Année: 2018 **Thèse n°: 114/ST2I**

École Nationale Supérieure d'Informatique et d'Analyse des systèmes
Centre d'Études Doctorales en Sciences des Technologies de l'Information et de l'Ingénieur

THÈSE DE DOCTORAT

Contrôle d'accès pour les systèmes collaboratifs dans un environnement Cloud

Présentée par
Mohamed Amine MADANI

Le 24 Juillet 2018

Formation doctorale: Informatique

Structure de recherche : Equipe Traitement de l'Information et e-Stratégie

JURY

Professeur Mohammed EL KOUTBI ENSIAS, Université Mohammed V de Rabat	**Président**
Professeur Mohammed ERRADI ENSIAS, Université Mohammed V de Rabat	**Directeur de thèse**
Professeur Najib NAJA INPT, Rabat	**Rapporteur**
Professeur Abdellah JAMALI ENSA Berrechid, Université Hassan 1er, Settat	**Rapporteur**
Professeur Boubker REGRAGUI ENSIAS, Université Mohammed V de Rabat	**Rapporteur**
Professeur Ahmed KHOUMSI Université de Sherbrooke, Canada	**Examinateur**

Remerciements

Au terme de ce travail, je saisis cette occasion pour exprimer mes vifs remerciements à toutes les personnes qui ont contribué dans l'aboutissement de cette thèse, aussi bien par leurs connaissances techniques, que par leur encouragement.

Je tiens tout d'abord, à exprimer ma profonde reconnaissance à mon directeur de thèse monsieur Mohammed ERRADI, Professeur à l'Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes, pour ses directives, son encadrement et son aide théorique et technique approfondie durant cette période de thèse. J'ai énormément apprécié ses conseils, sa patience, ses qualités humaines et son soutien tout au long de mes travaux de recherche.

Je tiens aussi à remercier vivement monsieur Mohammed EL KOUTBI professeur à l'ENSIAS d'avoir accepté de présider mon jury de thèse. J'en profite pour remercier également Ahmed KHOUMSI Professeur à l'université Sherbrooke de Canada d'avoir accepté de participer à mon jury de thèse. Mes vives expressions de remerciements à Mr. Najib NAJA professeur à l'INPT, Mr. Abdellah JAMALI professeur à l'ENSA de Berrechid et Mr Boubker REGRAGUI professeur à l'ENSIAS qui ont pris la charge d'évaluer ce travail malgré leurs occupations.

Je tiens aussi à remercier vivement monsieur Yahya BENKAOUZ, Professeur à la faculté de science de Rabat pour son encadrement, son soutien et ses conseils qui m'ont été bien utiles, aussi bien pour l'aboutissement de mes travaux de recherche que pour la rédaction de ce rapport. J'adresse également mes vifs remerciements au directeur du CEDOC, le professeur Mahmoud Nassar. J'exprime également mes gratitudes aux membres du jury, pour m'avoir fait l'honneur d'accepter de juger ce travail.

Pour finir, je tiens à remercier ma grande mère Lala Amina, mes parents Houcine et Hayat, ma femme Maryam, ma charmante fille Bayane, mes beaux parents Hassan et Fatima, et mes soeurs Nayira, Samah et Wassifa pour leur soutien sincère et leurs encouragements continus, merci à tous.

Résumé

De nos jours, la collaboration entre les organisations joue un rôle important pour assurer une utilisation optimale des ressources humaines et matérielles. D'un point de vue technique et vu les facilités offertes par le cloud computing, les collaborateurs utilisent souvent des services cloud. Dans ce contexte, les applications collaboratives jouent un rôle clé pour assurer la communication, l'interaction et l'échange des ressources durant la collaboration. Au niveau du cloud, la séparation des données et des services de chaque organisation est assurée par l'adaptation du concept multi-tenant. Ce concept permet aux organisations d'utiliser mutuellement des instances d'applications ou services fournis par le cloud.

Dans un tel environnement, assurer les services de sécurité représente un défi majeur à considérer lors d'une session collaborative. Dans ce travail, nous nous intéressons précisément au contrôle d'accès aux ressources échangées lors d'une session collaborative se déroulant dans un environnement multi-tenant. Les modèles de contrôle d'accès existants pour les systèmes multi-tenants, ne supportent pas les concepts : de "session collaborative", de "multi-tenancy" et de "workflow". Pour cela, notre contribution dans cette thèse consiste à l'extension des modèles de contrôle d'accès les plus répandus à savoir : $RBAC$, $OrBAC$, $TRBAC$ et $ABAC$. Les extensions proposées ont comme objectif d'améliorer l'expressivité des politiques de contrôle d'accès à un niveau élevé de granularité ainsi que la flexibilité des modèles.

Ces modèles étendus s'appuient sur un langage formel basé sur la logique du premier ordre. Ils permettent de prendre en considération les partages provisoires des ressources entre les tenants durant une session collaborative synchrone. De plus, ces modèles permettent de : (i) supporter l'aspect dynamique de la session collaborative se déroulant dans un environnement multi-tenant ; (ii) supporter les processus collaboratifs composés d'une suite de tâches accomplies par différents tenants ; (iii) garder l'autonomie et la confidentialité des tenants durant la collaboration. Enfin, nous avons proposé une architecture d'implémentation permettant d'intégrer ces modèles étendus dans la composante de stockage du cloud Openstack, la solution open source la plus utilisée.

Mots clés : Application collaborative, Session collaborative, Cloud, Multi-tenancy, Contrôle d'accès, Openstack.

Abstract

Nowadays, collaboration among multiple organizations plays an important role to ensure an optimal utilization of human and material resources to improve productivity and profits. In this context, collaborative applications play a key role for ensuring the communication, interaction and resource sharing during a collaborative session. As most organizations rely on cloud-based solutions to store their data, cloud platforms provide a considerable convenience to support the collaboration as well as the resource sharing among organizations. Most cloud providers isolate tenants from each other using multi-tenancy in order to secure customer data and services. The multi-tenancy concept allows organizations to mutually use instances of applications or services provided by the cloud.

In such an environment, ensuring the security services is a major challenge to consider in a collaborative session. In this work, we are especially interested in access control of the shared resources during the collaboration. The existing access control models for the collaboration do not support the concepts of "collaborative session", "multi-tenancy" and "workflows that are running in a multi-tenant environment". For this purpose, our main contribution in this thesis is to extend the well-known access control models : $RBAC$, $OrBAC$, $TRBAC$, and $ABAC$. The proposed extensions are intended to improve the expressiveness of access control policies at a high level of granularity as well as the flexibility of the models.

The suggested models allow ensuring cross tenant access and synchronous collaboration between tenants. In addition, these models allow to : (i) support the dynamic aspect of the collaborative session in a multi-tenant environment ; (ii) support collaborative processes defined as a sequence of tasks ; (iii) maintain the autonomy and confidentiality of the tenants during the collaboration. Finally, the suggested models have been implemented in storage component of the open source cloud-computing platform "OpenStack".

Keywords : Collaborative application, Collaborative session, Cloud, Multi-tenancy Access control, Openstack.

Table des matières

Table des figures

Liste des tableaux

Liste des Abréviations

— ABAC : Attribute-based access control
— CHU : Centre Hospitalier Universitaire
— DAC : Discretionary access control
— HA : Hôpital d'Accueil
— IaaS : Infrastructure as a Service (IaaS)
— MAC : Mandatory access control
— MTCS-OrBAC : Multi Tenant Collaborative Session for OrBAC
— MTCS-RBAC : Multi Tenant Collaborative Session for RBAC
— MTCS-TRBAC : Multi Tenant Collaborative Session for TRBAC
— MTCS-ABAC : Multi Tenant Collaborative Session for ABAC
— OrBAC : Organization-Based Access Control
— PaaS : Platform as a Service
— RBAC : Role-Based Access Control
— SaaS : Software as a Service
— SAMU : Service d'Aide Médicale Urgente
— TBAC : TASK Based Access Control
— TCAO : Travail coopératif assisté par ordinateur
— TRBAC : TASK Role Based Access Control
— VPN : Virtual Private Network

Publications

Ce rapport présente les travaux de recherche réalisés durant la période de thèse de doctorat sous la direction du professeur Mohammed Erradi. Ces travaux de thèse ont été publiés dans des conférences et des journaux internationaux :

— Mohamed A. Madani, Mohammed Erradi, "Formalisation d'une Session Collaborative Sécurisée", the National Days of Network Security and Systems 2012, (JNS2), published at IEEE digital library, Marrakech, Maroc, Avril 2012, P. 12-17.

— Mohamed A. Madani, Mohammed Erradi, "Towards A Ssession Based Or-BAC Model", International Conference on Multimedia Computing and Systems (ICMCS'12), IEEE Conference, published at IEEE digital library, Tanger, Morocco, May 2012, P. 918-923.

— Mohamed A. Madani, Mohammed Erradi, "How to secure a collaborative session in a single tenant environment", CFIP/NOTERE, Paris, France, 2015, P. 1-6

— Mohamed A. Madani, Mohammed Erradi, Yahya Benkaouz, "Access Control in a Collaborative Session in Multi Tenant Environment", 11th International Conference on Information Assurance and Security, Marrakech, December 2015, P. 129-134.

— Mohamed A. Madani, Mohammed Erradi, Yahya Benkaouz, "A Collaborative Task Role Based Access Control Model", Journal of Information Assurance and Security, Vol. 11, no. 6, P. 358-368, 2016.

— Mohamed A. Madani, Mohammed Erradi, Yahya Benkaouz, "ABAC Based Online Collaborations in the Cloud", First International EAI Conference, AFRICATEK 2017, LNICST Springer, Marrakech, Morocco, March 2017, P. 67-76.

— Mohamed A. Madani, Mohammed Erradi, Yahya Benkaouz, "C-ABAC : An ABAC based Model for Collaboration in Multi-tenant Environment", Journal of EAI Endorsed Transactions on Smart Cities, Volume 2, Issue 8, 26th June 2018.

Chapitre 1

Introduction générale

1.1 Contexte

Actuellement, les applications et les systèmes informatiques deviennent de plus en plus collaboratifs pour répondre aux nouvelles exigences qu'impose notre vie quotidienne. Depuis de nombreuses années, plusieurs travaux de recherche ont été menés sur la spécification et la mise en œuvre des environnements de travail collaboratif [14] pour répondre aux besoins de collaboration.

Dans ce contexte, les applications collaboratives sont des applications distribuées à travers lesquelles plusieurs utilisateurs peuvent collaborer, coopérer, communiquer et interagir simultanément sur les mêmes ressources pour atteindre un objectif commun. Parmi ces applications, on peut trouver celles de télé-médecine, où un ensemble des spécialistes géographiquement distants peuvent communiquer et collaborer en temps réel, en utilisant des outils de visioconférence [19]. Ceci est dans l'objectif de faire le diagnostic à des patients situés dans des régions éloignées de grands hôpitaux.

Les utilisateurs de ces applications peuvent se connecter à une session collaborative depuis divers endroits géographiques afin de travailler sur des objets partagés (le dossier médical, l'image de scan, etc). La notion de session collaborative est un élément essentiel pour le déroulement du processus de collaboration. Il s'agit d'une entité abstraite [59], regroupant un ensemble d'utilisateurs (appelés membres de la session), effectuant des tâches spécifiques en temps réel, en accédant à des ressources partagées pour atteindre un but commun.

L'objectif de ces applications est d'offrir une variété de services à un groupe

des utilisateurs appartenant à la même ou différentes organisations à savoir : la communication instantanée à faible coût ; le travail en groupe sur le même document en manipulant les mêmes informations, pour que la même version soit utilisable par l'ensemble des utilisateurs du groupe ; le partage des ressources matérielles et humaines entre les organisations pour optimiser leur utilisation afin d'améliorer la productivité et le profit.

D'un autre coté, ces dernières années, plusieurs organisations ont externalisé leurs infrastructures, plateformes et applications informatiques chez des fournisseurs de services cloud comme : Microsoft, Amazon, IBM Google, Amazon, IBM, Dell et Salesforce. Vu les demandes croissantes du marché, chaque mois, de nouvelles offres de services Cloud apparaissent. Le Cloud Computing [40], ou informatique en nuage, est un concept informatique récent qui consiste à fournir de services informatiques (logiciels, plates-formes, matériels) sous forme de services à la demande, élastiques, évolutifs accessibles via le réseau (généralement Internet). Le cloud offre de nouveaux avantages tels que : la disponibilité des services et des données ; le coût est lié à la consommation sans aucun investissement préalable et un déploiement facile et simple.

Vu les besoins de collaboration demandés par les organisations, plusieurs fournisseurs du cloud ont proposé des services permettant aux clients de collaborer et de coopérer via une plateforme cloud. Ces services assurent la collaboration, la communication et le partage des ressources entre les organisations, à travers des sessions collaboratives se déroulant au sein d'un même cloud. En outre, le fournisseur des services cloud (Cloud Provider) sépare les données et les services de chaque organisation cliente (Cloud Customer) en locataires (Tenants) afin d'assurer leur sécurité.

Le concept "tenant" est essentiel dans le cloud [53], il est défini comme étant une instance d'une application ou d'un service fournie à une organisation ou un utilisateur par le cloud. Par exemple dans le cas d'un système de télémédecine, supposons que l'on a deux centres hospitalier universitaire CHU_RABAT et CHU_FES souhaitant stocker leurs données chez le fournisseur du cloud Amazon. Ce dernier va fournir à chaque CHU une instance d'un service de stockage des données. Ces instances TE_RABAT et TE_FES représente les tenants de stockage de CHU_RABAT et CHU_FES respectivement. Ces tenants sont bien évidemment séparés entre eux, par l'adaptation de l'architecture multi-Tenant.

Cette architecture repose sur la notion de la mutualisation des applications et des ressources.

1.2 Motivation : Application collaborative

Le travail présenté dans ce mémoire s'inscrit dans le cadre du projet « PMARS : A Secure Collaboration Platform in the Cloud for Remote Medical Diagnosis » [1]. Ce projet vise à développer un prototype d'une solution sécurisée d'expertise et d'aide à la décision pour le télé-diagnostic dans le domaine de la neurologie notamment dans les situations d'urgence. Ceci s'inscrit dans le cadre de l'INDH (Initiative nationale pour le développement humain) qui a parmi ses objectifs l'amélioration du système de santé et la prise en charge des patients dans les zones rurales.

Dans ce travail, nous considérons un scénario relatif à cette plateforme de télé-diagnostic. Dans ce scénario, le centre hospitalier universitaire (CHU) de rabat, le service d'aide médicale urgente ($SAMU$) de rabat et l'hôpital d'accueil (HA) de khemisset sont trois entités collaborant via cette plate forme collaborative de télé-diagnostic. Cette collaboration est dans l'objectif de faire le diagnostic à distance d'un patient admis aux urgences de l'hôpital de Khemissat (HA) avec des troubles de la conscience et des troubles de la motricité.

Ces trois entités partagent un cloud privé commun qui leur fournit un service de stockage pour stocker leurs données et ressources (Figure 1.1). Chaque entité utilise une instance de ce service de stockage qui correspond à un seul tenant. Dans notre exemple, on considère que :

— Le tenant HA représente l'instance du service de stockage du client HA.

— Le tenant $SAMU$ représente l'instance du service de stockage du client $SAMU$.

— Le tenant CHU représente l'instance du service de stockage du client CHU.

Ce cloud privé fournit un service de collaboration à l'entité $SAMU$. Ce service permet à un groupe d'utilisateurs, provenant de différents tenants, de collaborer et de communiquer via des outils de vidéoconférence afin d'observer et de traiter des patients de l'hôpital d'accueil. Dans cet exemple, on considère une session collaborative $CS1$ de type télé-médecine crée par un utilisateur du $SAMU$. Ce dernier peut inviter d'autres spécialistes pour qu'ils rejoignent cette session $CS1$.

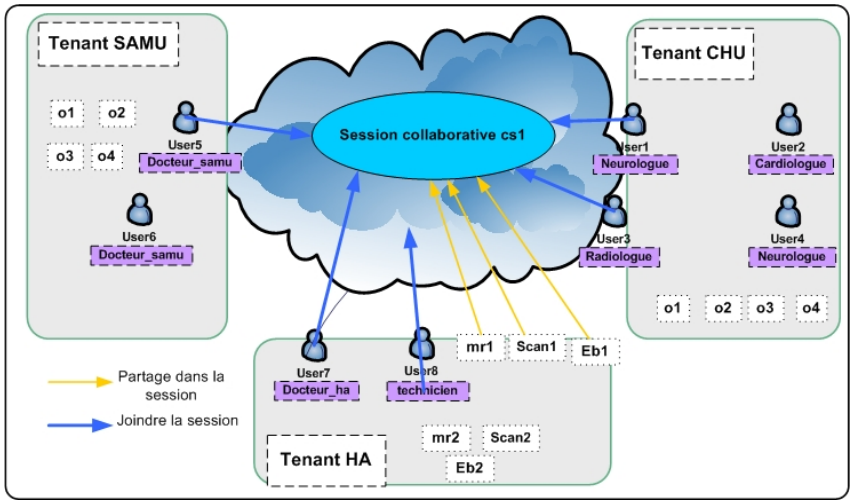

FIGURE 1.1: Scénario d'une application collaborative de télémédecine

Les spécialistes ayant accepté l'invitation deviennent les membres de la session. Par exemple, on considère que les membres de $CS1$ sont :

- $User1$ et $User4$ sont des neurologues dans le tenant CHU ;
- $User2$ est un cardiologue dans le tenant CHU ;
- $User3$ est un radiologue dans le tenant CHU ;
- $User5$ et $User6$: un médecin ($Docteur\_samu$) dans le tenant $SAMU$;
- $User7$: un médecin ($Docteur\_ha$) dans le tenant HA.
- $User8$: un technicien dans le tenant HA.

Durant la collaboration, le ($Docteur\_ha$) peut partager les ressources : dossier médical $mr1$ du patient, son image de scanner $scan1$ et son examen biologique $Eb1$ dans la session $cs1$. Une fois ces ressources sont partagées, les membres de la session peuvent y accéder selon leurs rôles et leurs permissions attribuées dans cette session. Les questions qui se posent autour de ce scénario sont : comment ces utilisateurs collaborent-ils de manière sécurisée ? et comment assurer le contrôle d'accès aux ressources partagées dans la session collaborative ? Sachant que les membres de la session et ressources partagées appartiennent aux différents tenants.

1.3 Problème

Bien que le cloud offre de nombreux avantages intéressants, de nombreuses questions se posent en effet autour du sujet de la sécurité des services et de la confidentialité des données stockées dans le cloud [52], [51]. La sécurité informatique, d'une manière générale, consiste à assurer que les infrastructures, les plateformes, les applications et les données d'une organisation soient uniquement utilisées par les utilisateurs autorisés sans un mauvais usage. Le contrôle d'accès constitue une branche de la sécurité informatique. Il permet de protéger les ressources et d'interdire les accès non autorisés à travers des politiques de sécurité.

Durant une session collaborative, les tenants participant à la collaboration, ont besoin d'accéder et d'utiliser des ressources partagées par d'autres tenants. Ces ressources contiennent souvent des données confidentielles, sensibles et à caractère personnel tel que le dossier médical d'un patient. Ces données seront partagées seulement durant des sessions collaboratives spécifiques. Le contrôle d'accès à ces ressources et la sécurité des interactions entre les tenants constituent un défi majeur auquel il faut faire face. Il est nécessaire de définir un modèle de contrôle d'accès dynamique, flexible et plus fin qui répond aux exigences liées à l'aspect dynamique de la session collaborative [30].

Dans cette thèse, nous nous intéressons en particulier au contrôle d'accès dans des sessions collaboratives se déroulant dans un environnement multi-tenant. Il s'agit de définir un modèle de contrôle d'accès qui vise à sécuriser l'échange des données et le partage des ressources entre les utilisateurs (appartenant aux différents tenants) lors d'une session collaborative [33], [34]. La définition de ce modèle doit prendre en compte un certain nombre d'exigences de sécurité tel que :

— Chaque organisation doit définir la politique de sécurité correspondant à son tenant d'une manière indépendante et autonome ;

— L'approche proposée doit répondre aux besoins des sessions collaboratives, tel que : l'accès des utilisateurs à la session et le partage des ressources dans la session ;

— La disponibilité des services offerts et l'intégrité des données utilisées doivent être garanties, notamment si l'on souhaite utiliser ces services dans une situation d'urgence ;

— La nécessité de définir des relations de confiances (Trust) entre les différents

tenants pour assurer l'accès inter-tenant ;

— Le modèle de contrôle d'accès proposé doit être adapté aux architectures multi-tenants ;

— L'aspect dynamique de la session collaborative : Lors d'une session de collaboration, les utilisateurs peuvent intervenir de façon dynamique sans aucune connaissance préalable de l'utilisateur qui va accéder à tels objets.

Les modèles de contrôles d'accès traditionnels [58] (ACL, $RBAC$, $TBAC$, $TMAC$, etc) ne permettent pas de définir des politiques d'accès pour les environnements multi-domaines. Généralement, ils sont destinés pour définir la politique d'accès locale d'une organisation donnée. D'autre part, les modèles de contrôle d'accès existants pour la collaboration et ceux pour les environnements multi-tenants (par exemple : [6], [24]) permettent d'assurer l'accès inter-domaines ainsi que l'accès multi-tenants. Cependant, ces modèles ne supportent pas le concept du partage des ressources dans une session collaborative se déroulant dans un environnement multi-tenant. De plus, ces modèles ne prennent pas en compte les processus collaboratifs (suite de tâches accomplies par différents tenants) s'exécutant dans un environnement multi-tenant.

