l'intersection de la décision d'accès renvoyée par le PDP et celle renvoyée par le module de traitement hiérarchique. cette décision peut être "permit" au "refusé".

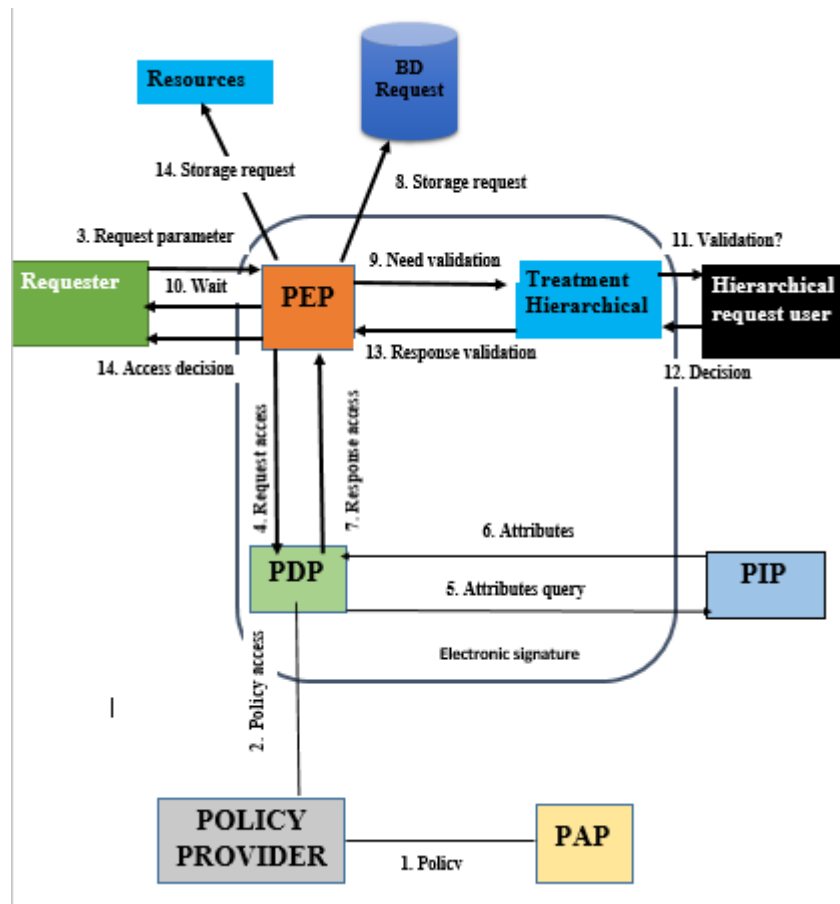


Figure 4.33 – Architecture de notre langage

CONCLUSION GÉNÉRALE

CONTENTS

Bilan	67
Quelques perspectives	67

Bilan

Dans ce mémoire, nous nous sommes intéressés à résoudre le problème de flexibilité et de granularité fine des politiques de sécurité du contrôle d'accès Basé sur la hiérarchie organisationnelle.

La résolution de ce problème nous a permis dans un premier temps de faire un état de l'art sur les modèles de contrôle d'accès existants. Il en ressort de cette revue de littérature que seul le modèle de contrôle d'accès HOr-BAC permet de contrôler le DBA au sein d'une organisation. Mais le fait que ce modèle exprime des politiques d'accès sur la base de l'unité organisationnelle à laquelle appartient un employé limite la flexibilité de ce modèle. par conséquent il ne permet qu'un contrôle à grain fin des ressources du système. Nous avons ensuite présenté une approche orienté attributs de ce modèle afin une spécification des politiques de sécurité qui permettent un contrôle à grain fin des ressources et ne sont limité que pas la puissance de calcul du langage et la taille des attributs des composants du système. Nous avons terminé ce travail par la proposition d'un langage basé sur le langage XACML qui nous permet de spécifier des politiques de contrôle d'accès basé non seulement sur l'unité organisationnelle à laquelle appartient un employé mais aussi sur les attributs.