1.4 Contributions

Dans cette thèse, nous contribuons à l'extension des modèles de contrôle d'accès traditionnels (OrBAC [26], RBAC [46], TRBAC [42] et ABAC [60]) pour qu'ils assurent le contrôle d'accès aux ressources partagées lors d'une session collaborative se déroulant dans un environnement multi-tenant. Les modèles étendus prennent en compte les points suivants : (1) la multi-tenancy et le partage des ressources entre tenants dans un environnement Cloud, c'est-à-dire un tenant accède à des ressources appartenant aux autres tenants ; (2) l'aspect dynamique de la session collaborative : d'une part, le changement des utilisateurs (Invitation, connexion et la déconnexion) lors de la session. D'autre part, le partage/ le départage des ressources durant la session collaborative ; (3) la prise en charge du concept "Tâche" vu qu'une collaboration peut être définie sous forme d'un processus composé d'une suite de tâches. Nous listons ci-dessous les extensions proposées des modèles existants, ce qui représente nos principales contributions :

— Modèle MTCS-OrBAC [35](Multi-Tenant Collaborative Session for OrBAC) est un modèle de contrôle d'accès basé sur le modèle OrBAC pour assurer le contrôle d'accès des ressources partagées lors d'une session collaborative se déroulant sur un environnement multi-tenant. L'objectif principal du modèle MTCS-OrBAC est d'étendre le modèle OrBAC pour qu'il prenne en compte le concept de la "multi-tenancy" et l'aspect dynamique de la session collaborative ;

— Modèle MTCS-RBAC [36] : (Multi-Tenant Collaborative Session for RBAC) est une extension du modèle RBAC en introduisant de nouvelles entités comme le tenant, la session collaborative ainsi que son Template. De plus, MTCS-RBAC introduit des relations de confiance établies entre deux tenants, afin de supporter le partage des ressources entre les tenants durant une session collaborative ;

— Modèle MTCS-TRBAC [37] : (Multi-Tenant Collaborative Session for TRBAC) est une extension du modèle TRBAC. Ce modèle MTCS-TRBAC est similaire au modèle précédent MTCS-RBAC, la seule différence est que le modèle MTCS-TRBAC prenne en compte le concept du « task » et « workflow » dans un environnement collaboratif Multi-Tenant. Ceci est dans le but de supporter les processus collaboratifs composés d'une suite de tâches, et dans lesquelles chaque tâche est accomplie par un tenant donné ;

— Modèle MTCS-ABAC [39] : (Multi-Tenant Collaborative Session for ABAC) est un modèle de contrôle d'accès basé sur le modèle ABAC pour assurer la collaboration entre plusieurs tenants. Ce modèle permet aux tenants participants à la collaboration de définir la politique globale de contrôle d'accès d'une manière collaborative sur un point central tout en conservant l'autonomie et la confidentialité de chaque tenant. De plus, ce modèle prend en compte les workflows collaboratifs s'exécutant sur un environnement multi-tenant ;

— Une architecture d'implémentation [38] permettant d'intégrer ces modèles étendus dans la composante de stockage du cloud Openstack. Cette implémentation nous permet de montrer la faisabilité de nos modèles et de vérifier leur cohérence et leur complétude.

1.5 Organisation du mémoire

Ce mémoire est organisé comme suit :

— Dans le chapitre 2 : Tout à d'abord, nous donnons une description détaillée des applications collaboratives distribuées, ainsi que leurs avantages. Ensuite, nous présentons une définition détaillée du concept du « Cloud computing ». Ensuite, nous décrivons les différents services du cloud et modèles de déploiement. Enfin, nous décrivons l'architecture du cloud Openstack qui est un ensemble de logiciels open source permettant de déployer des infrastructures IaaS. Enfin, nous présentons les concepts de base de la sécurité informatique à savoir les politiques de sécurité, les services de sécurité et le contrôle d'accès ;

— Dans le chapitre 3 : Nous présentons une étude bibliographique sur les différents modèles de contrôle d'accès traditionnels proposés dans la littérature. Parmi ces modèles on trouve : les politiques de contrôle d'accès discrétionnaire, les politiques de contrôle d'accès obligatoire, les politiques à base de rôles, les politiques à base de tâches, les politiques à base d'équipe, les politiques à base d'organisation et les politiques à base d'attributs ;

— Dans le chapitre 4 : Premièrement, nous présentons un état de l'art des modèles de contrôle d'accès pour les environnements Multi-organisationnels (PolyOrBAC, Multi-OrBAC, la décomposition de la politique, O2O et RT). Puis, nous montrons les avantages et les limites pour chaque modèle. Enfin, nous allons décrire les différentes approches de contrôle d'accès liés aux environnements multi-tenants, en détaillant à chaque fois, les différents avantages et limites de chaque approche ;

— Dans le chapitre 5 : Nous proposons le modèle MTCS-OrBAC [35] pour assurer le contrôle d'accès des ressources partagées dans d'une session collaborative se déroulant sur un environnement multi-tenant. Puis, nous décrivons les entités de base de notre modèle comme le tenant, la session collaborative et le type de la session collaborative. Ensuite, nous introduisons de nouvelles relations de confiance pour assurer l'accès inter-tenants. Enfin, nous décrivons l'implémentation de cette approche ;

— Chapitre 6 : Ce chapitre présente le modèle MTCS-RBAC [36] pour contrôler l'accès aux ressources partagées lors d'une session collaborative se dé-

roulant dans un environnement multi-tenant. Puis, nous présentons les différentes entités du modèle en introduisant des relations de "trust" pour établir la confiance entre les différents tenants participant à la collaboration. De plus, nous étendions cette approche pour qu'elle puisse supporter le concept du « task » et « workflow » dans un environnement collaboratif Multi-Tenant [37]. Ensuite, nous proposons le modèle d'administration. Enfin, nous décrivons l'implémentaion de ces modèles sur la composante de stockage swiftstack [3] de la plateforme Openstack ;

— Dans le chapitre 7, nous présentons les concepts de base du MTCS-ABAC qui est un modèle de contrôle d'accès basée sur ABAC pour assurer la collaboration entre multiple tenants [39]. Tout d'abord, nous introduisons la notion du tenant collaboratif. Puis, nous décrivons comment ce modèle peut supporter les processus collaboratifs s'exécutant sur un environnement multi-tenant. De plus, nous proposons de nouvelles fonctions permettant aux tenants de garder leur autonomie et leur confidentialité. Ensuite, nous présentons le modèle formel s'appuyant sur la logique du premier ordre. Enfin, nous décrivons l'architecture d'implémentation [38] ;

— Dans la conclusion, nous récapitulons les contributions apportées dans cette thèse et nous présentons les perspectives possibles relatives à ce travail.

Chapitre 2

Sécurité dans les systèmes collaboratifs à base de Cloud

Ce chapitre est composé de trois sections :

— Dans la première, nous donnons une description détaillée des applications collaboratives distribuées, ainsi que leurs avantages. Ensuite, nous présentons une définition de la notion de session collaborative.

— Dans la deuxième section, nous présentons une définition détaillée du concept du cloud computing. Puis, nous décrivons les différentes services du cloud et modèles de déploiement. Ensuite, nous discutons les avantages et les limites du cloud. Enfin, nous décrivons l'architecture du cloud Openstack qui offre un ensemble de logiciels open source permettant de déployer des infrastructures IaaS.

— Dans la troisième section, nous présentons les concepts de base de la sécurité informatique à savoir les politiques de sécurité, les services de sécurité ainsi que le contrôle d'accès.

2.1 Applications collaboratives

De nos jours, les applications et les plates-formes informatiques deviennent de plus en plus collaboratives pour répondre aux nouvelles exigences qu'impose notre vie quotidienne. Depuis de nombreuses années, plusieurs travaux de recherche ont été menés sur la spécification et la mise en œuvre des systèmes de travail coopératif assisté par ordinateur (TCAO) [14] pour répondre aux besoins liés à

la collaboration. Le travail coopératif assisté par ordinateur (TCAO) est connu aussi sous le nom de CSCW (Computer Supported Cooperative Work) qui a été introduit dans les années 1980 et développé dans les années 1990 [14].

2.1.1 Travail coopératif assisté par ordinateur : TCAO

Les systèmes (TCAO) sont des systèmes distribués à travers lesquels plusieurs utilisateurs peuvent collaborer, coopérer, communiquer et interagir simultanément sur les mêmes ressources pour atteindre un objectif commun. Ces systèmes collaboratifs permettent à une communauté d'utilisateurs ou d'organisations de partager des informations métiers nécessaires pour le déroulement des processus de collaboration. Le nom TCAO est souvent associé au terme "groupware" (ou collecticiel). Un groupware est "un type de système informatique qui permet à un groupe de personnes d'interagir sur des documents partagés à distance pour accomplir une tâche commune" [19]. Il existe plusieurs solutions sur le marché de "groupware" mais les plus répandues sont les solutions open source à savoir : KOLAB groupware, OBM Groupware, Open-Xchange, etc. Ces solutions offrent plusieurs services assurant la collaboration et le travail en groupe entre plusieurs utilisateurs ou organisations.

Dans les systèmes collaboratifs (TCAO), on entend souvent les termes communication, coopération, coordination et collaboration, dans ce qui suit, nous définissons ces différents concepts :

— La communication : est l'action d'échanger des messages entre les utilisateurs à travers l'utilisation des plates-formes de messagerie.

— La coordination : est l'action d'organiser, de coordonner et de répartir les tâches, les rôles et les responsabilités des membres d'un groupe donné. Ces responsabilités sont attribués de manière cohérente. Elle valorise le rôle de chacun pour atteindre l'objectif attendu.

— La coopération : est l'action permettant à un groupe d'utilisateurs de travailler conjointement sur le même projet en partageant des tâches entre eux. Elle valorise la production du groupe pour atteindre les résultats souhaités.

— La collaboration : est l'action de collaborer en groupe sans la répartition a priori des tâches et des rôles.

Plusieurs exemples d'applications collaboratives existent, mais les plus connues

sont les applications de télé-diagnostic médical et les applications d'Elearning. Ces applications (Elearning) permettent à un ensemble des utilisateurs géographiquement distants de suivre des formations en ligne et en temps réel, en utilisant des outils de visioconférence [19]. Ceci est dans l'objectif de faciliter l'accès à la connaissance pour les gens situés dans des régions éloignées.

2.1.2 Avantages des applications collaboratives

La puissance des applications collaboratives est le fait qu'elles offrent de nombreuses avantages à un groupe des utilisateurs appartenant à la même ou aux différentes entités à savoir : L'objectif de ces applications est d'offrir une variété d'avantages à un groupe des utilisateurs appartenant au même ou différentes organisations à savoir :

- Le faite de travailler ensemble sur le même document en manipulant les mêmes données, pour que la même version soit accessible par l'ensemble des membres du groupe collaboratif ;
- Le partage des ressources matérielles et humaines entre les organisations pour optimiser leur utilisation afin d'améliorer la productivité et le profit.
- Les utilisateurs appartenant aux différentes organisations et ayant des compétences complémentaires, peuvent travailler ensemble sur les mêmes documents en manipulant les mêmes données. Ceci est dans le but d'accomplir des tâches communes en minimisant les charges des organisations.
- Les ressources sont instantanément partagées et distribuées à l'ensemble des utilisateurs participant à la collaboration.
- L'accès distant à aux applications collaboratives permet aux utilisateurs de collaborer depuis divers emplacements.
- La communication en temps réel à faible coût ;

2.1.3 La session collaborative

Les utilisateurs des applications peuvent se connecter à une session collaborative depuis divers endroits géographiques afin d'accéder aux objets partagés (Ex, le dossier médical, l'image de scan,..). La notion de session collaborative est un élément essentiel pour le déroulement du processus de collaboration. La session

collaborative a été définie par [59] comme étant une entité abstraite, regroupant un ensemble d'utilisateurs (appelés membres de la session), effectuant des tâches spécifiques en temps réel, en accédant à des ressources partagées pour atteindre un but commun.

Durant une session collaborative, les utilisateurs peuvent avoir plusieurs rôles. Par exemple dans une session collaborative de télémédecine, un utilisateur peut jouer à la fois un rôle "créateur de la session" et un rôle neurologue. Ces utilisateurs peuvent avoir des droits d'accès différents en fonction de leurs rôles.

Une session collaborative peut adopter l'un des deux modes d'interaction suivants : synchrone ou asynchrone [29]. Dans une session asynchrone, la coprésence des différents membres n'est pas nécessaire, l'interaction se fait en mode asynchrone. Les participants communiquent à travers des outils de communication asynchrones tels que la messagerie électronique. Dans une session collaborative synchrone, la présence des utilisateurs est obligatoire en même temps. L'interaction sur les ressources partagées est faite en temps réel. Les réunions virtuelles à travers les outils de visioconférence peut être considérées comme des exemples d'une session collaborative synchrone.

2.2 Cloud Computing

Le cloud est un concept informatique apparu ces dernières années, peut offrir des services de collaboration à travers l'utilisation des sessions collaboratives. Dans ce rapport, nous nous intéressons précisément au partage des ressources entre les organisations durant une session collaborative se déroulant dans un environnement cloud.

Vu les facilités offertes par le cloud, plusieurs organisations ont externalisé leurs infrastructures, plateformes et applications informatiques chez un fournisseur de services cloud comme : Microsoft, Amazon, IBM, Google, Dell et Salesforce. Chaque mois, de nouvelles offres de services Cloud apparaissent, vu les demandes croissantes du marché.

Dans ce qui suit, nous présentons une définition détaillée du concept du cloud computing. Puis, nous décrivons les différentes services du cloud et modèles de déploiement. Ensuite, nous discutons les avantages et les limites du cloud. Enfin, nous décrivons l'architecture du cloud Openstack qui offre un ensemble de logiciels

open source permettant de déployer des infrastructures IaaS.

2.2.1 Concepts de base

Le Cloud computing est un concept qui consiste à déplacer des services informatiques traditionnellement détenus par un client sur des serveurs distants déployés chez un fournisseur de service. Ce concept se base sur l'ensemble des modèles, disciplines et technologies informatiques afin d'offrir des capacités informatiques (logiciels, plates-formes, matériels) à la manière d'un service à la demande, élastique et évolutif. Le NIST (National Institute of Standards and Technology) a proposé une définition plus précise pour ce concept Cloud Computing [40] : "modèle des systèmes d'information qui autorise un accès réseau pratique et à la demande à des ressources de calcul configurables partagées qui peuvent être mises à disposition et retirées rapidement avec un travail de gestion ou une intervention du fournisseur de service minimal".

Le cloud computing a été développé ces dernières années grâce aux fortes avancées de la technologie de virtualisation. Cette technologie permet de créer plusieurs serveurs virtuelles sur une seule machine physique afin d'optimiser les ressources matérielles. Le cloud offre plusieurs modes de déploiement dans le but de rendre une plate-forme à la demande élastique et flexible. Ces modes de déploiement sont : Le cloud privé : déployé en interne par l'organisation elle-même ; le cloud public : les services sont fournis par un prestataire externe et accessible via internet et le cloud hybrid : c'est l'utilisation des deux approches cloud privé et cloud public par une même organisation. De plus le Cloud Computing propose trois couches de services : l'infrastructure (IaaS : Infrastructure as a Service), la plate-forme (PaaS : Platform as a Service) et l'application (SaaS : Software as a Service). Ces couches seront bien détaillées dans la prochaine sous section.

Les cinq principales caractéristiques du concept Cloud ayant été définies par le National Institute of Standards and Technology (NIST) d'une manière précise [40] :

— Ubiquité : accès au cloud depuis n'importe quel endroit ;

— Un service libre à la demande ;

— Mesure de la qualité de services : évaluer et garantir un niveau de performance et de disponibilité adéquat aux besoins des clients ;

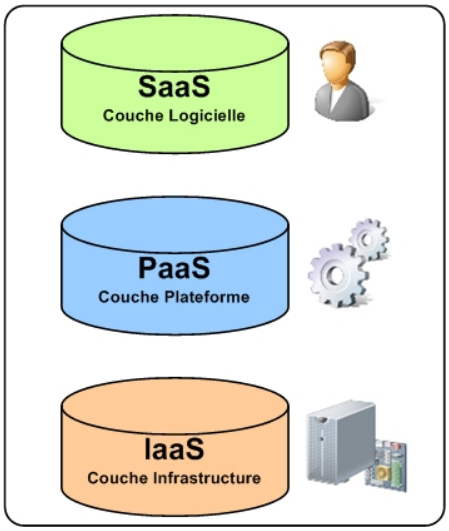

FIGURE 2.1: Différentes Couches de Cloud Computing

— Élasticité : l'action de fournir des capacités adaptées à des besoins spécifiques ;

— Mise en commun de ressources : collocation de serveurs et le partage des ressources.

2.2.2 Différentes couches du cloud

Le cloud computing offre une variété des services qui sont classés en trois couches : L'infrastructure comme un service (IaaS : Infrastructure as a Service), La plate-forme comme un service (PaaS : Platform as a Service) et L'application comme un sevice (SaaS : Software as a Service) (figure 2.1). Ces trois couches sont définis comme suit par l'agence NIST [40] :

a. Infrastructure as a Service : IaaS

C'est un modèle où l'organisation comporte une infrastructure informatique (serveurs, stockage, réseau) qui est hébergée chez un fournisseur de service. Ce concept offre une connectivité réseau et des capacités de calcul ainsi que des services de stockage. Les machines, les serveurs, les routeurs et équipements de stockage et autres équipements, sont fournis aux clients du cloud pour répondre à la charge de travail nécessaires pour le bon fonctionnement des applications. L'infrastructure comme un service, permet de fournir des ressources et des services informatiques à la demande, pour stocker des données, déployer des applications

et exécuter des traitements informatiques.

b. Platform as a Service : PaaS

C'est un modèle où l'entreprise dispose d'une plate-forme informatique qui se trouve chez un fournisseur de service. Ce modèle fournit la plate-forme comme un service (PaaS), afin de permettre aux clients du cloud de déployer leurs applications acquises ou développées en interne. Le PaaS fournit davantage un environnement de développement dédié aux développeurs des applications.

c. Software as a Service : SaaS

Le software comme un service offre une interface avec l'utilisateur final. Des applications fournies sous la forme de services (SaaS) par un fournisseur du cloud pour permettre d'utiliser les logiciels et les outils déployés sur l'infrastructure du fournisseur. Le fournisseur offre une variété de services de type SaaS, à savoir par exemple des progiciels de gestion, une application de gestion de la relation client (CRM), une solution de comptabilité, une application de gestion des ressources humaines, une solutions de messagerie, des applications collaboratives et des outils de BI (business intelligence) .

2.2.3 Modèles de déploiement

Il existe différentes modèles de déploiement du cloud à savoir [40], le cloud privé, le cloud public, le cloud hybride et le cloud communautaire :

— Cloud privé interne : hébergé et déployé en interne par l'organisation elle-même, parfois ce cloud peut être mutualisé en mode privatif avec les sous-organisations (filiales).

— Cloud privé externe : déployé chez un fournisseur, il est entièrement dédié à une seule organisation qui y accède à travers des réseaux sécurisés de type VPN.

— Cloud public : fournis et géré par un prestataire externe. Ce cloud est accessible via Internet. les ressources peuvent être mutualisées entre plusieurs organisations.

— Cloud hybride ou mixte : est une combinaison, pour une même organisation, d'un cloud privé et d'un cloud public.

— Cloud communautaire : dédié à une communauté professionnelle regroupant plusieurs organisations, sous-organisations ou sous-traitants afin d'accom-

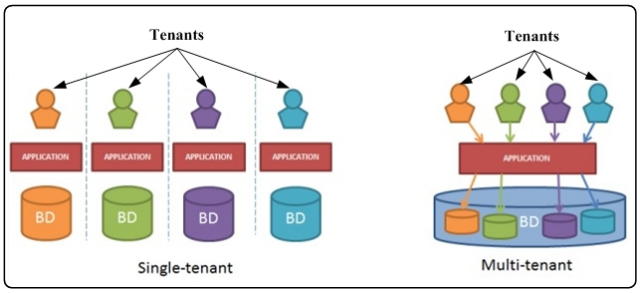

FIGURE 2.2: Single-tenant Vs Multi-tenant [40]

plir les mêmes tâches de manière collaborative.

2.2.4 Architecture multi-tenant

Le concept "tenant" est essentiel dans le cloud [53], il est défini comme étant une instance d'une application ou d'un service fourni à une organisation ou un utilisateur par le cloud. le fournisseur de services cloud (Cloud Provider) sépare les données et les services de chaque organisation cliente (Cloud Customer) en locataires (Tenants) afin d'assurer leur sécurité.

Par exemple dans le cas d'un système de télémédecine, supposons qu'un centre hospitalier universitaire souhaite stocker ses données chez le fournisseur du cloud Amazon. Ce dernier va lui fournir une instance d'un service de stockage des données. Cette instance représente le "tenant" de stockage de l'organisation CHU. Ce tenant est bien évidemment séparé des autres tenants de stockage existant chez Amazon, par l'adaptation de l'architecture Multi-Tenant(figure 2.2). Cette architecture repose sur la notion de la mutualisation des applications et des ressources, où une seule d'application va être utilisée par plusieurs clients. Par contre, dans une architecture Single-Tenant, chaque client détient ses propres matérielles et applications.

2.2.5 Avantages et limites du Cloud

Le cloud promet un très grand avenir vu qu'il offre généralement de nombreuses avantages aux clients et aux utilisateurs telles que :

— garantir la disponibilité des services et des données ;

— le coût dépend aux consommations ;

— la facilité de mettre en place une infrastructure cloud ;

— l'infrastructure technique du cloud est paramétrable est adaptable aux besoins de l'organisation ;

— Fournir des services sur mesure, sans un pré requis demandé et un investissement préalable.

Bien que le cloud offre de nombreux avantages potentiels, certaines entreprises ont peur d'adapter des solutions cloud pour les raisons suivantes :

— Sécurité : les organisations perdent le contrôle total sur leurs données et ressources ainsi que d'administration des applications.

— Connexion : Pour que les organisation accèdent à un service de cloud computing, la qualité du réseau doit être quasiment parfaite. Par ailleurs, la disponibilité d'un service dépend de la qualité du réseau.

2.2.6 Plate-forme Openstack

Dans ce travail, nous avons opté pour la plateforme open source Openstack afin de implémenter et vérifier la faisabilité des modèles propopsés. OpenStack [2] est un cloud open source regroupant un ensemble des composants permettant de mettre en place des infrastructures de cloud computing comme un service (infrastructure as a service). Openstack repose sur une architecture technologique modulaire composée de plusieurs modules corrélés (Nova, Cinder, Swift, kestone, Glance...) qui permettent d'administrer et gérer les différents services fournis tels que : le stockage des ressources et des données, les machines virtuelles, le réseau, le routage et la puissance de calcul.

Le choix de l'utilisation de la solution Openstack a été justifié pour plusieurs raisons à savoir :

— Une plateforme modulaire composée de plusieurs modules, il suffit d'installer les modules dont on a besoin ;

— Une plateforme logiciel open source : les modules s'installent gratuitement, et non pas à travers des licences d'utilisation.

La plupart des grandes entreprises d'informatique participent au développement de cette plateforme qui a été initié en juillet 2010. Openstack comporte plusieurs composants (figure 2.3). :

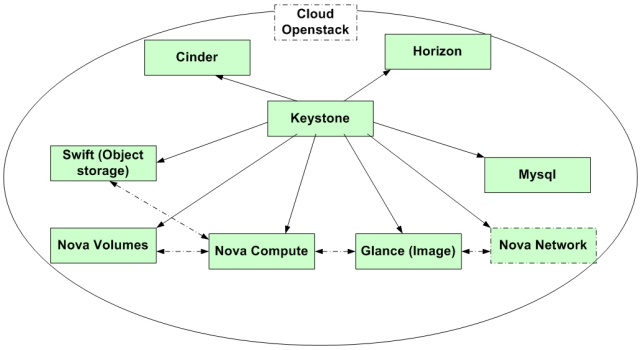

FIGURE 2.3: composants du cloud Openstack

— OpenStack Compute : Nova

— OpenStack Compute : Networking

— OpenStack Block Storage : Cinder

— OpenStack Imaging Service : Glance

— OpenStack Storage : Swift

— OpenStack Identity : Keystone

— OpenStack Dashboard : Horizon

— OpenStack DataBase : Mysql

a. OpenStack Compute : nova

La partie compute d'OpenStack, Nova, permet de communiquer avec l'hyperviseur pour lancer et gérer les machines virtuelles. Plusieurs types d'hyperviseurs sont actuellement disponibles tels que Xen, VMWare (ESX) et Hyper-V. Mais KVM est le mieux supporté par OpenStack.

Ce module nova compute permet de communiquer avec l'hyperviseur afin de gérer les instances des machines virtuelles dans un cloud OpenStack. Plusieurs types d'hyperviseurs existe actuellement sur le marché tels que Xen, KVM et VMWare (ESX). Mais Openstack utilise l'hyperviseurs KVM pour des raisons de compatibilité. De plus, Nova gère plusieurs services tels que : la planification, la création et la désactivation des machines virtuelles. D'un autre coté, nova offre une interface d'administration et une API pour la gestion des services du Cloud.

b. OpenStack Networking : Neutron

Netron permet d'assurer la connectivité réseau entre les machines virtuelles et de fournir le réseau en tant que service pour d'autres services d'OpenStack.

Ce module propose une API utilisateur pour spécifier les réseaux ainsi que les machines appartenant à ces réseaux. De plus, ce module offre de nouvelles technologies pour mettre en place une infrastructure réseau, et la possibilité d'utiliser les bridges définis dans Linux.

c. OpenStack Block Storage

Le stockage par bloc est à la base de la sauvegarde des machines virtuelles et de leurs données. Par défaut, les VM utilisent un stockage éphémère (les données stockées sont détruites en même temps que la VM). Cinder, de son côté, permet de créer des volumes persistants de stockage par bloc (block storage).

Cinder fournit un stockage par bloc pour sauvegarder les instances des machines virtuelles qui sont en cours d'exécution. Il repose sur une architecture comprenant plusieurs drivers pour faciliter la création et l'administration des blocs de stockage. Historiquement, ce composant faisait partie de Nova et se nommait nova-volume. Cinder peut également être installé et utilisé de manière indépendante de la plateforme OpenStack.

d. OpenStack Imaging Service

Glance stocke et gère des images disques utilisées par les machines virtuelles afin qu'OpenStack Nova puisse les exploiter lors du démarrage des instances. Nova communique avec Glance en utilisant son API lorsqu'une instance est déployée. Glance offre également des services de stockages pour les images disques utilisées par les machines virtuelles. De plus, ce module fournit une API compatible REST afin d'assurer la communication avec Nova une fois qu'une instance est déployée.

e. OpenStack Object Storage

Swift [3] permet de stocker des objets de données non structurées à travers l'utilisation d'une API RESTFUL basée sur HTTP. Ce module dispose d'un environnement scalable et redondant pour le stockage de plusieurs copies de données. Swift a été développé pour le stockage à long terme de grandes masses de donnée. Il stocke les ressources sur plusieurs nœuds répartis en offrant plusieurs zones d'accès pour assurer leur disponibilité en cas qu'un nœud tombe en panne.

f. OpenStack Identity

Keystone offre un annuaire comportant l'ensemble des services et des utilisateurs de la plateforme Openstack ainsi que leurs rôles et droits d'accès nécessaires pour l'utilisation de tous ces services. Il fournit une interface permettant à tous les utilisateurs d'openstack d'accéder à tous les services d'OpenStack de manière

sécurisée. Keystone repose sur un système de gestion de base de données de type MySQL.

g. OpenStack Dashboard : Horizon

OpenStack offre un tableau de bord qui se nomme Horizon. Ce dernier fournit un portail web permettant d'administrer facilement la création des machines virtuelles, des instances, de volume et d'image. Il s'agit d'un tableau de bord permettant aux administrateurs de gérer les services fournis par OpenStack à travers une interface graphique et web. Ce tableau de bord est développé en Python en utilisant les frameworks de développement web tels que Django.

2.3 Sécurité informatique

Les systèmes d'information constituent des éléments clés essentiels et stratégiques au sein des organisations. Avec le développement de technologies de l'information, de plus en plus les organisations ouvrent leurs applications informatiques à leurs collaborateurs, fournisseurs et leurs partenaires. Pour cela, il est donc nécessaire de mettre en place de mécanismes de sécurité pour empêcher l'accès non-autorisé aux systèmes et ressources d'une organisation. Afin d'assurer la sécurité d'un système, il est donc primordial de déterminer les menaces possibles qui peuvent survenir, et donc de connaître les mesures de sécurité à mettre en place pour surmonter ces menaces.

Généralement, la sécurité informatique consiste à assurer que les infrastructures, les plateformes, les applications et les données d'une organisation sont accessibles que par les utilisateurs autorisés sans un mauvais usage. La sécurité informatique devient un facteur clé dans le développement et la mise en œuvre des systèmes d'information modernes.

La sécurité informatique, d'une manière doit garantir que les autorisations d'accès aux ressources d'un système soient attribuées qu'aux utilisateurs autorisés. Ceci est assuré par l'adaptation des mécanismes d'authentification et de contrôle d'accès. De plus, ces mécanismes doivent être mis en place de telle sorte à ne pas empêcher les personnes autorisés d'utiliser les fonctionnalités d'un système.

2.3.1 Services de sécurité

Pour surmonter les failles d'un système, la sécurité informatique repose sur un certain nombre de services qui permettent d'améliorer la sécurité des systèmes d'informations, par conséquent celle des systèmes collaboratifs à base de cloud. Dans ce qui suit, on va décrire les principaux services de sécurité :

a. Disponibilité

C'est l'action de préserver le bon fonctionnement d'un système et faire en sorte que les utilisateurs autorisés puissent accéder à l'information et aux ressources dont ils ont besoin. C'est-à-dire, les utilisateurs autorisés utilisent les fonctionnalités d'un système sans un blocage. Ceci est assuré par la mise en place d'un mécanisme veillant à la disponibilité de l'information.

b. Authentification

Le service qui consiste à garantir que seules les personnes autorisées puissent accéder aux ressources. Il permet également à une entité d'assurer que l'entité avec laquelle elle communique, est bien celle qu'elle prétend représenter et vice-versa. Le service d'authentification le plus connu est le service du mot de passe. Plus précisément, l'identification est assurée par le login et l'authentification par le mot de passe.

c. Intégrité

L'intégrité consiste à garantir que le contenu d'une information ne soit pas modifié. C'est-à-dire empêcher une modification de l'information par des utilisateurs non autorisés. Il existe deux types d'intégrité :

— Intégrité des données : service qui permet de détecter les modifications partielles ou intégrales des données entre expéditeur et récepteur.

— Intégrité des flux : service qui permet de prouver qu'une transmission de données est unique, c'est à dire que la donnée n'est pas envoyée plusieurs fois.

L'intégrité est souvent assurée à travers l'utilisation des algorithmes de hachage. Enfin, la signature électronique permet d'assurer à la fois l'intégrité et l'authentification du message en utilisant un algorithme de hachage et le chiffrement. La signature électronique repose sur les architectures à clé publique.

d. Confidentialité

La confidentialité consiste à garantir que seules les utilisateurs autorisées aient

le droit d'accéder aux ressources échangées. De même, il doit empêcher les utilisateurs autorisés de divulguer une information à d'autres utilisateurs non autorisés. Il existe deux types de confidentialité :

— Des données : service qui assure la protection des données contre toute divulgation à des personnes non autorisées.

— Des flux : service qui permet de cacher l'existence de communications entre deux entités ainsi que la fréquence de ces communications.

e. La non répudiation

Ce service qui permet de assurer que l'expéditeur et le récepteur sont bien les entités qui ont respectivement envoyé ou reçu le message. La non répudiation est classée en deux types :

— La non répudiation à l'origine : pour fournir au récepteur une preuve empêchant l'expéditeur de revendiquer l'envoi d'un message effectivement reçu.

— La non répudiation à l'arrivée : pour fournir à l'expéditeur une preuve empêchant le récepteur de revendiquer la réception d'un message effectivement reçu.

2.3.2 Contrôle d'accès

Dans notre travail de thèse, on s'intéresse plus particulièrement au contrôle d'accès qui est une branche essentielle de la sécurité informatique. Il permet de protéger les ressources et d'interdire les accès non autorisés à travers la définition des politiques de contrôle d'accès. Les règles de contrôle d'accès peuvent être définies dans le système d'information ou stockées à part.

Dans un système d'information, on peut avoir des objets qui sont accessibles par tous les utilisateurs, d'autres le sont que par certains utilisateurs et d'autres sont accessibles que par leurs propriétaires. Ces besoins sont généralement spécifiés à travers des politiques de contrôle d'accès. Plus précisément, pour bien différencier les notions "modèle de contrôle d'accès", "mécanisme/système de contrôle d'accès" et "politique de contrôle d'accès", nous utilisons les définitions suivantes [18], [29]

— Un mécanisme de contrôle d'accès est un composant d'un système informatique qui permet d'assurer le contrôle l'accès aux ressources. Les utilisateurs accèdent à ces ressources selon les droits d'accès attribués par le système.

— Un modèle de contrôle d'accès est un ensemble d'entités et de relations qui représentent les éléments d'un mécanisme de contrôle d'accès. Il permet de définir la méthode de gestion d'accès aux ressources et comment les décisions d'accès seront prises ;

— Une politique de contrôle d'accès d'un système [18] est un ensemble de lois, de règles et de méthodes qui permettent de définir les actions autorisées dans ce système. La politique de contrôle d'accès formelle permet de spécifier de manière non ambigüe des règles d'accès afin de pouvoir par exemple évaluer mathématiquement les requêtes d'accès en se basant sur la formalisation de ces règles. Elle représente un formalisme qui définit ce qui est autorisé et ce qui ne l'est pas dans le contexte d'un modèle formel. Une politique de sécurité s'appuie sur :

  — La définition des ensembles des sujets, des objets, des attributs et des actions permettant d'accéder aux ressources ;

  — La définition de l'ensemble de règles pour le contrôle d'accès entre les sujets et les objets, dans le but d'empêcher l'accès non autorisé.

2.4 Conclusion

Dans ce chapitre, nous avons décrit les différents concepts qui s'articulent autour de notre thématique à savoir : les systèmes collaboratifs, les concepts de base du cloud et la sécurité informatique ainsi que ses services. Dans le chapitre suivant, nous allons étudier les différents modèles traditionnels de contrôle d'accès qui existent dans la littérature, afin de déterminer les points forts et les limites de chaque modèle. Il existe plusieurs modèles de contrôle d'accès, statiques ou dynamiques, hiérarchiques ou non, supportant les contraintes contextuelles et de séparation des pouvoirs, et d'autres modèles avec délégation ou non.

Chapitre 3

Modèles de Contrôle d'accès

Le contrôle d'accès est le service qui permet de protéger les ressources contre les accès non autorisés à travers la définition des politiques de contrôle d'accès. Il existe plusieurs modèles de contrôle d'accès, statiques ou dynamiques, hiérarchiques ou non, supportant les contraintes contextuelles et de séparation des pouvoirs, et d'autres modèles avec délégation ou non.

Parmi ces modèles on trouve : DAC, MAC, RBAC, TBAC, TMAC, OrBAC et ABAC, qui sont les plus répandus dans la littérature. Dans ce chapitre, nous allons décrire les concepts de base de chaque modèle, afin de déterminer les points forts et les limites de chaque modèle.

3.1 Modèle de contrôle d'accès discrétionnaire :DAC

Les politiques de contrôle d'accès discrétionnaire [31] (ou DAC pour Discretionary Access Control) sont basées sur les entités sujets, objets et privilèges d'accès. Les sujets correspondent aux entités actives tandis que les objets sont des entités passives. Un sujet peut être un utilisateur, un processus ou une organisation. Les politiques de contrôle d'accès sont définies dans le modèle DAC en s'appuyant sur la notions de propriété d'un objet. Autrement dit, tout sujet propriétaire d'un objet, a le droit d'effectuer n'importe quelle opération sur cet objet.

Dans ce modèle, chaque utilisateur crée une ressource, devient alors son propriétaire. Les droits d'accès à chaque objet sont définis d'une manière libre et autonome par le propriétaire de cet objet. Une action donne à un sujet l'accès direct à un objet. Elle correspond généralement aux opérations de base qu'un

sujet peut exécuter sur un objet, telles que : $lire$, $ecrire$, $modifier$ et $supprimer$.

Dans un tel système, les droits d'accès à un fichier sont définis et modifiables de manière autonome et indépendante par l'utilisateur propriétaire de ce fichier. le propriétaire d'un fichier, est celui qui détermine les utilisateurs qui ont le droit d'accéder à ce fichier. Le modèle de contrôle d'accès discrétionnaire (DAC) peut être utilisé dans le cas où l'on peut avoir confiance aux utilisateurs du système. Dans ce modèle, Les règles de contrôle d'accès s'expriment sous la forme d'une matrice M. Telle que $M(s, o)$ correspond à l'ensemble des actions que le sujet s est permis d'effectuer sur l'objet o.

Dans ce modèle, on considère l'exemple suivant : un utilisateur $user1$ est autorisé à accéder à un dossier médical $dm1$ (dont il n'est pas le propriétaire), l'utilisateur $user1$ peut créer un autre fichier $dm2$ dont il devient le propriétaire. Puis cet utilisateur donne le droit à un autre utilisateur $user2$ d'accéder au fichier $dm2$. Le seul problème qui se pose en utilisant ce modèle est que l'utilisateur $user1$ peut alors recopier le dossier médical $dm1$ dans le fichier $dm2$ pour que $user2$ puisse accéder au contenu du fichier $dm1$. Alors que cet utilisateur $user2$ n'est pas autorisé d'accéder à ce fichier. De plus, ce modèle est très static et ne permet pas de définir des politique d'accès adéquates pour des situations contextuelles. Ce modèle est vulnérable aux chevaux de Troie.

Parmi les approches de contrôle d'accès discrétionnaires les plus utilisées est la liste de contrôle d'accès [58] (ACL : Access control list). Dans cette approche (Voir la figure3.1), Chaque objet est associé à une liste ACL, indiquant pour chaque sujet du système les accès que le sujet est autorisé à exécuter sur l'objet. Par exemple, le fichier $File1$ est associé à la liste des sujets $John$, $Alice$ et Bob pour les actions (Own, R, W), (R) et (R, W) respectivement.

3.2 Modèle de contrôle d'accès obligatoire : MAC

Le Mandatory Access control (MAC) [8], [58] ou contrôle d'accès obligatoire est un modèle de contrôle d'accès dans lequel les ressources sont attribuées à des niveaux hiérarchiques de sécurité. Dans ce modèle, chaque sujet obtient un niveau d'habilitation tandis que chaque objet est affecté à un niveau de classification. Si le niveau d'habilitation d'un sujet est supérieur ou égal au niveau de classification d'un objet, alors ce sujet est autorisé à accéder à cet objet. A titre d'exemple,

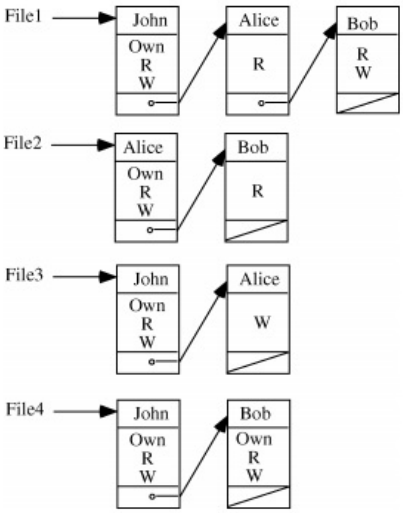

FIGURE 3.1: Liste de contrôle d'accès [58]

public, confidentiel, secret, top secret est un ensemble de niveaux de classification ordonné et hiérarchique. chaque objet sera assigné à un de ces niveaux en fonction de son degré de sensibilité.

Une politique de contrôle d'accès obligatoire contrôle non plus l'accès aux objets, mais également le flux de l'information contenue dans les objets. Ce modèle est très simple à appliquer et mettre en œuvre car il suffit de comparer le niveau d'habilitation de l'utilisateur avec le niveau de classification de la ressource afin d'évaluer les requêtes d'accès.

Il existe dans la littérature plusieurs approches implémentant le modèle MAC, le plus connu et le plus utilisé, est : le modèle de Bell et Lapadula [8] qui a été proposé par département de la défense américaine dans le but d'assurer la confidentialité et l'intégrité des données. Une politique de contrôle d'accès obligatoire de type MAC peut être combinée avec celle de contrôle d'accès discrétionnaire, vu qu'elles sont complémentaires.

3.3 Modèle de contrôle d'accès à base de rôle : RBAC

Le modèle RBAC [20], [21], [46] (Role Based Access Control) est une technique intéressante pour la gestion de droits d'accès pour les systèmes informatiques. Ce

modèle permet de modéliser les fonctions métiers ainsi que les autorisations d'accès à des ressources d'une organisation donnée. Dans ce modèle, les permissions d'accès ne sont pas attribuées directement à des utilisateurs comme dans les modèles DAC et MAC. Le modèle RBAC introduit la notion de rôle. Le rôle est une entité intermédiaire entre les permissions et les utilisateurs (par exemple, médecin généraliste, neurologue, infirmier, etc sont des exemples de rôles).

Le modèle RBAC permet de structurer d'une manière hiérarchique l'ensemble des utilisateurs appartenant à une organisation. Dans ce modèle, les permissions d'accès sont attribuées aux rôles pas aux utilisateurs, tandis que les utilisateurs sont affectés aux rôles. Alors que les utilisateurs obtiennent les permissions d'accès selon leurs rôles attribués. Le principe de base de ce modèle est que deux utilisateurs qui ont le même rôle ont les mêmes privilèges dans le système. Les entités de base du modèle RBAC définies dans [46] sont les suivantes(Figure 3.2) :

— Utilisateurs : Sont les utilisateurs interagissant sur le système ;

— Rôles : sont affectés aux utilisateurs en se basant sur la fonction attribuée à ces utilisateurs dans l'organisation ;

— Objets : sont les objets à protéger et sont accessibles à travers les autorisations attribuées à un rôle ;

— Opérations : sont les opérations possibles à effectuer sur les objets

— Permissions : autorisation d'effectuer une opération donnée sur un objet donné ;

— Sessions : une instance d'une activité d'un seul utilisateur, chaque session est associée à un utilisateur pour une période du temps limitée.

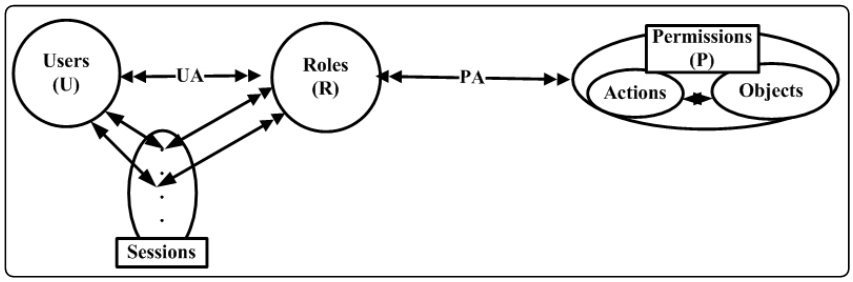

FIGURE 3.2: Modèle RBAC

Chaque utilisateur est affecté un ou plusieurs rôles. une fois il se connecte à la session, un ou plusieurs rôles lui seront activés. Par conséquent, un ensemble

des permissions lui sont attribuées selon ces rôles activés. Une permission est une opération sur un objet. Cependant, Ce modèle reste statique, il manque la flexibilité et la souplesse pour définir des règles comportant des contraintes contextuelles. Dans la littérature, il existe plusieurs extensions du modèle RBAC. Parmi ces extensions, on trouve :

a. Modèle RBAC hiérarchique

Une extension du modèle RBAC [47], il ajoute au modèle de base la notion de hiérarchie de rôles. Un ensemble de permissions est toujours associées à un rôle mais ce dernier peut hériter en plus d'un ou plusieurs autres rôles. A cet égard, un utilisateur affecté à un rôle $R1$, reçoit les permissions associées à ce rôle ainsi que celles associées aux rôles qui lui sont hiérarchiquement inférieurs. Ceci permet de faciliter l'administration et de réduire le nombre de relations entre les rôles et les permissions.

b. Modèle RBAC contrainte

Une extension du modèle RBAC [5], permettant d'intégrer des contraintes sur l'association utilisateur-rôle. Ces contraintes peuvent être des règles de cardinalité et séparation de pouvoirs statiques et dynamiques ainsi que les contraintes d'exclusion mutuelle qui peuvent être prise en compte dans la définition des rôles. Par exemple, même si certains rôles peuvent être attribués à un utilisateur donné, ces rôles ne peuvent pas être activés simultanément dans la même session.

C. Modèle RBAC temporel Le modèle RBAC temporel (Temporal-RBAC) [10] permet d'ajouter au modèle de base RBAC la prise en compte de la dimension temporelle pour supporter les contraintes temporelles. Ceci est conçu en s'appuyant sur des actions qui sont exécutées automatiquement en fonction du temps. Ces actions peuvent être associées à un rôle, dans le but d'activer un rôle (role enabling), ou de désactiver un rôle (role disabling) durant un intervalle de temps.

Avantages du modèle RBAC :

D'après l'étude faite sur le modèle RBAC, on peut déduire que ce modèle est facile à utiliser et à administrer. En outre, il est très simple à mettre en oeuvre en répondant aux besoins de chaque organisation. En utilisant RBAC, il est facile d'ajouter un utilisateur : il suffit de lui attribuer le rôle qu'il va jouer selon les fonctions qu'il exerce dans l'organisation. Par ailleurs, il n'est plus nécessaire créer de nouvelles associations (de type utilisateur et permission) à chaque fois qu'un

nouvel utilisateur est ajouté au système. De même, il est aussi facile d'intégrer des nouvelles permissions : il suffit d'associer ces permissions aux rôles concernés.