Quelques perspectives

Les résultats de notre travail sont encourageants, mais pour conclure ils faut déployer le modèle AHor-BAC et son langage en grandeur nature sur un système existant .

BIBLIOGRAPHY

- [1] Mahamat Ahmat Abakar. *Etude et mise en oeuvre d'une architecture pour l'authentification et la gestion de documents numériques certifiés: application dans le contexte des services en ligne pour le grand public*. PhD thesis, Saint-Etienne, 2012.
- [2] Anas Abou El Kalam, Rania El Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Mieke, Claire Saurel, and Gilles Trouessin. Orbac: un modèle de contrôle d'accès basé sur les organisations. *Cahiers francophones de la recherche en sécurité de l'information*. II, 30, 43, 2003.
- [3] Muhammad Umar Aftab, Muhammad Asif Habib, Nasir Mehmood, Mubeen Aslam, and Muhammad Irfan. Attributed role based access control model. In *2015 Conference on Information Assurance and Cyber Security (CIACS)*, pages 83–89. IEEE, 2015.
- [4] Anne Anderson, Anthony Nadalin, B Parducci, D Engovatov, H Lockhart, M Kudo, P Humenn, S Godik, S Anderson, S Crocker, et al. extensible access control markup language (xacml) version 1.0. *OASIS*, 2003.
- [5] Carlo Bellettini, Elisa Bertino, and Elena Ferrari. Role based access control models. *Information security technical report*, 2(6):21–29, 2001.
- [6] Mohammed Errachid. Vérification des politiques xacml avec le langage event-b. 2011.
- [7] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.
- [8] Bill Fisher, Norm Brickman, Prescott Burden, Santos Jha, Brian Johnson, Andrew Keller, Ted Kolovos, Sudhi Umarji, and Sarah Weeks. Attribute based access control. *NIST Special publication*, page 3B, 1800.
- [9] Simon Godik and Tim Moses. Oasis extensible access control markup language (xacml). *OASIS Committee Specification cs-xacml-specification-1.0*, 2002.
- [10] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162):1–54, 2013.

- [11] Christian Jahl. The information technology security evaluation criteria. In *Proceedings-13th International Conference on Software Engineering*, pages 306–307. IEEE Computer Society, 1991.
- [12] Christian Damsgaard Jensen. *Un modèle de contrôle d'accès générique et sa réalisation dans la mémoire virtuelle répartie unique Arias*. PhD thesis, Université Joseph-Fourier-Grenoble I, 1999.
- [13] Carole S Jordan. *Guide to Understanding Discretionary Access Control in Trusted Systems*. DIANE Publishing, 1987.
- [14] Anas Abou El Kalam, R El Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Mieke, Claire Saurel, and Gilles Trouessin. Organization based access control. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 120–131. IEEE, 2003.
- [15] Hakan Lindqvist. Mandatory access control. *Master's thesis in computing science, Umea University, Department of Computing Science, SE-901, 87*, 2006.
- [16] Qasim Mahmood Rajpoot, Christian Damsgaard Jensen, and Ram Krishnan. Attributes enhanced role-based access control model. In *International Conference on Trust and Privacy in Digital Business*, pages 3–17. Springer, 2015.
- [17] Walid Rjaibi and Paul Bird. A multi-purpose implementation of mandatory access control in relational database management systems. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1010–1020, 2004.
- [18] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.
- [19] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.
- [20] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.
- [21] Mahendra Pratap Singh, S Sudharsan, and M Vani. Arbac: Attribute-enabled role based access control model. In *International Conference on Security & Privacy*, pages 97–111. Springer, 2019.
- [22] OASIS Standard. extensible access control markup language (xacml) version 3.0. A:(22 January 2013). URL: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>, 2013.
- [23] Laura P. Fotso T. Azanguezet Quimatio. *Approche hiérarchique pour la modélisation du contrôle d'accès aux données et aux traitements dans les systèmes d'information des organisation*. PhD thesis, 2016.

- [24] David Ferraiolo Vincent Hu, D.Richard Kuhn. Attributes-based access control. *Computer*, 48(2):85–88, 2015.