Le modèle RBAC permet de simplifier l'administration de la politique d'accès pour mieux maitriser la gestion des privilèges d'accès [43]. En utilisant le modèle RBAC hiérarchique, il est tellement primordial d'organiser les rôles de manière hiérarchique. C'est à dire, un rôle héritant des autorisations des rôles qui lui sont hiérarchiquement inférieurs. Par exemple, le rôle médecin spécialiste gérite toutes les permissions accordées au rôle médecin généraliste.

Inconvénients du modèle RBAC :

Le principal inconvénient du modèle RBAC est le fait que tous les utilisateurs jouant le même rôle possèdent forcément les mêmes privilèges d'accès. Par exemple, en utilisant RBAC, il est difficile de définir les règles d'accès de type « Un neurologue n'ait accès qu'aux dossiers médicaux des patients qu'il traite à distance». Pour définir cette règle, il faut soit définir autant de rôles « neurologue traitant du patient X » que de patients. Ceci va rendre les tâches d'administration de la politique d'accès plus complexes vu que l'on aura un nombre important des rôles. Un autre inconvénient est qu'il n'est pas possible dans le modèle RBAC de définir des politiques d'accès spécifiques à des situations contextuelles. Enfin, RBAC ne permet pas de supporter le contrôle d'accès dans un environnement collaboratif comportant plusieurs organisations, car ce modèle est quasi statique et destiné uniquement pour définir la politique de contrôle d'accès d'une seule organisation.

3.4 Modèle de contrôle d'accès à base de tâche : TBAC

Le modèle TBAC [56] (Task Based Access Control) est le premier modèle qui a introduit la notion de tâche. Ensuite, il a été conçu pour contrôler l'exécution des activités dans un système de workflow. Le modèle TBAC est une extension des modèles traditionnels de contrôle d'accès à base d'objet/sujet (Par exemple les modèles DAC et MAC) en ajoutant la notion des domaines qui contiennent des informations contextuelles à base des tâches. La politique de contrôle d'accès dans ce modèle est basée sur le principe des tâches. Les permissions sont attribuées en fonction de l'état d'avancement des tâches.

Par exemple dans un tel système, dès qu'on arrive à une tâche X un ensemble de permissions relatives à cette tâche X seront activées et de même si l'on bascule à la tache Y. Chaque étape est associée à un état de protection contenant un ensemble d'autorisations. Le modèle TBAC est un modèle souple permettant une gestion dynamique des autorisations relatives aux tâches. En outre, chaque autorisation a une période temporelle d'exécution et des contraintes d'expiration. De même, le modèle TBAC ne permet pas de prendre en compte le contrôle d'accès dans un système collaboratif contenant plusieurs entités, car ce modèle est destiné uniquement pour définir la politique de contrôle d'accès d'une seule entité.

3.5 Modèle de contrôle d'accès à base d'équipe : TMAC

Le modèle de contrôle d'accès à base d'équipe (TMAC : Team-based Access Control) a été conçu par K.Thomas [57]. L'objectif est de proposer un modèle de contrôle d'accès convenable à un contexte collaboratif en intégrant les éléments forts du modèle (RBAC). L'entité de base de TMAC, équipe ou "team", est une abstraction qui regroupe un ensemble d'utilisateurs, qui jouent des rôles différents et qui coopèrent dans le but d'atteindre un objectif commun. Cependant, les permissions accordées à chaque utilisateur sont définies par le rôle qu'il joue et l'activité courante de l'équipe.

Le concept TMAC repose sur la notion d'attribution des permissions et d'activation des permissions. Les permissions sont attribuées à un utilisateur selon le rôle qui lui est activé une fois qu'il se connecte à la session. Les permissions activées sont celles qui dépendent des paramètres contextuelles (lieu, temps, etc.). Cependant, le concept d'équipe est représenté comme une entité intermédiaire entre l'utilisateur et le contexte. Dans le modèle TMAC, le contexte de collaboration d'une équipe donnée doit tenir compte de deux types de contexte :

— Contexte utilisateur : les utilisateurs qui sont membres de l'équipe à un moment donné ;

— Contexte objet : les instances d'objets que l'équipe utilise pour réaliser sa tâche.

Le modèle TMAC a été défini en utilisant la formalisme ci-dessous :

- T est l'ensemble des équipes,
- TU, les membres d'une équipe ;
- TR, l'ensemble des rôles associés à une équipe ($TR \subseteq R$), où R désigne l'ensemble des rôles possibles définis dans le système ;
- $h \subseteq TR$, représente le rôle "responsable de l'équipe" ;
- OT, un ensemble d'objets associés à l'équipe ;
- TP, un ensemble de permissions attribuées à l'équipe ($TP \subseteq TR \times OT$).
- UC, un contexte-utilisateur ($UC \subseteq TR \times TU$) ;
- OC, un contexte-objet ($OC \subseteq OT \times O$), tel que O désigne l'ensemble des objets.

Le modèle TMAC propose une idée intéressante basée sur la notion d'équipe. Ceci est dans le but de définir des politiques de contrôle d'accès adaptées pour une application collaborative utilisée par une seule organisation. Par contre, il n'est pas possible dans le modèle TMAC de définir des politiques d'accès spécifiques à un environnement collaboratif comportant plusieurs organisations.

D'un autre coté, si un utilisateur est membre d'une équipe, il obtient toutes les permissions associées à cette équipe. Ceci se contredit avec le concept de la session collaborative, où l'on considère que les membres d'une session n'ont pas forcément les mêmes privilèges durant la collaboration. De plus, la notion d'équipe définie dans TMAC est statique, vu que l'appartenance des membres à une équipe est fixée au préalable par l'administrateur de la politique. Contrairement à la session collaborative, où les membres changent d'une manière dynamique. En outre, une session collaborative est créée seulement pour une durée limitée et sera détruite une fois la collaboration est clôturée.

3.6 Modèle de contrôle d'accès à base d'organisation : OrBAC

Le modèle de contrôle d'accès OrBAC [26], [17] (Organization Based Access Control) vise à surmonter les limites des modèles de contrôle d'accès existants et à concevoir une politique de contrôle d'accès plus abstraite. Il est basé sur les

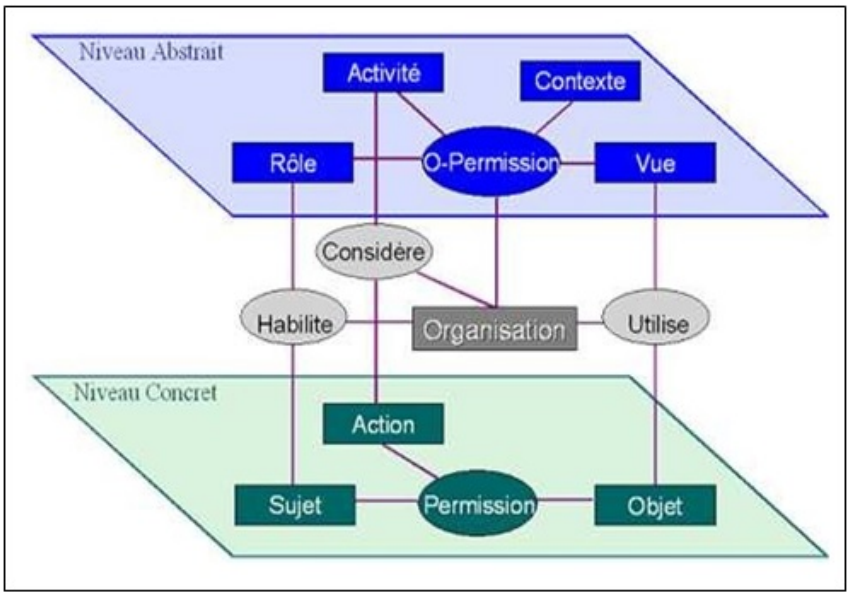

FIGURE 3.3: Modèle OrBAC [33]

mêmes principes du modèle RBAC en intégrant de nouvelles entités. Il reprend les concepts de rôle, d'activité, de vue et de contexte. L'idée principale est d'exprimer la politique de sécurité avec des entités abstraites et de séparer la spécification de la politique de contrôle d'accès de son implémentation.

Dans le modèle OrBAC, l'expression d'une politique d'accès est centrée sur la notion d'organisation (contrairement à RBAC où la politique d'accès est centrée sur le la notion de rôle). Une organisation est un groupe structurée d'entités actives. Dans la plateforme collaborative de télé-diagnostic (décrite dans l'introduction), le service d'aide médical d'urgente (SAMU) est un exemple d'organisation. OrBAC organise les sujets, les objets et les actions respectivement en rôles (comme dans RBAC), vues et activités. Les concepts de rôles, de vues et d'activités sont des concepts organisationnels. Chaque organisation définit ainsi les rôles, les activités et les vues des objets auxquels elle doit contrôler l'accès à travers la définition d'une politique d'autorisation (voir figure 3.3). Le modèle OrBAC introduit non seulement les permissions, mais aussi les interdictions et les obligations et les recommandations. Le modèle OrBAC définit aussi la notion de contexte comme une situation spécifique dans laquelle une règle est valide. Il définit des permissions par la règle de la forme $Permission(org, r, v, a, c)$, où org est une organisation, r est un rôle, v est une vue, a est une activité et c un contexte. De la même manière, on définit les interdictions $Interdiction(org, r, v, a, c)$, les obliga-

tions $obligation(org, r, v, a, c)$ et les recommandations $recommandation(org, r, v, a, c)$.

a. organisations

L'entité centrale dans le modèle OrBAC est l'organisation. Une organisation peut être vue comme un groupe structuré d'entités actives, c'est-à-dire des utilisateurs jouant certains rôles. Dans le l'exemple de l'application de télé-diagnostic, nous pouvons considérer les organisations suivantes : l'hôpital d'accueil, le centre hospitalier universitaire (CHU) et le service d'aide médical d'urgente ($SAMU$). De même, les établissements, les entreprises ainsi que les organismes publics, sont des exemples d'organisations.

b. Concept rôle dans l'organisation

Dans le modèle OrBAC, les rôles sont affectés aux sujets en fonction de leur fonction dans l'organisation. Un sujet peut être soit une entité active, c'est-à-dire un utilisateur, soit une autre organisation. Par exemple, $user1$, $user2$, etc., peuvent être des sujets, tout comme les organisations $SAMU$. Dans ce modèle, la notion Rôle est utilisée pour exprimer la relation entre les sujets et les organisations. Dans notre scénario, les rôles $Docteur\_ha$ et $technicien$, sont joués par des utilisateurs de l'organisation HA alors que le rôle $Docteur\_samu$ est définie le $SAMU$. De plus, le modèle OrBAC introduit une relation entre les sujets, les rôles et l'organisation : la relation $Habilite()$.

— $Habilite(org, s, r)$, signifie que l'organisation org habilite le sujet s à jouer le rôle r.

— $Habilite(CHU, mohamed, neurologue)$ signifie que l'organisation CHU affecte $mohamed$ au rôle $neurologue$.

c. Concept vue dans l'organisation

D'après la définition des rôles dans le paragraphe précédent, les rôles sont des entités abstraites qui permettent de représenter les sujets ayant une fonction commune dans l'organisation. De la même manière, les vues sont définies comme étant des entités abstraites qui permettent de représenter les objets ayant une propriété commune dans l'organisation. Alors les vues représentent la manière dont les ressources sont utilisés dans l'organisation. De même, le modèle OrBAC introduit une relation entre les objets, les vues et l'organisation : la relation $Utilise()$

— $Utilise(org, o, v)$, signifie que l'organisation org utilise l'objet o dans la vue v.

— $Utilise(\text{CHU}, MR, dossier\_medical)$: signifie que l'organisation CHU utilise le fichier MR comme un dossier médical.

d. Concept activité dans l'organisation

Dans le modèle OrBAC, l'entité Action comporte principalement les actions informatiques (les opérations) définies sur les objets comme $lire$, $modifier$, $ecrire$, $supprimer$, $creer$, etc. De la même manière que les rôles et les vues sont des entités abstraites des sujets et des objets respectivement, l'activité est définie comme étant une entité abstraite des actions. Alors une activité représente les actions (opérations) qui ont un but commun dans l'organisation. Le modèle OrBAC définit une relation entre les actions, les activités et l'organisation : la relation $Considere()$ est définie comme suit :

— $Considere(org, a, A)$, signifiant que l'organisation org considère que l'action a fait partie de l'activité A.

— $Considere(CHU, select, consultation)$: l'hôpital CHU considère que l'action $select$ comme une activité consultation (dans cet exemple, on parle d'une consultation informatique et non pas d'une consultation médicale).

e. Concept contexte dans l'organisation

Le modèle OrBAC introduit la notion du contexte pour exprimer les circonstances et les conditions concrètes dans lesquelles une organisation accorde un autorisation à un rôle d'effectuer des activités sur des vues. Dans le domaine médical, la notion du contexte permet de spécifier des situations telles que l'urgence. De même, le modèle OrBAC définit une nouvelle relation entre l'organisation, le sujet, l'objet, l'action et le contexte appelée $Definit()$:

— $Definit(org, s, a, o, c)$, exprime que dans l'organisation org, le contexte c est vrai pour le sujet s, l'objet o et l'action a.

— $Definit(CHU, user1, lire, MR, urgence)$, signifie que dans l'organisation CHU, le contexte $urgence$ est vrai pour le triplet :
$(user1, lire, MR)$

f. Politique de sécurité dans le modèle OrBAC

Dans le modèle OrBAC, la relation $Permission$ est définie par les entités organisation, rôle, vue, activité et contexte. Si org est une organisation, r est un rôle, a est une activité, v est une vue et c est un contexte, alors $Permission(org, r, a, v, c)$ signifie que org accorde au rôle r l'autorisation d'effectuer l'action a sur la vue v

dans le contexte c. Une permission de contrôle d'accès est définie dans le modèle OrBAC comme suit :

$$\begin{aligned} &\forall r \forall a \forall v \forall c \; Si : permission(org, r, a, v, c) \wedge \\ &Habilite(org, s, r) \wedge \\ &Utilise(org, o, v) \wedge \\ &Considre(org, x, a) \wedge \\ &Definit(org, s, x, o, c) \\ &\rightarrow Est\_permis(s, x, o) \end{aligned} \tag{3.1}$$

Les relations Interdiction, Obligation et Recommandation sont définies de la même manière que la relation permission.

3.7 Modèle de contrôle d'accès à base d'attribut : ABAC

Le modèle ABAC a été introduit par Eric Yuan [60], dans le but de répondre aux exigences de contrôle d'accès liées aux architectures web services. En effet, les accès à des ressources au niveau de ces architectures web services doivent être contrôlés en se basant sur un modèle de contrôle d'accès plus fine et dynamique. Les modèles traditionnels que nous avons vu dans les sections précédentes, sont généralement statiques, car ils se basent sur des attributs fixes. Ceci ne permet pas de définir des autorisations spécifiques pour des situations contextuelles. Par contre, ABAC [25] est un modèle plus adapté pour les environnements collaboratifs et les architectures web services, en permettant d'exprimer des règles dynamiques et contextuelles.

Le modèle ABAC définit les autorisations d'accès d'un utilisateur à une ressource pour une action donnée en se basant sur les attributs de cet utilisateur, cette ressource et l'environnement dans lequel cette action s'exécute. A cet effet, le modèle ABAC proposé dans [25] s'appuie sur trois types d'attribut :

— Les attributs d'utilisateur : un utilisateur est l'entité qui demande d'accéder à un objet. Dans ABAC, chaque utilisateur est identifié et caractérisé par un ensemble d'attributs d'utilisateur. Par exemple, l'âge, la fonction, le grade et l'affectation d'un utilisateur peuvent aussi être considéré comme

des attributs.

— Les attributs d'objet : Un objet est l'entité à laquelle la politique de sécurité contrôle son accès à partir des utilisateurs du système. Un objet peut être des données, des documents, des fichiers, des répertoires et des périphériques. De même, chaque objet est caractérisé par un ensemble des attributs définissant son identité et sa nature. Le type d'objet, le nom de l'objet, son auteur, son propriétaire, la date de création et son extension sont des exemples des attributs d'objet.

— Les attributs d'environnement : l'environnement est caractérisé par un ensemble des paramètres contextuelles décrivant la situation dans laquelle l'accès à la ressource s'effectue. De même, l'environnement est associé à un ensemble d'attributs définissant une situation spécifique et des circonstances concrètes. La date d'accès, type de réseau et l'outil utilisé sont des exemples des attributs d'environnement.

Dans ce qui suit, nous allons décrire formellement le concept de base du modèle ABAC proposé dans [25] et [60]

— U, O et E représentent des ensembles finis d'utilisateurs et d'objets et d'environnements respectivement. A est un ensemble fini d'actions, on note $A = \{create, read, update, delete\}$.

— $UATT$, $OATT$ and $EATT$ représentent des ensembles finis de fonctions d'attribut utilisateur, objet et environnement respectivement.

— Pour chaque attribut $att \in UATT \cup OATT \cup EATT$, $range(att)$ représente la plage de l'attribut, qui est un ensemble fini de valeurs atomiques.

— $attType : UATT \cup OATT \cup EATT \rightarrow \{set, atomic\}$, détermine le type d'attribut, il peut être ensemble ou des valeurs atomiques.

— Chaque fonction d'attribut mappe des éléments en U à une valeur atomique ou un ensemble.

  — $\forall ua \subseteq UATT.\, ua : U \rightarrow Range(ua)\, if\, attType(ua) = atomic$

  — $\forall ua \subseteq UATT.\, ua : U \rightarrow 2^{Range(ua)}\, if\, attType(ua) = set$

— Chaque fonction d'attribut mappe des éléments en O à une valeur atomique ou un ensemble :

  — $\forall oa \subseteq OATT.oa : O \rightarrow Range(oa)\, if\, attType(oa) = atomic$

  - $\forall oa \subseteq OATT.oa : O \rightarrow 2^{Range(oa)}\ if\ attType(oa) = set$
- Chaque fonction d'attribut mappe des éléments en E à une valeur atomique ou un ensemble :
  - $\forall ea \subseteq EATT.ea : E \rightarrow Range(ea)\ if\ attType(ea) = atomic$
  - $\forall ea \subseteq EATT.ea : E \rightarrow 2^{Range(ea)}\ if\ attType(ea) = set$
- Une autorisation du modèle ABAC est définie comme étant une fonction booléenne qui prend en paramètres les attributs de u, o et e et retourne $true$ ou $false$. Elle retourne $true$ si l'utilisateur u est autorisé à effectuer l'action a sur l'objet o. Sinon, cette fonction retourne $false$. D'autre part, une politique de contrôle d'accès est définie comme étant un ensemble d'autorisations regroupant plusieurs utilisateurs et plusieurs objets au sein d'une même organisation. Une autorisation d'accès d'un utilisateur u à un objet o pour une action a dans un environnement e est exprimée par le formalisme ci-dessous : $authorization_a(u, o, e) \leftarrow f(ATTR(u), ATTR(o)\ , ATTR(e))$.

Par exemple, $can\_accessa(u, o, e) \leftarrow (Role(u) = Neurologue \wedge ObjType(o) = Scan \wedge MemberCS(u) = CS1)$: Cette autorisation est valide si seulement si : (1) l'utilisateur u joue le rôle *neurologue* ; (2) l'objet o est de type $Scan$; et (3) l'utilisateur u est membre de la session collaborative $CS1$.

Parmi les avantages du modèle ABAC, on peut citer :

- Le modèle ABAC prend en compte la notion du contexte à travers la définition des attributs d'environnement. Pour cela, ABAC permet de définir des politiques d'accès plus flexibles qui peuvent s'adapter aux différentes situations contextuelles.
- Dans le modèle ABAC, il est possible de spécifier des politiques d'accès basée sur d'autres modèles de contrôle d'accès en utilisant ABAC. A titre d'exemple, on peut définir des politiques d'accès RBAC et TBAC en considérant que les entités « rôle » et « tâche » respectivement, sont des attributs du modèle ABAC.
- Les politiques d'accès dans le modèle ABAC repose sur la notion des attributs qui permet de fournir une granularité très fine pour définir les permissions d'accès. Pour tenir en compte un nouveau paramètre ou une nouvelle condition dans la définition de la politique d'accès, il suffit d'intégrer de nouveaux attributs correspondant à cette condition ou ce paramètre.

Le modèle ABAC permet de définir des politiques d'accès adaptées aux sessions collaborative se déroulant dans une seule organisation : C'est-à-dire, les utilisateurs membres de la session et les ressources partagées dans la session appartiennent à la même organisation. Cependant, il n'est pas possible en utilisant seulement le modèle ABAC, de définir des politiques d'accès spécifiques aux sessions collaboratives se déroulant dans un environnement collaboratif multi-organisationnelle. Vu que le modèle ABAC est destiné seulement pour définir la politique d'accès locale de chaque organisation et non pas pour pour les environnements multi-organisationnels.

3.8 Conclusion

Dans ce chapitre, nous avons étudié et analysé l'état de l'art des modèles de sécurité les plus répandus (TBAC, RBAC, OrBAC, ABAC). Ces modèles de contrôle d'accès classiques ne permettent pas de définir des politiques d'accès adaptées à un environnement multi-tenant collaboratif. L'objectif de cette étude était de déterminer un modèle de contrôle d'accès que nous pourrions étendre pour définir une politique de contrôle d'accès adaptée pour les systèmes multi-tenants collaboratifs. Dans le chapitre suivant, nous allons présenter un état de l'art des modèles de contrôle d'accès pour les environnements multi-organisationnels et multi-tenants.

Chapitre 4

Etat de l'art des modèles de contrôle d'accès pour des environnements multi tenants collaboratifs

Dans ce chapitre, nous présentons un état de l'art des modèles de contrôle d'accès pour la collaboration : O2O [15], Multi-OrBAC [28], PolyOrBAC [27], décomposition de la politique [32]. Puis, nous montrons les avantages et les limites pour chaque modèle. Ensuite, nous allons voir les différentes approches de contrôle d'accès liés aux environnements multi-tenants, en détaillant à chaque fois, les avantages et limites de chaque approche. Enfin, nous allons faire une étude comparative de différentes approches de contrôle d'accès décrites dans ce chapitre.

4.1 Modèles de contrôle d'accès pour la collaboration

Dans la littérature, il existe plusieurs modèles de contrôle d'accès liés aux systèmes collaboratifs. Les modèles existants se classent en deux catégories principales : les approches centralisées et les approches décentralisées (pair ou pair).

Dans les modèles centralisés, la politique de contrôle d'accès globale de la

collaboration se définit et s'applique sur un point central qui impose la politique à tous les autres collaborateurs [49], [61]. L'approche centralisée (Voir figure 4.1) de contrôle d'accès consiste à définir les règles d'accès globales des entités qui collaborent A et B par une autre entité centrale C (super entité). Les entités collaboratrices doivent faire confiance à l'entité C dans la gestion de la politique d'accès et les décisions d'accès.

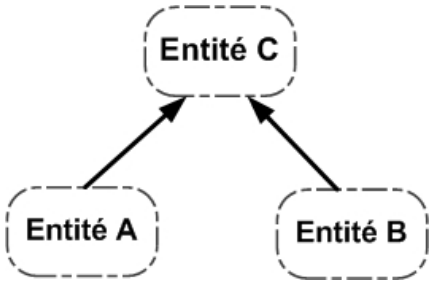

FIGURE 4.1: Architecture centralisée

Tandis que, dans les modèles décentralisés, la définition de la politique globale de contrôle d'accès et les décisions d'accès se font d'une manière autonome par chaque entité participant à la collaboration. Pour cela, la politique d'accès des entités qui collaborent, se définit en intégrant la politique d'accès globale de collaboration dans la politique locale de chaque entité (voir figure 4.2). Ce type d'approches permet à chaque entité de contrôler sa propre politique d'accès librement, tout en gardant son autonomie.

FIGURE 4.2: Architecture décentralisée

Dans cette section, nous allons décrire les concepts de base des deux approches (modèle ROBAC [62] et approche de décomposition [32]) adaptant l'architecture centralisée et les modèles (VO, O2O, Multi-OrBAC et PolyOrBAC) adaptant l'architecture décentralisée.

4.1.1 Modèle ROBAC

Les auteurs de [62] ont proposé un modèle évolutif de contrôle d'accès basé sur RBAC pour prendre en compte les applications B2B (Business to Business)

et B2C (Business to Consumer). Ces applications impliquent souvent un grand nombre d'organisations. Le modèle proposé nommé ROBAC (Role and Organization Based Access Control) s'appuie sur les deux entités "Rôle" et "Organisation", contrairement au modèle RBAC, où la politique d'accès est uniquement centrée sur l'entité "Rôle".

L'idée principale du modèle ROBAC est assez simple. Un utilisateur est affecté au couple (rôle, organisation) au lieu de rôle uniquement. De plus, les permissions dans ROBAC sont définies comme des opérations sur des types d'objets au lieu d'opérations sur des objets. Un utilisateur peut accéder à un objet si et seulement si l'utilisateur est affecté au couple rôle et organisation. Ce rôle a le droit d'accéder au type de cet objet et cette organisation est propriétaire de cet objet.

Dans ce travail, les auteurs ont proposé une série des modèles ROBAC à savoir : ROBAC0, ROBAC1, ROBAC2 et ROBAC3. ROBAC0 est le modèle de base. ROBAC1 est une extension du ROBAC0 en intégrant la hiérarchie des rôles et la hiérarchie d'organisation. De même, ROBAC2 est une extension du modèle ROBAC0 en prenant en considération les contraintes. Le modèle ROBAC3 est une combinaison entre les modèles ROBAC1 et ROBAC2. ROBAC0 comporte les composants suivants (voir figure 4.3) :

— U est un ensembles des utilisateurs ;

— S est un ensemble des sessions ;

— R est un ensemble des rôles ;

— O est un ensemble des organisations ;

— Op est un ensemble des opérations ;

— A est un ensemble des objets (Assets) ;

— At est un ensemble des types d'objet (Asset types) ;

— $P \subseteq Op \times At$, est un ensemble des permissions ;

— $RO \subseteq R \times O$, un ensemble de relations d'association entre rôles et organisations ;

— $PA \subseteq P \times R$, est une relation plusieurs-à-plusieurs mappant un rôle à un ensemble des permissions ;

— $UA \subseteq U \times RO$, est une relation plusieurs-à-plusieurs mappant un utilisateur à un ensemble des couples (rôle, organisation) ;

— $user : S \rightarrow U$, est une fonction mappant une session si à un seul utilisateur u ;

— $atype : A \rightarrow At$, est une fonction mappant chaque objet à son type ;

— $aorg : A \rightarrow O$, est une fonction mappant chaque objet à son propriétaire organisation ;

— $assigned\_role\_orgs : U \rightarrow 2^{RO}$, est une fonction mappant un utilisateur à un ensemble des couples (rôle, organisation) qui lui sont attribués, $assigned\_role\_orgs(u) = \{(r, o)|(u, (r, o)) \in UA\}$;

— $active\_role\_orgs : S \rightarrow 2^{RO}$, est une fonction mappant une session si à un ensemble des couples (rôle, organisation) qui lui sont activés lors de cette session ; $active\_role\_orgs(si) \subseteq assigned\_role\_orgs(user(si))$;

— $can\_access(S, Op, A)$, est un prédicat défini comme suit : $can\_access(s, op, a)$=true si $\exists(r, o) \in active\_role\_orgs(s) \wedge aorg(a) = o \wedge ((op, atype(a)), r) \in PA$.

Le prédicat $can\_access(s, op, a)$ est vrai, signifie que l'utilisateur ayant activé la session s peut effectuer une opération sur l'objet a pendant la session. La définition de can_access dans ROBAC0 signifie qu'un utilisateur $user(s)$ membre de la session s peut effectuer une opération op sur un l'objet a si et seulement si : l'utilisateur est affecté au couple (r, o), où r est le rôle actif et o est l'organisation dans cette session ; le rôle r a le droit d'effectuer l'opération op sur le type dont l'objet a fait partie ; et l'objet a est détenu par l'organisation o.

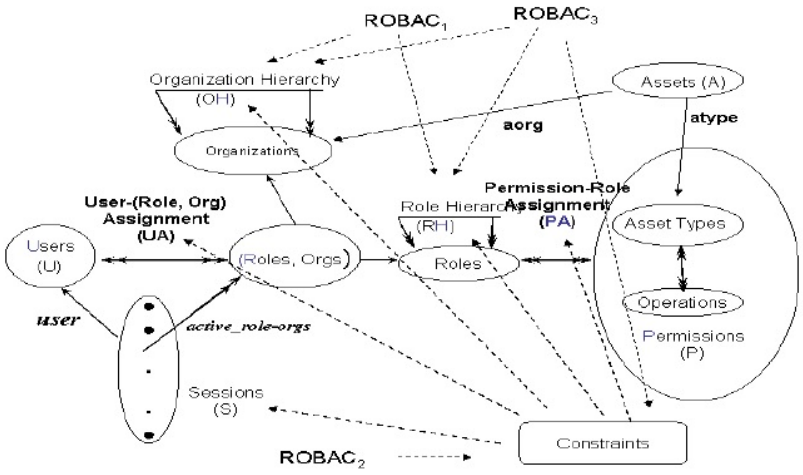

FIGURE 4.3: Modèle ROBAC [62]

Avantages et inconvénients

ROBAC est conçu spécifiquement pour les organisations ayant une structure homogène et hiérarchique. Ce modèle permet de faciliter l'administration des politiques d'accès et de réduire considérablement la complexité des décisions d'accès basées sur les rôles. Néanmoins, le modèle ROBAC ne prend pas en compte les exigences d'accès liés à la session collaborative. En outre, il ne permet pas de supporter les processus métiers collaboratifs s'exécutant sur un environnement multi-tenant, où un processus collaboratif se compose d'une suite de tâches, dans lequel chaque tâche est accomplie par un tenant donné.

De plus, dans ce modèle, l'administrateur de l'organisation centrale a le contrôle total sur l'affectation des utilisateurs aux rôles et l'attribution des permissions à ces rôles. Ceci est contradictoire avec le besoin de l'autonomie et l'indépendance (de la spécification de la politique d'accès) que souhaitent les tenants garder durant la collaboration. Enfin, l'organisation centrale doit avoir une connaissance totale sur la structure interne de chaque organisation à savoir : ses utilisateurs, ses objets, ses rôles définis et ses types d'objet. Ce qui contredit avec l'exigence de la confidentialité.

4.1.2 Approche de décomposition

Les auteurs ont proposé dans [32]) une architecture de contrôle d'accès basée sur XACML pour supporter le mécanisme de contrôle d'accès collaboratif. Ce mécanisme permet à plusieurs entités de collaborer pour prendre une décision d'accès d'un utilisateur qui est en mesure d'accéder à une ressource donnée. En utilisant cette architecture, la politique globale d'accès doit être définie et mise en œuvre par l'ensemble de parties collaboratrices sans compromettre les exigences d'autonomie et de confidentialité de ces parties.

L'idée principale de ce travail consiste à décomposer la politique d'accès à des sous politiques, et chaque sous politique sera intégrée dans la politique locale de l'entité concernée. L'implémentation de cette politique consiste alors à évaluer de façon parallèle les requêtes envoyées par les différentes entités, tout en garantissant la cohérence et la complétude des décisions de contrôle d'accès. Ce travail a introduit la notion de contrôle d'accès collaboratif.

Pour bien comprendre cette notion de décomposition, on prend l'exemple décrit précédemment, d'une application collaborative de télé-médecine. Supposant que l'on a une règle de contrôle d'accès qui dit "seul le neurologue du CHU,

membre d'une session collaborative cs a le droit de lire le dossier médical partagé dans cette session". La spécification de la politique de contrôle d'accès relative à cette règle est la suivante : $P = (B1 \wedge B2 \wedge B3 \wedge B4)$ tels que $B1$, $B2$, $B3$ et $B4$ sont des prédicats atomiques :

- $B1$: "$role(u) = neurologue$", il sera évalué au niveau de l'organisation CHU ;
- $B2$: "$\exists cs \in memberCS(u) \wedge SharedCS(o)$", il sera évalué au niveau du $SAMU$;
- $B3$: "$ObjType(o) = DM$", il sera évalué au niveau du HA ;
- $B4$: "$Action = Read$".

La politique P sera décomposée en trois sous politiques $P1$, $P2$ et $P3$ en utilisant l'algorithme de décomposition proposé dans ce travail. La politique $P = P1 \wedge P2 \wedge P3$, telles que les sous politiques :

- $P1 = B1 \wedge B4$, elle sera évaluée au niveau de l'organisation CHU.
- $P2 = B2 \wedge B4$, elle sera évaluée au niveau de l'entité $SAMU$.
- $P3 = B3 \wedge B4$, elle sera évaluée au niveau de l'entité HA.

Avantages et inconvénients

Cette approche permet de spécifier de façon homogène les différentes politiques de contrôle d'accès qui peuvent être appliquées durant le processus de collaboration. Le principal avantage de cette approche est de permettre à chaque organisation participante à la collaboration de garder la confidentialité de ses propres informations utilisées lors de décisions de contrôle d'accès. Cependant, la politique globale de contrôle d'accès est définie sur une entité centrale qui doit connaitre la structuration interne (objets, rôles, attributs, etc.) de chaque organisation. Ceci peut entrainer une divulgation des informations internes de chaque organisation.

De plus, l'entité centrale a le droit total de définir la politique globale de collaboration, ce qui contredit avec l'exigence d'autonomie et d'indépendance de chaque organisation. Enfin, la défaillance de cette entité centrale peut entrainer des risques liés à la disponibilité du système.

4.1.3 Modèle O2O (Organisation 2 Organisation)

O2O [15], [16] est une approche formelle pour assurer la collaboration entre les entités ayant leurs propres politiques d'accès dans un environnement multi-domaine. L'approche O2O se base sur le modèle OrBAC et repose sur deux concepts clés : organisation virtuelle privée (VPO) et le rôle Single-Sign-On (RSSO). Dans cette approche, chaque organisation définit de manière autonome sa politique locale qui gère aussi bien les accès internes à ses propres objets que les accès externes provenant des organisations participantes à la collaboration. La gestion des accès externes se concrétise par la création d'une sous-organisation, nommée VPO (Virtual Private Organization).

Cette notion de VPO est inspirée du concept d'organisation virtuelle (VO) [41]. Une Organisation Virtuelle est définie comme étant une communauté virtuelle regroupant plusieurs entreprises qui s'appuient sur leurs compétences humaines et ressources matérielles pour réaliser des tâches communes. Cette approche propose que la collaboration entre les organisations se fait à travers l'organisation virtuelle.

Une VPO permet à une organisation qui collabore avec d'autres organisations de garder le contrôle sur ses propres ressources partagées durant la collaboration. Pour bien comprendre le fonctionnement de l'approche, on considère qu'on a une organisation $SAMU$ qui veut communiquer avec l'organisation CHU (voir figure 4.4). Dans ce cas, chaque organisation ($SAMU$ et CHU) doit créer sa (VPO), respectivement appelé $SAMU2CHU$ et $CHU2SAMU$. Dans la VPO $SAMU2CHU$, l'administrateur $SAMU$ définit une politique d'accès selon laquelle les utilisateurs de $SAMU$ peuvent accéder aux ressources appartenant au CHU. De la même manière, dans la VPO $CHU2SAMU$, l'administrateur CHU définit une politique d'accès pour gérer les accès des utilisateurs de CHU aux ressources de $SAMU$. Les politiques de contrôle d'accès définies dans les VPO $SAMU2CHU$ et $CHU2SAMU$, sont de type OrBAC. Ainsi que les VPO $SAMU2CHU$ et $CHU2SAMU$ sont considérées comme des organisations du modèle OrBAC. Chaque organisation $SAMU$ ou CHU peut définir une "sphère d'autorité" dans laquelle les règles d'accès externes sont exprimées de façon formelle. A titre d'exemple, l'organisation $SAMU$ est dans la sphère d'autorité de l'organisation CHU si la politique associée aux accès d'utilisateurs de $SAMU$ aux ressources de CHU est définie et gérée par CHU.

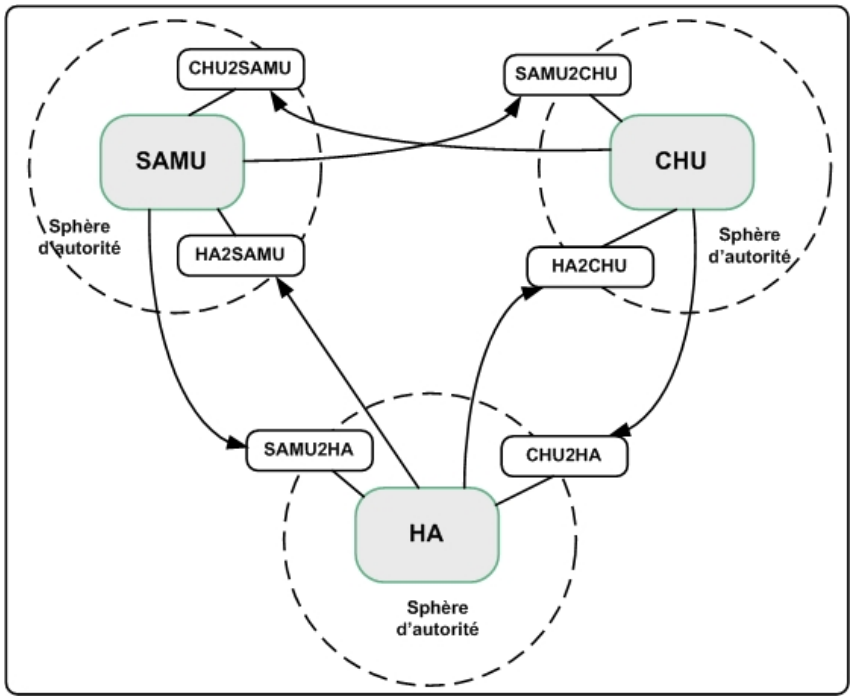

FIGURE 4.4: Architecture du modèle O2O

Avantages et inconvénients

Parmi les avantages du modèle O2O est que chaque organisation définit sa propre politique de contrôle d'accès de manière autonome, tout en assurant l'interopérabilité et la collaboration. De plus, l'adoption du principe RSSO permet à un utilisateur donné de jouer le même rôle R lorsqu'il accède aux ressources d'autres organisations. Chaque organisation virtuelle privée (VPO) peut attribuer un ensemble de droits d'accès à ce rôle R. Alors, même si ce rôle est le même, les permissions attribuées seront différentes dans chaque VPO.

Cependant, cette approche a quelques limitations discutées ci-après :

— Durant la collaboration, Pour qu'une organisation $SAMU$ définisse dans sa sphère d'autorité une politique de contrôle d'accès appliquée sur les utilisateurs de l'organisation CHU. L'organisation $SAMU$ doit avoir connaissance sur la structure interne de CHU. Ceci peut réduire la confidentialité du CHU ;

— Une organisation virtuelle privée VPO est crée et utilisée pour une durée temporelle durant laquelle une organisation accède à une ressource appartenant à une autre organisation. Une fois que cette organisation termine

son accès à cette ressource, la VPO sera détruite. En effet, la gestion des VPO peut devenir plus complexe et très lourde, surtout si nous avons des opérations répétitives.

4.1.4 Modèle Multi-OrBAC

Le modèle Multi-OrBAC a été introduit par [28], [7] comme une extension du modèle OrBAC pour assurer le contrôle d'accès dans un environnement collaboratif et inter-organisationnel. Multi-OrBAC est un modèle de contrôle d'accès dynamique et flexible permettant à chaque organisation participant à la collaboration, de définir ses propres politiques d'accès locales, et d'introduire d'une manière compatible des règles d'accès globales liées à la collaboration entre organisations dans la politique locale de chaque organisation. Ce modèle est bien adapté pour les applications collaboratives, distribués, hétérogènes et interopérables.

Un utilisateur ne joue pas forcément le même rôle dans toutes les organisations d'un environnement multi-organisationnel et un rôle n'a pas la même sémantique dans toutes les organisations. Pour cela, les notions de rôles, vues, activités dans le modèle Multi-OrBAC sont différentes aux celles du modèle OrBAC. Dans Multi-OrBAC, le concept de rôle dans l'organisation (RdO) est défini comme une extension du concept de rôle du modèle OrBAC.

De la même manière, une vue dans l'organisation (VdO) est définie comme une extension du concept vue du modèle OrBAC, donc on peut avoir plusieurs définitions possibles d'une même vue selon l'organisation. De la même manière, une activité dans l'organisation (AdO) est un ensemble d'actions ayant des propriétés communes et elle peut être définie différemment selon les organisations. Donc, on peut avoir plusieurs définitions possibles d'une même activité. De la même manière, la notion contexte dans l'organisation (CdO) a été définie par le modèle Multi-OrBAC. Le CdO pourrait être par exemple « urgence dans le CHU » ou « collaboration dans le SAMU ». Dans chaque organisation, l'administrateur de la politique de sécurité doit spécifier dans chaque nouvelle règle un ensemble de contraintes qui seront évaluées pour déterminer si un CdO est vrai ou pas à un instant donné. Une règle du modèle Multi-OrBAC est exprimée de la façon suivante : Une organisation org attribue au RdO r la permission (ou l'interdiction, ou l'obligation, ou la recommandation) de réaliser l'AdO a sur la VdO v, si

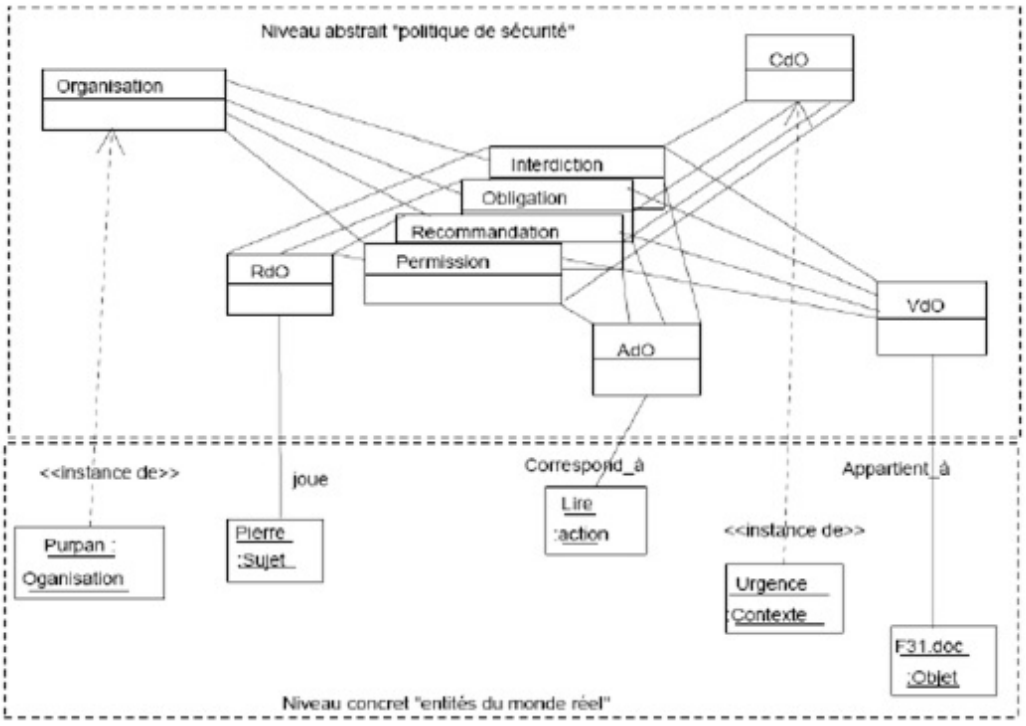

FIGURE 4.5: Exemple d'instanciation d'une règle de sécurité Multi-OrBAC [7]

le contexte CdO c est vrai.

Dans Multi-Orbac (voir 4.5) comme dans OrBAC, on distingue deux niveaux de permission : le niveau abstrait et le niveau concret. Dans le niveau abstrait, les permissions sont définies à l'aide des entités : organisations, RdO, VdO, AdO et CdO. De plus, les permissions de la politique d'une organisation peuvent être définies sur des entités faisant partie d'autres organisations. En faite, une permission abstraite de la politique de l'organisation $SAMU$ peut être spécifiée sur un RdO de l'organisation CHU et une VdO de l'organisation HA.

Avantages et inconvénients

Le modèle Multi-OrBAC se base sur le modèle OrBAC donc il bénéficie de tous les avantages d'OrBAC comme la dynamicité et la flexibilité. De plus, Multi-OrBAC est bien adapté pour supporter des systèmes multi-organisationnels, collaboratifs et interopérables. Le Multi-OrBAC introduit les concepts de rôle, vue, activité et contexte dans l'organisation, lui donne une flexibilité intéressante dans la définition des règles de contrôle d'accès pour les organisations collaborant dans un même système tout en supportant les problèmes d'incohérence entre les poli-

tiques de sécurité hétérogènes. Le modèle Multi-OrBAC offre un mécanisme pour détecter et résoudre le problème de conflits entre les politiques de sécurité locales et la politique de sécurité globale (de collaboration).

Le plus grand inconvénient du Multi-OrBAC est que chaque organisation n'est pas autonome dans la définition des règles de contrôle d'accès liée à la collaboration. Ainsi, par exemple, si l'organisation $SAMU$ définit des règles accordant des permissions aux utilisateurs de l'organisation CHU, alors CHU doit avoir confiance au $SAMU$ pour la gestion des utilisateurs, des rôles et des droits d'accès. Ce qui contredit avec les exigences d'autonomie et d'indépendance. De même, chaque organisation participant à la collaboration, doit garder la confidentialité de ses propres entités et de sa propre politique de sécurité. Ce qui n'est pas le cas en utilisant le modèle Multi-OrBAC.

4.1.5 Modèle PolyOrBAC

Abou El Kalam et al. a proposé le modèle PolyOrBAC [27] et [7] : une nouvelle extension du modèle OrBAC, pour modéliser et mettre en œuvre une collaboration sécurisée entre les organisations. Ce modèle PolyOrBAC se base sur deux concepts principaux : (1) Le modèle de contrôle d'accès OrBAC, utilisé pour définir la politique d'accès locale à chaque organisation. (2) La technologie des services Web qui permet d'assurer la collaboration et la communication entre les organisations, avec des extensions pour permettre de définir les règles de contrôle d'accès liées aux interactions.

En utilisant le modèle PolyOrBAC, il est donc primordial de spécifier des politiques de contrôles d'accès locales pour chacune des organisations qui participent à la collaboration. Dans le modèle PolyOrBAC, on peut distinguer deux types d'organisations : le client (bénéficiaire) qui bénéficie du service et le fournisseur du service qui offre ou met à disposition le service. Le client doit attribuer les droits d'accès à ses propres utilisateurs pour qu'ils puissent accéder aux services externes. Tandis que le prestataire doit contrôler et gérer les accès à ses propres services locaux par ses propres utilisateurs et par les utilisateurs externes appartenant à d'autres organisations.

Le modèle PolyOrBAC est assez flexible pour définir les politiques d'accès locales à chaque organisation, vu qu'il se base sur le modèle OrBAC. En outre, PolyOrBAC a introduit deux nouvelles notions à propos de l'utilisation des ser-

vices Web : (1) Image de service Web a été introduit pour que la cliente organisation puisse contrôler et gérer l'invocation d'un service Web externe dans sa propre politique d'accès locale. (2) Utilisateur virtuel a été introduit afin que le fournisseur de service puisse représenter une organisation (bénéficiaire du service) dans sa politique locale.

Pour pouvoir définir dans la politique locale de $SAMU$ le droit d'invoquer le service Web SW (voir figure 4.6), l'administrateur de sécurité de $SAMU$ doit créer un objet $Image\,de\,SW$, puis il définit une permission d'accès à cet objet pour l'action *invoquer*. Pour que l'utilisateur $user1$ appartenant à l'organisation CHU puisse invoquer $SW1$, il faut ajouter dans la politique d'accès du $SAMU$ une règle OrBAC qui attribue à un rôle joué par $User1$ le droit d'exécuter l'activité liée à l'action *invoquer* sur une vue à laquelle l'objet $Image\,de\,SW$ appartient. D'autre part, une fois l'action *invoquer* sur $Image\,de\,SW1$ s'est exécutée, un programme qui envoie le message SOAP d'appel du service Web $SW1$ à l'organisation CHU sera déclenché. Tandis que les contraintes exprimées dans le contrat signé entre le $SAMU$ et le CHU pour le service web SW, sont définies dans l'annuaire UDDI. De la même manière, afin que l'invocation de ce service web $SW1$ soit gérée

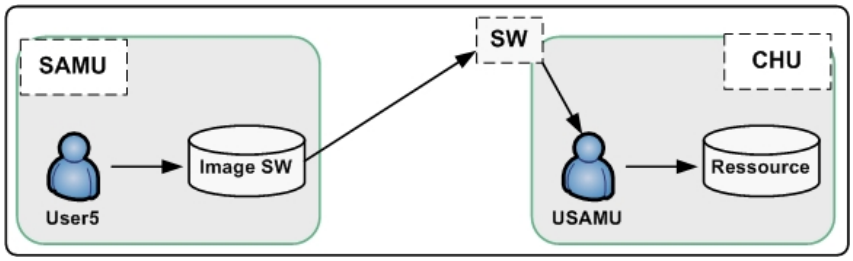

FIGURE 4.6: Utilisateur virtuel et image service web

dans l'organisation CHU, l'administrateur de sécurité de CHU doit créer un utilisateur virtuel USA jouant un rôle défini dans CHU, lui permettant d'accéder à la ressource liée à ce service Web $SW1$. Pour cela, il faut que l'administrateur du CHU insère une règle dans la politique d'accès locale de CHU qui attribue cette permission à ce rôle. Notez aussi, une fois l'invocation par l'organisation $SAMU$ au service Web $SW1$ est reçue, un programme s'exécute par l'utilisateur $USAMU$ pour réaliser toutes les actions nécessaires pour $SW1$.

Avantages et inconvénients

Le modèle PolyOrBAC est une solution intéressante pour assurer la collaboration entre les organisations tout en gardant l'autonomie et la confidentialité

de chacune. Cependant, ce modèle ne prend pas en compte la collaboration via les sessions collaboratives, ainsi que le partage des ressources entre les tenants durant une session collaborative. En raison de l'aspect dynamique de la session. les utilisateurs peuvent intervenir de façon dynamique sans aucune connaissance préalable de l'utilisateur qui va interagir sur telle ressource partagée dans la session. De plus, ce modèle ne supporte pas les workflows collaboratifs composés d'une suite de tâches, et chaque tâche sera accomplie par une organisation donnée.

4.2 Modèles de contrôle d'accès pour les environnements Multi-tenants

Les exigences du contrôle d'accès pour les systèmes multi-tenants ont certains points en communs avec celles du contrôle d'accès pour les environnements traditionnels, multi-organisationnels et collaboratifs. Dans ce qui suit, nous allons voir les différentes approches de contrôle d'accès liés aux environnements multi-tenants en détaillant à chaque fois, les différents avantages et limites de chaque approche.

4.2.1 Multi-tenants pour les services cloud

Dans [13], Calero et al ont proposé un système d'autorisation multi-tenant sous forme d'un middleware, fourni comme un service du cloud dans la couche PaaS. Ce système d'autorisation fournit un mécanisme de contrôle d'accès aux ressources et aux services appartenant aux différents tenants qui utilisent l'infrastructure de cloud. Dans ce travail, les auteurs considèrent que chaque organisation peut offrir plusieurs services cloud qui peuvent collaborer avec d'autres services appartenant à la même ou différentes organisations. L'objectif de ce système d'autorisation est de prendre en charge la collaboration et les interactions entre ces services. Dans ce qui suit, nous décrivons ce modèle d'autorisation.

Un modèle d'autorisation doit déterminer si un sujet a le privilège d'exécuter une action donnée sur l'objet contrôlé. Cela peut être représenté par le 3-triplet $(Sujet, Action, Objet)$. De plus, ce 3-triplet est étendu pour supporter la multi-tenancy dans laquelle différents tenants utilisent simultanément le même système

d'autorisation. Ainsi, un 4-uplet $(tenant, sujet, action, objet)$ est défini. De plus, le modèle d'autorisation est étendu pour protéger différents types d'objets : le 4-uplet précédent peut être étendu dans le 5-tuple
$(tenant, Sujet, Privilge, Interface, Objet)$, où Interface représente la façon dont l'objet peut être interprété. Par exemple, $(Jose, Nigel, Read, CloudStorage, root)$ peut être interprété comme : $Jose$ dit que $Nigel$ peut lire le dossier $root$ associé au service $CloudStorage$. Notez qu'un privilège peut activer plusieurs actions. Par exemple, l'action écrire pourrait activer les actions lire et modifier.

Ce système d'autorisation est basé la notion du rôle défini dans le modèle RBAC et un rôle est associé à un ensemble des privilèges. Ainsi, l'association d'un utilisateur à un rôle donné peut être définie par un triplet $(Tenant, User, R1)$ qui peut être interprété comme suit : le tenant déclare que l'utilisateur $User$ joue le rôle $R1$. Le 5-uplet précédent est étendu pour définir les permissions associées aux rôles comme suit : $(metteur, [utilisateur|role], privilege, interface, objet)$. En outre, ce modèle supporte la hiérarchie des rôles (c'est-à-dire un rôle peut hériter les privilèges accordés à un autre rôle) et la hiérarchie d'objets basée sur son chemin.

Cependant, la relation de confiance introduite dans ce travail, est avec une granularité grossière. De plus, les politiques d'autorisations et les relations de confiance sont stockées dans une base de connaissances centralisée. Les décisions d'autorisation sont également prises dans un point de décision de la politique (PDP) centralisé. Enfin, le modèle de confiance et le modèle d'autorisation sont décrits d'une manière informelle.

4.2.2 Modèle CTTM

Les auteurs de [54] ont proposé un modèle de confiance entre les tenants (CTTM : Cross-Tenant Trust Models) qui englobe les différents types de relations de confiance qui peuvent être établies entre deux tenants. Une relation de confiance entre tenants devrait être réflexive, mais pas transitive, symétrique ou anti-symétrique. Dans ce travail, les auteurs ont identifiés quatre types de relation de confiance entre les tenants. En se basant sur ces relations de confiance entre les tenants, les auteurs ont formalisé le modèle CTTM et ont présenté une extension du CTTM à base du rôle (RB-CTTM).

Le modèle de confiance (CTTM) propose différents types de relations de

confiance unilatérales qui répondent aux différents besoins de contrôle d'accès entre deux tenants, le fournisseur de confiance (truster) et le bénéficiaire de confiance (trustee).

Ce travail a présenté une étude sur les relations de confiance établies entre les tenants $(TT) \subseteq T \times T$ qui est la partie principale des modèles de contrôles d'accès pour les environnements multi-tenants. TT (également écrit comme $\rightarrow$) est une relation binaire du truster au trustee. On considère que T est l'ensemble de tous les tenants. Pour $\forall A\,;\, B\,;\, C \in T$, la relation TT est :

- réflexive : $(A \rightarrow A)$;
- pas transitif : $(A \rightarrow B \wedge B \rightarrow C$ n'implique pas que $A \rightarrow C)$;
- elle n'est ni symétrique : $(A \rightarrow B$ n'implique pas que $B \rightarrow A)$;
- ni antisymétrique : $(A \rightarrow B \wedge B \rightarrow A$, n'implique pas que A et B sont le même tenant).

En ce qui concerne l'utilisation de la relation TT, les auteurs ont identifié quatre types potentiels de relations de confiance pour assurer et contrôler les accès inter-tenants. Par la suite, on utilise l'exemple des tenants des relations de confiance qui peuvent être établies entre le $SAMU$ et le CHU :

- $Type-a(SAMU \rightarrow a\,CHU)$: le truster peut donner l'accès au trustee. En utilisant cette relation, le truster $SAMU$ peut affecter les utilisateurs du CHU à ses propres permissions. Dans ce cas, le truster détient le contrôle total dans l'affectation des permissions ;
- $Type-b(CHU \rightarrow b\,SAMU)$: le trustee peut donner l'accès au truster. En utilisant ce type de relation, le truster CHU expose quelques informations d'utilisateurs nécessaires pour que le $SAMU$ puisse affecter les utilisateurs du CHU à ses propres permissions. Dans ce cas, ces accès sont contrôlés par le $SAMU$ et le CHU ;
- $Type-c(SAMU \rightarrow c\,CHU)$: le trustee peut prendre l'accès du truster. En utilisant ce type de relation, le truster $SAMU$ délègue au trustee CHU pour attribuer ses propres permissions aux utilisateurs du CHU. De même, l'accès inter-tenants est contrôlé par le $SAMU$ et le CHU ;
- $Type-d(CHU \rightarrow d\,SAMU)$: le truster peut prendre l'accès du trustee. Ce type de relation de confiance n'est pas utile dans la réalité. En utilisant ce type, le CHU fait confiance au $SAMU$ et affecte les permissions du $SAMU$

à ses propres utilisateurs. Dans ce cas, le CHU possède tout seul le contrôle total aux accès inter-tenants.

Dans la formalisation de CTTM, schématisée dans (figure 4.7), le modèle comporte trois entités : les utilisateurs (U), les permissions (P) et les tenants (T). L'entité tenant a été introduite pour spécifier les règles d'accès relatives aux systèmes multi-tenant. Dans ce modèle, un utilisateur dans U et une permission dans P sont détenues respectivement par un tenant dans T afin qu'ils puissent être identifiés de manière unique dans un environnement multi-tenant.

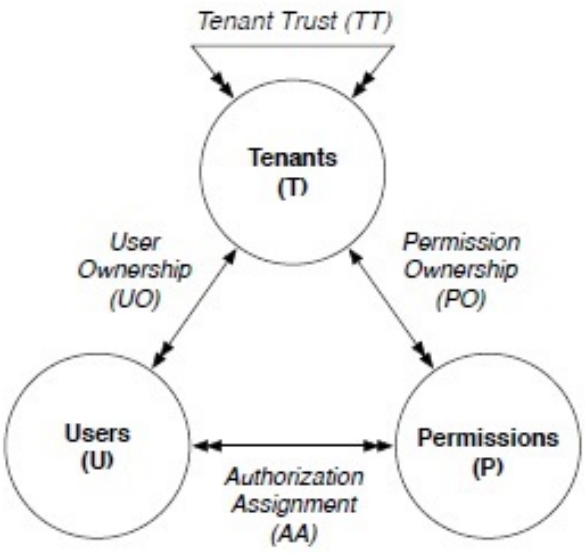

FIGURE 4.7: Modèle CTTM [54]

— $(AA) \subseteq U \times P$, est une relation plusieurs-à-plusieurs mappant chaque utilisateur à un ensemble des permissions qui lui sont attribuées telles que $(u, p) \in AA$ seulement si
$userOwner(u) = permissionOwner(p) \vee$
$permissionOwner(u) \rightarrow auserOwner(p) \vee$
$userOwner(p) \rightarrow bpermissionOwner(u) \vee$
$permissionOwner(u) \rightarrow cuserOwner(p)$

Notez que les fonctions $UOwner(u)$ et $POwner(p)$ retournent respectivement le tenant propriétaire de l'utilisateur u et celui de la permission p.

Avantages et inconvénients

Ce travail présente les différents types de relations de confiance unilatérales qui répondent aux différents besoins de contrôle d'accès entre deux tenants, le truster

(celui qui établit la confiance) et le trustee (celui qui reçoit la confiance). Ceci permet ainsi d'assurer le contrôle d'accès dans les environnements multi-tenants. Cependant, les relations de confiance identifiées dans ce travail sont très statiques. En faite, si un tenant (truster) fait confiance à un autre tenant (trustee) pour que ce dernier puisse avoir quelques permissions du truster. Alors, en utilisant ce modèle, l'ensemble des permissions définies par le truster, sont attribuées au trustee. Ceci est contradictoire avec les exigences de contrôle d'accès relatives à la collaboration.

D'autre part, ces relations de confiance ne supportent pas les règles d'accès dynamiques exprimant des situations contextuelles qui peuvent entrainer lors des sessions collaboratives. De plus, en utilisant ces relations de confiance, les tenants en questions partagent entre eux des ressources sans garder la confidentialité de la politique d'accès et des informations d'utilisateurs. Pour cela, il est nécessaire d'avoir un modèle qui supporte toutes les exigences de collaboration entre les tenants.

4.2.3 Système d'autorisations multi-tenant (MTAS)

Bo et al [55], formalise l'approche MTAS proposée par Calero et al [13], en développant une extension du MTAS, pour assurer un contrôle d'accès plus fin dans les systèmes inter-tenants. Le modèle proposé est basé sur RBAC, en utilisant la relation de confiance de type b défini dans [54] pour supporter l'accès inter-tenant. Le modèle MTAC (figure 4.8), est composé de cinq entités : émetteurs (issuers) (I), utilisateurs (users) (U), permissions (P), rôles (R) et sessions (S). Dans ce modèle, les entités : utilisateurs, permissions et rôles sont associées à leur émetteur (Issuer) propriétaire. Les trois relations (UO), (RO) et (PO) sont des relations un-à-plusieurs entre les utilisateurs, les rôles et les permissions respectivement et leur issuer propriétaire. La relation de confiance entre les issuers (introduite dans ce travail) $(IT) \subseteq I \times T$ est une relation réflexive plusieurs-à-plusieurs entre le Truster Tr et le Trustee. On note cette relation par le symbole "$\rightarrow$". On considère que I est l'ensemble de tous les issuers. $\forall\ i1,\ i2,\ i3 \in I$, la relation IT est :

— réflexive : $(i1 \rightarrow i1)$;

— pas transitif : $((i1 \rightarrow i2) \wedge (i2 \rightarrow i3)$ n'implique pas que $(i1 \rightarrow i3))$;

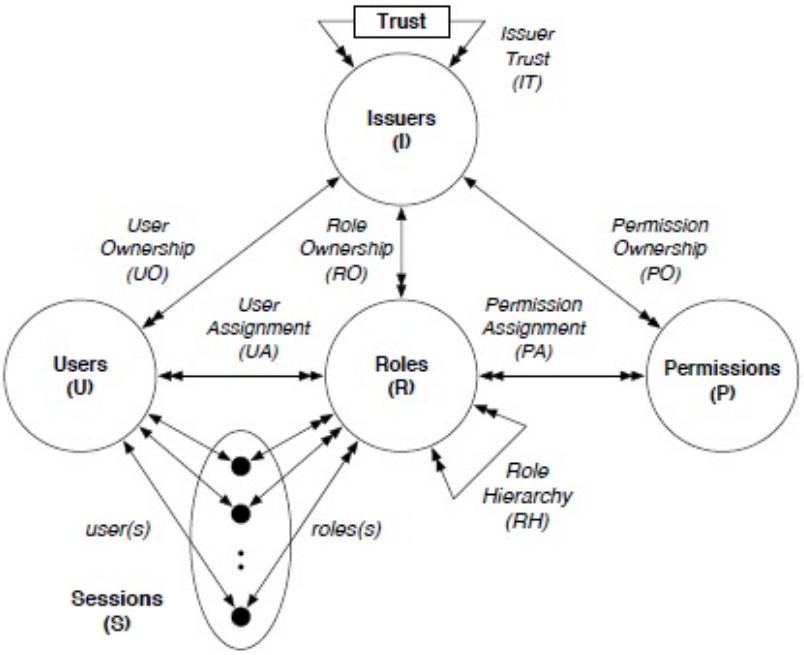

FIGURE 4.8: Modèle MTAS [55]

— et elle n'est ni symétrique : $((i1 \rightarrow i2)$ n'implique pas que $(i2 \rightarrow i1))$;

— ni antisymétrique : $((i1 \rightarrow i2) \wedge (i2 \rightarrow i1)$, n'implique pas que $i1$ et $i2$ sont le même issuer).

On considère que A et B sont deux issuers. A établit une relation de confiance d'issuer (IT) avec B $(A \rightarrow B)$. Alors, A expose tous ses rôles de base ainsi que les rôles hérités à B, de telle sorte que B puisse effectuer les deux opérations suivantes :

1) Affecter les permissions de B aux rôles de A ; ou

2) Affecter les rôles de B en tant que des rôles juniors de ceux de A.

Par exemple si on considère qu'on a deux issuers le $SAMU$ et le CHU. Ce dernier veut attribuer au rôle $docteur\_samu$ du $SAMU$ la permission "lire le dossier médical 1 : $(read, DM1)$". Pour assurer l'accès inter-tenant, le $SAMU$ doit faire confiance au CHU $(SAMU \rightarrow CHU)$. Dans ce cas, le CHU affecte la permission "$(read, DM1)$" du CHU au rôle $docteur\_samu$ du $SAMU$.

Avantages et inconvénients

Le modèle MTAS permet d'assurer le contrôle d'accès aux ressources échangées entre les tenants. Cependant, vu l'aspect dynamique de la session collaborative, ce modèle ne supporte pas le partage des ressources entre les tenants lors

d'une session collaborative. De plus, ce modèle n'assure pas la collaboration définie sous forme d'un workflow composé d'une suite de tâches, telle que chaque tâche est accomplie par un tenant. Enfin, dans ce modèle, le truster doit divulguer les informations de ses propres rôles au trustee, ce qui contredit avec le besoin de confidentialité que souhaitent les tenants garder durant la collaboration.

4.2.4 Modèle MT-RBAC

Tang et al ont proposé dans [53], le modèle de contrôle d'accès MT-RBAC (Multi-Tenants Role Based Access Control). Il est basé sur le modèle RBAC pour assurer une collaboration sécurisée dans un environnement multi-tenant. Ce modèle a introduit une relation de confiance entre les tenants pour supporter l'accès inter-tenants. Le modèle MT-RBAC (figure 4.9), est composé de six entités : émetteurs (issuers) (I), tenants (T), utilisateurs (users) (U), permissions (P), rôles (R) et sessions (S). Dans ce modèle, les entités traditionnelles du modèle

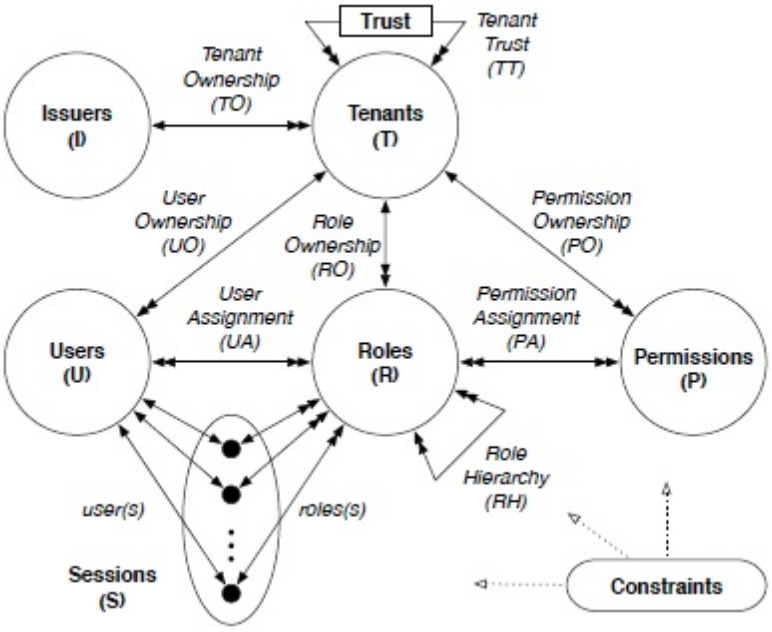

FIGURE 4.9: Modèle MT-RBAC [53]

RBAC : utilisateurs, permissions et rôles sont associées à leur tenant propriétaire. Les trois relations (UO), (RO) et (PO) sont des relations un-à-plusieurs respectivement entre les utilisateurs, les rôles et les permissions et leur tenant pro-

priétaire. De plus, il existe une autre relation entre les tenants et leur émetteur (Issuer) propriétaire.

Cette relation de confiance entre les tenants $(TT) \subseteq T \times T$ est une relation réflexive plusieurs-à-plusieurs entre le Truster Tr et le Trustee Te : $Trust(tr, te : T) \rightarrow 2^R$. Cette relation signifie que le tenant Tr autorise Te à utiliser certains rôles définis par Tr pour que le tenant Te puisse affecter ces utilisateurs à ces rôles de Tr. Par exemple : $Trust(SAMU, CHU) = \{neurologue, radiologue\}$, signifie que le tenant $SAMU$ expose les deux rôles neurologue et radiologue pour qu'ils soient utilisés par le tenant CHU.

Les auteurs ont défini la fonction $canUse : R \rightarrow 2^T$, pour déterminer l'ensemble des tenants qui peuvent utiliser un rôle donné. Cette fonction prend en paramètre un rôle $(r \in R)$ et retourne un ensemble de tenants 2^T, telle que :

— $canUse(r) = \{t\} \cup \{tx \in T, r \in Trust(tx, t)\}$, t est le tenant propriétaire du rôle r.

Avantages et inconvénients

Ce modèle MT-RBAC permet d'assurer un partage sécurisé des ressources entre les tenants. La relation de confiance utilisée dans ce modèle est de type C (voir la sous section 4.2.2). Cependant, ce modèle ne supporte pas le partage des ressources entre les tenants durant la session collaborative vu l'aspect dynamique de la session. En faite, les utilisateurs peuvent intervenir de façon dynamique sans aucune connaissance préalable de l'utilisateur qui va interagir dans la session. De plus, ce modèle n'assure pas une collaboration définie sous forme d'un workflow composé d'une suite de tâches, et chaque tâche est accomplie par un tenant.

4.2.5 Modèle MT-ABAC

Ce travail [45] présente un nouveau modèle (MT-ABAC : Multi-Tenant Attribute Based Access Control) de contrôle d'accès basé sur les attributs pour assurer la collaboration entre tenants dans un IaaS cloud. Ainsi que, cette approche permet l'affectation d'attributs entre tenants pour assurer un accès sécurisé aux ressources partagées durant la collaboration. Plus précisément, la relation de confiance établie entre les tenants permet au trustee d'affecter ses propres attributs aux utilisateurs provenant du tenant truster. Ceci permet ainsi à ces utilisateurs d'accéder aux ressources appartenant au trustee.

Ce modèle MT-ABAC hérite les points forts du modèle ABAC, tels que la

flexibilité. Dans le modèle ABAC, l'accès d'un utilisateur à une ressource dépend des valeurs d'attributs de cet utilisateur et de cette ressource. Dans ce modèle (voir figure 4.10), les auteurs ont étendu le modèle basique ABAC0 en introduisant l'entité tenant (T) aux autres entités utilisateurs et objets. Ceci est pour que le modèle ABAC puisse supporter la notion de multi-tenant. Dans ce modèle,

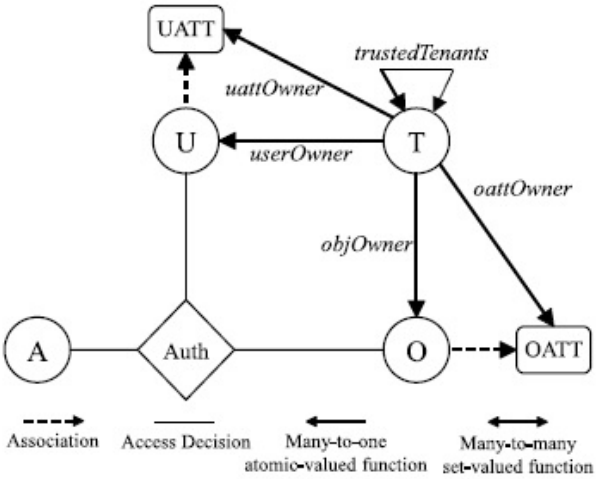

FIGURE 4.10: Modèle MT-ABAC [45]

chaque utilisateur et chaque objet est détenu uniquement par un seul tenant. Pour cette raison, le modèle nécessite que chaque utilisateur possède un attribut utilisateur défini par le système, *userOwner*, qui est une fonction atomique plusieurs-à-un. Cette fonction prend en paramètre l'utilisateur et retourne son propriétaire tenant. De même, il est nécessaire que chaque objet ait un attribut système défini *objOwner* qui, de même, est une fonction atomique plusieurs-à-un, prenant en paramètre des objets O et retourne leur propriétaire tenant T.

De plus, chaque attribut d'utilisateur et chaque attribut d'objet appartient également à un tenant unique. Le concept crucial est que chaque tenant est responsable de l'attribution des valeurs aux attributs qu'il possède. En utilisant le concept d'isolation des tenants, un tenant ne peut attribuer des valeurs d'attribut que pour ses propres attributs et uniquement à ses propres entités (utilisateurs et objets). De même, dans ce modèle, pour supporter l'accès inter-tenant, les auteurs ont introduit une nouvelle relation de confiance.

Cette relation de confiance entre les tenants $(TT) \subseteq T \times T$ est définie comme étant une fonction réflexive plusieurs-à-plusieurs entre le fournisseur de confiance

(Truster) Tr et le bénéficiaire de confiance (Trustee). Nous utilisons le symbole "$\rightarrow$" pour représenter la relation de confiance entre deux tenants TA et TB où, $TA \rightarrow TB$ signifie que $TB \in trustedTenants(TA)$. En utilisant cette relation, TA approuve qu'il fasse confiance à TB.

$TA \rightarrow TB$, cela signifie que le tenant TB est autorisé à attribuer des valeurs pour les attributs d'utilisateur de TB aux utilisateurs du tenant TA. Le tenant TA contrôle l'établissement de la confiance avec TB tandis que TB contrôle l'attribution des valeurs d'attribut entre les tenants. Plus précisément, en utilisant cette définition, on ne peut pas attribuer à un utilisateur donné u une valeur pour l'attribut $uatt$ que si :
$uattOwner(uatt) = userOwner(u) \cup$
$uattOwner(uatt) \in trustedTenants(userOwner(u))$.

Telle que la fonction $uattOwner(uatt)$, représente le tenant propriétaire de l'attribut u. Enfin, dans chaque prédicat d'autorisation $authorization_a(u, o)$, le modèle doit vérifier la compatibilité entre les attributs d'utilisateur et ceux d'objet. Pour cette raison, tout attribut utilisateur $uatt$ ou attribut objet $oatt$ utilisé dans le prédicat d'autorisation $authorization_a(u, o)$, pour une action donnée, doit satisfaire la condition suivante : $uattOwner(uatt(u)) = oattOwner(oatt(o)) \vee$
$oattOwner(oatt(o)) \in trustedTenants(uattOwner(uatt(u)))$

Avantages et inconvénients

Ce travail présente une extension du modèle ABAC pour contrôler l'accès aux ressources partagées dans un environnement multi-tenant. Le modèle étendu MT-ABAC se base sur la relation de confiance de type b, où la collaboration est assurée à travers l'affectation inter-tenants des valeurs d'attributs. MT-ABAC est un modèle plus fin, dynamique et flexible, il permet de supporter n'importe quelle situation contextuelle, ainsi que les sessions collaboratives.

Néanmoins, le modèle MT-ABAC n'assure que le partage des ressources entre les tenants. En faite, il ne permet pas de supporter les processus métiers collaboratifs s'exécutant sur un système multi-tenant. Où un processus se compose d'une suite de tâches, telle que chaque tâche est accomplie par un collaborateur (tenant). De plus, dans ce modèle, le truster doit divulguer les informations de ses propres utilisateurs à savoir (la liste des utilisateurs, leur fonction..) au trustee, afin que ce dernier puisse leur attribuer les valeurs de ses propres attributs. Dans ce sens, l'exigence de "garder la confidentialité" n'est pas respectée. Enfin,

le trustee définit tout seul les autorisations inter-tenants liées aux utilisateurs du truster sans la participation de ce dernier. Ceci est contradictoire avec le besoin de l'autonomie et l'indépendance (dans la définition des politiques de sécurité) que souhaitent les tenants avoir durant la collaboration.

4.3 Étude comparative

Dans cette section, nous allons faire une étude comparative entre les différentes approches de contrôle d'accès (celles pour la collaboration et celles pour les environnements multi-tenants) décrites dans ce chapitre (Table 6.1). Cette étude s'appuie sur un certaine nombre d'exigences et besoins de contrôle d'accès pour un système collaboratif à base de cloud. L'objectif de cette étude est de déterminer les exigences que supporte chaque approche. Un système collaboratif à base de cloud doit prendre en compte certaines exigences et besoins que l'on va détailler dans ce qui suit :

— Multi-tenant/Multi-domaine : Chaque tenant (ou domaine) peut partager un ou plusieurs ressources avec d'autres tenants (domaines) dans le but d'atteindre un objectif commun ;

— Agilité du tenant : Un tenant dans le cloud peut être créé pour un usage temporaire et supprimé par la suite. Les modèles de contrôle d'accès pour les environnements multi-tenants devraient donc être suffisamment agiles et souples pour faire face à ce type d'utilisation.

— Agilité de la session collaborative : Une session collaborative est créée et se déroule pour une durée temporaire et sera clôturée et détruite dès la fin du processus de collaboration. De plus, une session collaborative peut être désactivée et réactivée à tout moment par le gestionnaire de la session. En outre, la connexion / la déconnexion des utilisateurs à/de la session collaborative se fait d'une manière dynamique. Le modèle de contrôle d'accès devrait donc être suffisamment agile pour répondre aux exigeantes relatives à la session collaborative.

— Partage des ressources dans la session entre plusieurs tenants : Durant une session collaborative, un utilisateur appartenant à tenant peut partager une ressource avec d'autres utilisateurs (appartenant aux différents tenants)

pour un usage temporaire. Une fois la ressource est partagée, elle sera soumise aux règles de contrôle d'accès de la session collaborative. Le modèle de contrôle d'accès devrait prendre en compte une telle situation.

— Processus collaboratifs (Tâches et workflow) : Chaque processus collaboratif est composé d'une suite des tâches et dans lequel chaque tâche est accomplie par un tenant participant à la collaboration. Les modèles de contrôle d'accès devraient donc supporter les processus collaboratifs s'exécutant dans un environnement multi-tenant.

— Flexibilité : le modèle de contrôle d'accès doit être flexible autant que possible pour définir des politiques d'accès adaptées à n'importe quel scénario ;

— Multi Cloud : le modèle de contrôle d'accès devrait prendre en considération les sessions collaboratives se déroulant dans un environnement multi-cloud [44]. C'est-à-dire, les tenants participants à la collaboration, appartenant aux différents fournisseurs de cloud ;

— Autonomie et indépendance : chaque administrateur local d'un tenant doit avoir l'autorité totale sur ses propres utilisateurs et ressources. Chaque tenant définit sa politique de contrôle d'accès locale d'une manière autonome, tout en respectant les règles de la politique globale ;

— Confidentialité : Chaque tenant doit garder la confidentialité de ses données, de ses attributs, de ses valeurs d'attributs et de sa politique de contrôle d'accès locale vis-à-vis les autres tenants participant à la collaboration.

— Hétérogénéité : Le modèle de contrôle d'accès doit prendre en compte la situation où les tenants participant à la collaboration utilisent des modèles de contrôle d'accès hétérogènes [9].

L'étude faite dans cette section, montre que les travaux existants ne traitent pas le concept de la session collaborative ainsi que le partage des ressources dans la session entre plusieurs tenants. De plus, ces travaux ne prend pas en compte de la notion de tâche et workflow, multi cloud et l'hétérogénéité. Pour cela, notre contribution dans cette thèse consiste à l'extension des modèles de contrôle d'accès les plus répandus à savoir : $RBAC$, $OrBAC$, $TRBAC$ et $ABAC$.

Ces modèles étendus s'appuient sur un langage formel basé sur la logique du premier ordre. Ils permettent de (1) supporter l'aspect dynamique de la session collaborative se déroulant dans un environnement multi-tenant ; (2) supporter

les processus collaboratifs s'exécutant sur un système multi-tenant ; (3) garder l'autonomie et la confidentialité des tenants durant la collaboration.

Critères	***ROBAC***	***Décomposition de la politique***	***Multi-OrBAC et O2O***	***Poly-OrBAC***	***MTAS, CTTM et MT-RBAC***	***MT-ABAC***
Multi-tenant	*OUI*	*OUI*	*OUI*	*OUI*	*OUI*	*OUI*
Agilité du tenant	*NON*	*NON*	*NON*	*NON*	*OUI*	*OUI*
Agilité de la session collaborative	*NON*	*NON*	*NON*	*NON*	*NON*	*NON*
Partage des ressources dans la session	*NON*	*NON*	*NON*	*NON*	*NON*	*NON*
Processus collaboratifs	*NON*	*NON*	*NON*	*NON*	*NON*	*NON*
Multi Cloud	*NON*	*NON*	*NON*	*NON*	*NON*	*NON*
Flexibilité	*NON*	*NON*	*OUI*	*OUI*	*NON*	*OUI*
Autonomie et indépendance	*NON*	*NON*	*OUI*	*OUI*	*NON*	*NON*
Confidentialité	*NON*	*OUI*	*NON*	*OUI*	*NON*	*OUI*
Hétérogénéité	*NON*	*NON*	*NON*	*NON*	*NON*	*NON*

TABLE 4.1: Étude comparative des approches de contrôle d'accès pour les environnements multi-organisationnels et multi-tenants

4.4 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art des modèles de contrôle d'accès pour les environnements Multi-organisationnels : PolyOrBAC, Multi-OrBAC, la décomposition de la politique et O2O. Puis, nous avons montré les avantages et les limites pour chaque modèle. Ensuite, nous avons vu les différentes approches de contrôle d'accès liés aux environnements multi-tenants, en détaillant à chaque fois, les différents avantages et limites de chaque approche. Enfin, nous avons fait une étude comparative de différentes approches de contrôle d'accès décrites dans ce chapitre. Dans le chapitre suivant, nous présentons une nouvelle approche que nous avons appelée MTCS-OrBAC [35] (Multi-Tenant Collaborative Session for OrBAC), pour assurer le contrôle d'accès aux ressources partagées lors d'une session collaborative.

Chapitre 5

Modèle session collaborative multi-tenant pour OrBAC : MTCS-OrBAC

Dans ce chapitre, nous présentons une nouvelle approche que nous avons appelée le modèle MTCS-OrBAC (Multi-Tenant Collaborative Session for OrBAC Model), pour assurer le contrôle d'accès des ressources échangées lors d'une session collaborative se déroulant sur un environnement multi-tenants. MTCS-OrBAC est basée sur le modèle de contrôle d'accès OrBAC : (A Organization Based Access control Model) en introduisant de nouvelles entités comme le tenant, la session collaborative et le type de la session collaborative. De plus, nous proposons de nouvelles relations de confiance pour supporter l'accès inter-tenant. Le principal objectif du modèle MTCS-OrBAC est d'étendre le modèle OrBAC pour qu'il prenne en compte le concept de la "multi-tenancy" et l'aspect dynamique de la session collaborative. Cette approche a été implémentée sur swift : la composante de stockage du Cloud plateforme Openstack.

Le modèle MTCS-OrBAC [35] est une extension du modèle OrBAC en prenant en compte les exigences suivantes : (1) la multi-tenancy et l'échange des ressources entre tenants dans un environnement Cloud, c'est-à-dire un tenant accède à des ressources détenues par autres tenants ; (2) l'aspect dynamique de la session collaborative : d'une part, le changement des utilisateurs (Invitation, connexion et la déconnexion) durant la session. D'autre part, le partage/ le départage des ressources dans/de la session collaborative. De plus, durant une session

collaborative, nous considérons que seuls ses membres peuvent accéder aux ressources partagées dans cette session selon les privilèges qui leur sont attribuées.

Le modèle OrBAC a été choisi dans ce travail en tant que modèle de base pour sa capacité d'abstraction permettant de définir de politiques de contrôle d'accès flexibles et contextuelles. Dans ce chapitre, tout d'abord, nous commençons par étendre le modèle OrBAC pour qu'il supporte la notion "Session collaborative" dans un environnement d'un seul tenant (Single tenant). Par la suite, nous présentons le modèle étendu MTCS-OrBAC en considérant un environnement multi-tenant, c'est-à-dire les utilisateurs membres de la session et les ressources partagées dans cette session appartiennent aux différents tenants.

5.1 Modèle MTCS-OrBAC pour un seul tenant

Dans cette section, nous considérons des sessions collaboratives se déroulant dans un environnement d'un seul tenant. C'est-à-dire, les utilisateurs membres de la session et les ressources partagées dans cette session appartiennent à un seul tenant. Dans notre modèle MTCS-OrBAC, nous étendions le modèle de base OrBAC [26] comportant déjà les entités organisation, rôle, activité, vue, etc, par l'ajout des entités « Session collaborative (CS) » et « Type de la session collaborative (TCS) ». Ceci est dans de le but de prendre en considération la notion de session ainsi que la session collaborative.

Une session est une instance d'activité établie par un seul utilisateur. Elle correspond à un utilisateur unique. On la nomme une session individuelle pour la différencier de la session collaborative. Cette dernière correspond à une instance d'activité de collaboration. Elle peut comporter un ou plusieurs utilisateurs. La session collaborative est l'entité de base dans notre modèle MTCS-OrBAC. Elle comprend un ensemble d'utilisateurs (appelés membres de la session), effectuant des tâches spécifiques, en accédant à des ressources partagées pour atteindre un objectif commun.

Chaque session collaborative est instanciée à partir d'une activité de collaboration, nommée "Type de la session collaborative". Une organisation peut définir plusieurs types de session collaborative. Session publique, session privée, urgence dans la neurologie, etc) sont des exemples de types de session. Ce modèle introduit la notion "rôle dans la session", c'est à dire le rôle qu'un utilisateur a activé

durant la session collaborative. A cet égard, ce modèle fait la différence entre les rôles affectés et les rôles activés. Les rôles affectés sont les rôles attribués à un utilisateur par l'administrateur de sécurité. Cet utilisateur peut activer un ou plusieurs rôles attribués lors de la session selon les contraintes d'activation d'un rôle. De même, le modèle introduit la notion "vue dans la session", c'est à dire, les objets appartenant à une vue donnée et qui sont partagés dans la session collaborative.

Rappelons que le modèle OrBAC comporte deux niveaux de permissions : les permissions abstraites et permissions concrètes. Dans le niveau abstrait, une permission est définie par les entités organisation, rôle, vue, activité et contexte. Tandis que dans le niveau concret, une permission est définie par les entités utilisateur, action et objet. De même, dans notre modèle MTCS-OrBAC, une permission abstraite est définie par le 6-uplet des entités (tenant, type de la session, rôle, activité, vue et contexte). Dans le niveau concret, une permission est définie par le 4-uplet (session collaborative, utilisateur, action, objet).

Dans ce modèle, comme il est schématisé dans la figure 5.1, nous avons plusieurs entités [35] Tenants (T), sessions collaboratives (CS), types de la session collaborative (TCS), rôles (R), activités (A), vues (V), contextes (C), utilisateurs (U), actions (AC) et les objets (O). Les différentes entités du modèle à savoir (les rôles, les activités, les vues, les types de session collaborative, les sessions collaboratives, les utilisateurs et les objets) sont associées par la relation un-ou-plusieurs à leur tenant propriétaire. Dans ce qui suit on va décrire en détail chaque entité du modèle.

5.1.1 Tenants

Un tenant est une partition virtuelle d'un service cloud ou d'une instance d'application logicielle fournie par le fournisseur des services cloud au client. Le tenant dans notre modèle étendu correspond à l'entité organisation définie dans OrBAC. Un tenant peut avoir plusieurs utilisateurs et objets. De plus, un tenant peut définir ses propres entités type de la session, rôle, activité, vue et contexte. L'administrateur du tenant affecte les utilisateurs, les actions et objets aux rôles, activités et vues respectivement. Cet administrateur définit la politique de contrôle d'accès locale en se basant sur ces entités. On note T est l'ensemble des tenants possibles.

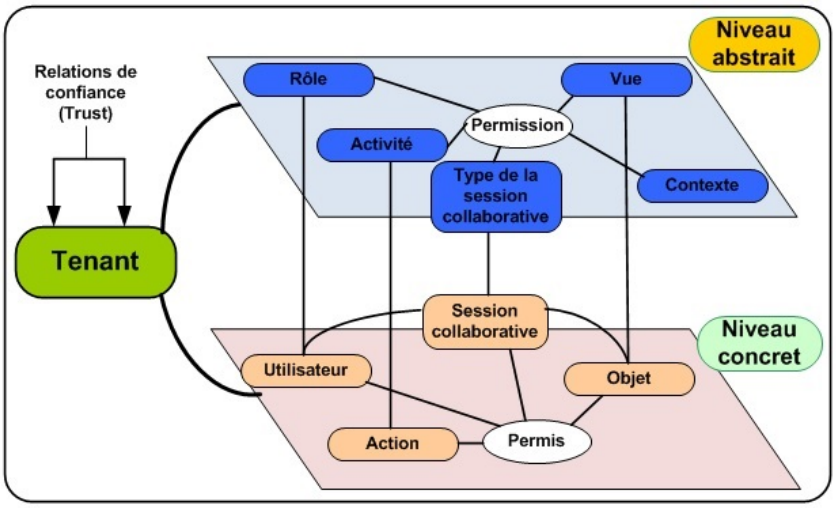

FIGURE 5.1: Modèle MTCS-OrBAC

5.1.2 Sessions collaboratives

Une session collaborative est l'entité de base de notre modèle. Il s'agit d'une entité abstraite, comprenant un ensemble d'utilisateurs (appelés membres de la session), effectuant des tâches spécifiques, en accédant à des ressources partagées dans la session pour atteindre un objectif commun. Une session collaborative se caractérise par un ensemble de paramètres à savoir : son identifiant, son type, les membres de la session, les objets partagés dans la session et les états de la session. Une session collaborative est une instance d'une activité de collaboration.

D'une part, une session individuelle (telle qu'elle est définie dans le modèle RBAC [46]) est une instance d'activité établie par un seul utilisateur. Elle correspond à un utilisateur unique. Dans ce modèle, on considère que la session individuelle est une session collaborative comportant un seul utilisateur. On note CS est l'ensemble des sessions collaboratives possibles. La relation entre la session collaborative et le tenant est définie par la relation $CSOwner(cs) = t$, signifiant que le tenant t est le propriétaire de la session cs.

5.1.3 Types de session collaborative

Chaque session collaborative est instanciée à partir d'une activité de collaboration, nommée "Type de la session collaborative". Un tenant peut définir plusieurs types de session collaborative selon les activités de collaboration. Parmi

ces types, on peut trouver par exemple : (session publique, session privée, urgence dans la neurologie, urgence dans le cardiologie, etc). La relation entre le type de session collaborative et le tenant est définie comme suit : $TCSOwner(tcs) = t$ signifie que le type de la session tcs est défini par le tenant t. Par exemple, $TCSOwner(neuroEmergency) = SAMU$ représente la relation entre le type de la session urgence en neurologie $neuroEmergency$ et son propriétaire $SAMU$.

En outre, la relation entre la session collaborative, son type et son tenant est définie par la relation : $type(t, cs, tcs)$ signifiant que dans le tenant t, la session collaborative cs est de type tcs. Par exemple, $type(SAMU, cs1, neuroEmergency)$ signifie que dans le tenant $SAMU$ la session collaborative $cs1$ est instanciée à partir du type $neuroEmergency$. Notez que TCS est l'ensemble des types possibles de session collaborative. Dans ce modèle, nous considérons que toutes les sessions individuelles sont de type $Default\_Type$ ($Default\_Type \in TCS$).

5.1.4 Rôles

Un rôle est une collection d'utilisateurs ayant la même fonction dans le tenant. Un tenant peut définir plusieurs rôles alors qu'un rôle est associé à un seul tenant. L'administrateur du tenant est celui qui attribue les rôles aux utilisateurs. Ce modèle introduit la notion "rôle dans la session", c'est à dire, le rôle qu'un utilisateur donné a activé durant la session collaborative. Cet utilisateur peut activer un rôle lors de la session selon des contraintes définies par l'administrateur. Parmi ces contraintes, on peut trouver : les contraintes de séparation de pouvoirs, les contraintes de cardinalité (par exemple, on peut avoir qu'un seul rôle *neurologue* activé durant la session) et les contraintes liées au type de la session (Par exemple, un rôle X ne peut être activé que durant les sessions collaboratives de type Y).

La relation entre le rôle et son tenant est définie comme suit : $ROwner(r) = t$ signifie que le rôle r est défini par le tenant t. Par exemple, $ROwner(neurologiste) = SAMU$ représente la relation entre le rôle *neurologist* et son tenant $SAMU$. En outre, nous définissons la relation $Assign$ entre le tenant, l'utilisateur et le rôle : $AssignUser(t, u, r)$, signifie que dans le tenant t, l'utilisateur est affecté au rôle r. De même, on note que R est l'ensemble des rôles possibles.

5.1.5 Activités

De la même manière que OrBAC, ce modèle classe les actions dans des catégories appelées « activités ». Une activité est définie comme une entité abstraite d'actions ; Il s'agit soit d'une ou de plusieurs actions (opérations) ayant un objectif commun dans un tenant. La relation entre l'activité et le tenant est définie par la fonction : $AOwner(t) = a$ signifie que le tenant t est le propriétaire de l'activité a. Par exemple, $AOwner(decision) = SAMU$, signifie que l'activité *decision* a été définie par le tenant $SAMU$. De plus, nous utilisons la relation *consider* entre le tenant, l'action et l'activité : $consider(t, ac, a)$, signifiant que dans le tenant t, l'action ac appartient à l'activité a.

5.1.6 Vues

De même, le modèle OrBAC classe les objets dans des catégories appelées « vues ». Une vue est un groupe des objets satisfaisant une propriété commune. La relation entre la vue et le tenant est définie par la fonction : $VOwner(v) = t$ signifie que le tenant t est le propriétaire de la vue v. Par exemple, $VOwner(MR) = HA$, signifie que la vue dossier médical MR est définie par le tenant HA. De plus, dans OrBAC, la relation entre le tenant, l'objet et la vue est définie comme suit : $AssignObject(t, o, v)$, signifie que dans le tenant t, l'objet o est affecté par l'administrateur à la vue v. De même, le modèle introduit la notion "vue dans la session", c'est à dire, les objets appartenant à une vue donnée et qui sont partagés dans la session collaborative. Notez que V représente l'ensemble des vues possibles.

5.1.7 Contexte

La notion de contexte est utilisée pour exprimer une situation spécifique et des circonstances concrètes qui conditionnent la validité d'une règle. La relation entre le contexte et le tenant est définie par la fonction : $COwner(c) = t$, signifiant que le tenant t est celui qui a défini le contexte c. Par exemple, $COwner(urgence) = SAMU$, signifie que le contexte *urgence* a été défini par le tenant $SAMU$. En outre, nous redéfinissons la relation *hold* entre le tenant, la session collaborative, l'utilisateur, l'action, l'objet et le contexte comme suit : $hold(t, cs, u, ac, o, c)$ signifie que dans le tenant t, le contexte c est vrai pour le 4-tuple (cs, u, ac, o).

5.1.8 Utilisateurs

Un utilisateur est l'entité qui peut exécuter des actions sur des objets dans le tenant. Un utilisateur peut jouer un ou plusieurs rôles dans un tenant. La relation entre l'utilisateur et le tenant est définie comme suit : $UOwner(u) = t$ signifiant que le tenant t est le propriétaire de l'utilisateur u. Par exemple, la $UOwner(user5) = SAMU$ veut dire que l'utilisateur $user5$ est détenu par le tenant $SAMU$. De plus, nous définissons une nouvelle relation entre l'utilisateur et la session collaborative : $member(u, cs)$, signifiant que l'utilisateur u est membre de la session collaborative cs. D'une part, nous redéfinissons la relation $Empower()$ entre le tenant, l'utilisateur, le rôle et la session collaborative comme suit : $Empower(t, u, r, cs)$, signifiant que dans le tenant t, l'utilisateur u a activé le rôle attribué r lors de la session cs.

$Empower(t, u, r, cs)\ si :\ AssignUser(t, u, r) \wedge member(u, cs) \wedge CA$

Notez que, CA représente les contraintes d'activations d'un rôle.

5.1.9 Objets

Un objet est la ressource à laquelle la politique de sécurité contrôle son accès à partir des utilisateurs non autorisés. Un objet peut être des données, des documents, des fichiers, des répertoires, des périphériques et des ports. La relation entre l'objet et son propriétaire tenant est définie par la fonction : $OOwner(o) = t$ signifiant que le tenant t est le propriétaire de l'objet o. Par exemple, la relation $OOwner(mr5, HA)$ le tenant HA est le propriétaire du dossier médical $mr5$. De même, on note que O est l'ensemble des objet possibles. En outre, nous définissons une nouvelle relation de partage entre l'objet et la session collaborative : $shared(o, cs)$ signifiant que l'objet o est partagé dans la session collaborative cs. De plus, nous redéfinissons la relation $use()$ entre le tenant, l'objet, la vue et la session collaborative comme suit : $use(t, o, v, cs)$, signifiant que dans le tenant t, l'objet o fait partie de la vue v durant de la session cs.

$use(t, o, v, cs)\ si :\ AssignObject(t, o, v) \wedge shared(o, cs)$

Dans ce modèle, nous considérons que tous les objets d'un tenant sont partagés par défaut dans les sessions individuelles (de type $Default\_type$). Nous exprimons cette hypothèse par le formalisme suivant :

$\forall t \in T$, $\forall o \in O$ et $\forall s \in CS$ tel que :

$Type(t, s, Default\_type) \wedge OOwner(o, t)$
$\rightarrow shared(o, s)$;

5.1.10 Permissions

Une permission est l'autorisation d'effectuer une opération donnée par utilisateur sur un objet donné. Dans le modèle MTCS-OrBAC, une permission abstraite est définie par le 6-uplet des entités (tenant, type de la session, rôle, activité, vue et contexte). Tandis que dans le niveau concret, une permission est définie par le 4-uplet (session collaborative, utilisateur, action, objet). Pour dériver les permissions d'accès du niveau abstrait au niveau concret, nous utilisons formellement le formalisme de dérivation suivant :

$\forall\ t, tcs, r, a, v, c, cs, u, ac$ et o un tenant, un type de session collaborative, un rôle, une activité, une vue, un contexte, une session collaborative, un utilisateur, une action et un objet respectivement.

$$\begin{aligned} &Permission(t, tcs, r, a, v, c) \wedge \\ &Type(t, cs, tcs) \wedge \\ &Empower(t, u, r, cs) \wedge \\ &use(t, o, v, cs) \wedge \\ &Consider(t, ac, a) \wedge \\ &hold(t, cs, u, ac, o, c) \\ &\rightarrow IsPermitted(cs, u, ac, o) \end{aligned} \tag{5.1}$$

Ce formalisme s'interprète de la façon suivante : l'utilisateur u est autorisé à exécuter l'action ac sur l'objet o, si seulement si : (1) si dans le tenant t, le rôle r est autorisé à effectuer l'activité a sur la vue v pour le contexte c et lors d'une session collaborative de type tcs. (2) dans le tenant t, la session collaborative cs est de type tcs ; (3) dans le tenant t, l'utilisateur u active le rôle r durant la session cs ; (4) Dans le tenant t, l'objet o appartient à la vue V de la session cs ; (5) dans le tenant t l'action ac fait partie de l'activité a ; et (6) dans le tenant t, le contexte c est vrai pour le 5-uplet (t, cs, u, a, o).

5.2 Management de la session collaboration dans MTCS-OrBAC

Dans cette section, nous formalisons les règles de sécurité liées aux actions utilisées pour gérer les sessions collaboratives telles que : Initier une session collaborative, Inviter les utilisateurs à y adhérer, joindre / se déconnecter de la session collaborative et partager une ressource dans la session collaborative.

5.2.1 Initiation d'une session collaborative

La règle de sécurité liée à l'action "initier une session collaborative du type CSessionType", est exprimée comme suit :
$Permission(Te, Default\_type, Urgentiste, Initiate, CSessionType, Context)$.

Par exemple : nous définissons la permission de contrôle d'accès autorisant l'urgentiste du $SAMU$ à initier une session collaborative $cs1$ de type $NeuroEmergency$.

$$\begin{aligned} & Permission(SAMU, Default\_Type, Urgentiste, \\ & Initiate, NeuroEmergency, c) \wedge \\ & \exists s1 \in CS \ : \ Type(SAMU, s1, Default\_Type) \wedge \\ & Empower(SAMU, u, Urgentist, s1) \wedge \\ & use(SAMU, cs1, NeuroEmergency, s1) \wedge \\ & consider(SAMU, a, Initiate) \wedge \\ & hold(SAMU, s1, u, a, cs1, c) \\ & \rightarrow IsPermitted(s1, u, a, cs1) \end{aligned} \tag{5.2}$$

5.2.2 Joindre la session collaborative

La règle de de contrôle d'accès liée à l'action "initier une session collaborative du type CSType" est exprimée comme suit :
$Permission(Te, Default\_type, neurologist, join, CSType, Context)$

Par exemple : On définie la permission de contrôle d'accès permettant au rôle

neurologist de joindre la session collaborative $cs1$ de type $NeuroEmergency$.

$$\begin{aligned}
&Si \quad Permission(SAMU, Default_Type, neurologist,\\
&join, neuroEmergency, c) \wedge\\
&\exists s1 \in CS \ : \ Type(SAMU, S1, Default_Type) \wedge\\
&Empower(SAMU, u, neurologist, s1) \wedge\\
&use(SAMU, cs1, NeuroEmergency, s1) \wedge\\
&consider(SAMU, a, join) \wedge\\
&hold(SAMU, s1, u, a, cs1, c)\\
&Tel\,que\, c = GetInvitaion(cs, u)\\
&\rightarrow IsPermitted(s1, u, a, cs1)
\end{aligned} \tag{5.3}$$

$GetInvitaion(cs, u)$: cela signifie que l'initiateur de la session envoie une invitation à l'utilisateur u pour rejoindre la session collaborative cs.

5.2.3 Partage une ressource dans une session collaborative

La règle de sécurité relative à l'action "partage une ressource dans une session collaborative est exprimée comme suit :
$Permission(Te, Ty, doctor\_ha, share, v, c)$

Par exemple : nous définissions la permission de contrôle d'accès permettant au médecin de l'hôpital d'accueil $doctor\_ha$ de partager un dossier médical $mr1$ dans une session collaborative $cs1$ de type $NeuroEmergency$.

$$\begin{aligned}
&Permission(SAMU, neuroEmergency, doctor_ha,\\
&share, MR, c) \wedge\\
&\exists cs \in CS \ : \ Type(SAMU, cs, neuroEmergency) \wedge\\
&Empower(SAMU, u, doctor_h a, cs) \wedge\\
&use(SAMU, mr1, MR, cs) \wedge\\
&consider(SAMU, a, share) \wedge\\
&hold(SAMU, cs, u, a, mr1, c)\\
&\rightarrow IsPermitted(cs, u, a, mr1)
\end{aligned} \tag{5.4}$$

5.3 Modèle MTCS-OrBAC pour un environnement multi-tenant

Dans cette section, nous considérons un environnement multi-tenant, ce qui signifie que les utilisateurs participant à la collaboration et les ressources partagées peuvent appartenir aux différents tenants. Dans un système multi-tenant, notre modèle de contrôle d'accès doit tenir en compte les deux situations :

— La première situation : est qu'un utilisateur appartenant à un tenant A accède à des services fournis par un autre tenant B. Dans ce cas, le tenant B définit un rôle dans sa politique locale permettant à cet utilisateur d'accéder à ces services. Ceci est assuré par l'utilisation de la relation de confiance de type b définie définie dans [54]. De plus, cette relation de confiance nommée $TrustRoles()$ sera étendue pour supporter la notion du contexte ;

— La deuxième situation : est qu'un utilisateur du tenant C partage une ressource (appartenant à ce tenant C), pour une durée limitée, dans une session collaborative se déroulant dans un autre tenant A. Dans ce cas, une fois cette ressource est partagée dans la session, elle sera soumise aux règles de contrôle d'accès définies par A (le propriétaire de la session). Pour cela, on définit une nouvelle relation de confiance $TrustViews()$ qui sera établie entre tenants, pour supporter cette notion de partage de la ressource dans la session.

5.3.1 Relation TrustRoles()

La relation $TrustRoles()$ est une relation de confiance entre les tenants $(TT) \subseteq T \times T$ est une relation réflexive plusieurs-à-plusieurs entre le fournisseur de confiance (Truster) Tr et le bénéficiaire de confiance Te, $TrustRoles(tr, te : T) \rightarrow 2^R$, tels que R est l'ensemble des rôles possibles et T est l'ensemble des tenants possibles. Cette relation signifie que le fournisseur tr autorise le bénéficiaire à utiliser certains rôles définis par tr pour que le bénéficiaire te puisse affecter ces utilisateurs aux rôles de tr.

Par exemple (figure 5.2) : $TrustRoles(SAMU, CHU) = \{neuroSamu, radioSamu\}$, signifie que le tenant $SAMU$ autorise le tenant CHU pour qu'il utilise les deux rôles $neuroSamu$ et $radioSamu$. En utilisant cette relation de confiance,

la relation $AssignUser()$ entre le tenant, le rôle et l'utilisateur sera redéfinie de la manière suivante : $\forall\ te, r$ et u tenant, rôle et utilisateur respectivement ;

$$\begin{aligned} &AssignUser(te, u, r)\ si \\ &UOwner(u) = ROwner(r) = te \vee \\ &(\exists t\ un\ tenant\ tel\ que,\ t = ROwner(r) \wedge \\ &UOwner(u) = te \wedge r \in TrustRoles(t, te)\ ) \end{aligned} \tag{5.5}$$

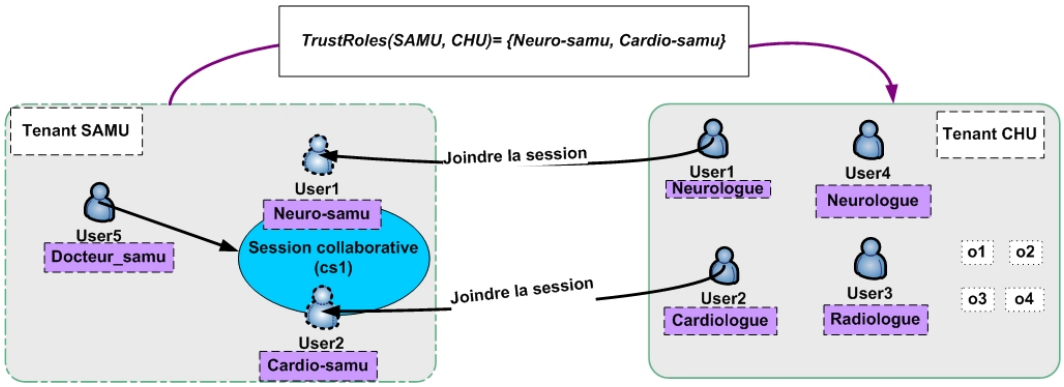

FIGURE 5.2: Relation trustRoles()

En outre, nous avons étendu la relation de confiance $TrustRoles$ pour qu'elle prend en considération la notion du contexte. Cette relation devient : $TrustRoles\ (tr, te : T) \rightarrow 2^{(R,C)}$. On reprend l'exemple précédent : $TrustRoles(SAMU, CHU) = (neuroSamu, c1), (radioSamu, c2)$, signifie que le tenant $SAMU$ autorise le tenant CHU afin qu'il utilise les deux rôles $neuroSamu$ et $radioSamu$ dans le contexte $c1$ et $c2$ respectivement.

5.3.2 Relation TrustViews()

La relation de confiance $TrustViews$ entre les tenants $(TT) \subseteq T \times T$ est une relation plusieurs-à-plusieurs entre le truster (fournisseur de confiance) tr et le trustee(bénéficiaire de confiance) te. Cette relation est définie comme suit : $TrustViews(tr, te : T) \rightarrow 2^{(A,V)}$, tels que T, A et V est l'ensemble des tenants, activités et vues respectivement. Cette relation signifie que les objets du tenant tr, appartenant à la vue $vi \in V$, peuvent être partagés dans des sessions collaboratives cs se déroulant dans le tenant te uniquement pour les actions de l'activité $ai \in A$.

Par exemple (figure 5.3), $TrustViews(HA, SAMU) = \{(readWrite, DM), (read, Scan)\}$, signifie que durant une session collaborative $cs1$ se déroulant dans le tenant $SAMU$, un utilisateur du tenant HA peut partager dans cette session des objets appartenant à la vue DM et $Scan$ seulement pour les actions de l'activité $readWrite$ et $read$ respectivement. Une fois un de ces objets est partagé dans la session $cs1$, il sera soumis aux règles de la politique locale du tenant $SAMU$. En utilisant cette relation de confiance $TrustViews()$, la relation $AssignObject()$ entre le tenant, l'objet et la vue sera redéfinie de la manière suivante : $\forall\ tr, v(te)$ et o tenant, objet et vue définie par tr pour le tenant te respectivement ;
tel que $\exists a \in A, (ai, v(Te)) \in trustViews(tr, te)$

$$\begin{aligned} &AssignObject(tr, o, v(te))\ si \\ &OOwner(o) = VOwner(v(te)) = tr; \end{aligned} \tag{5.6}$$

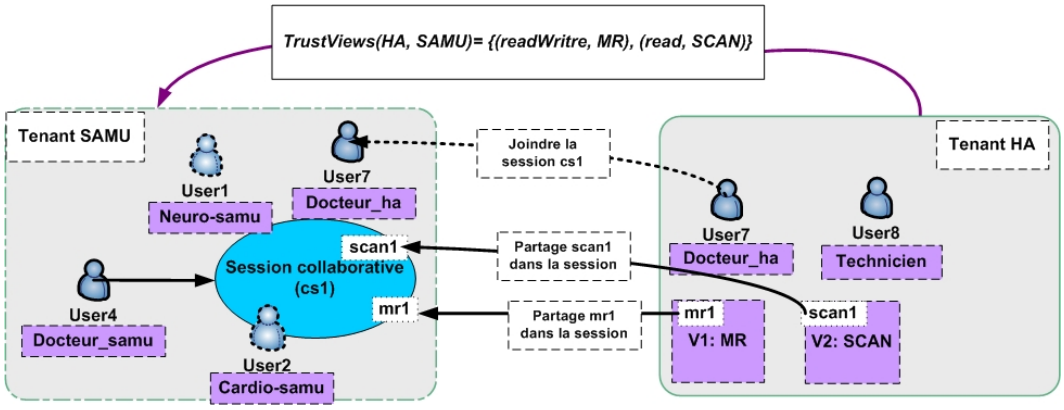

FIGURE 5.3: Relation trustViews()

Notez que la définition de cette relation de confiance n'implique pas que les ressources sont partagées, mais cela signifie qu'elles peuvent être partagés lorsqu'un utilisateur autorisé veut les partager dans une session collaborative. Pour cela, nous redéfinissons la relation $use()$ entre le tenant, l'objet, la vue et la session collaborative. $use(tr, o, v(te), cs) = True$: Si un utilisateur autorisé partage cet objet o dans une session collaborative cs se déroulant dans le te. Ceci est

exprimé en utilisant le formalise suivant :

$$\begin{aligned} &use(tr, o, v(te), cs)\ si \\ &AssignObject(tr, o, v(te))\ \wedge\ shared(o, cs1) \wedge \\ &CSOwner(cs1)\ =\ te \wedge \\ &\exists a \in A\, telle\, que\, (a, v(te)) \in trustViews(tr, te); \end{aligned} \tag{5.7}$$

5.3.3 Permissions dans un environnement multi-tenant

En utilisant les relations de confiance introduites dans les sous-sections précédentes $trustRoles()$ et $TrustViews$, nous pouvons à présent redéfinir des permissions de contrôle d'accès appliquées à tel tenant dans le modèle MTCS-OrBAC : $\forall$ t, tcs, r, a, v et c tenant, Type de session collaborative, rôle, activité, vue et contexte respectivement ;

$$\begin{aligned} &permission(t, tcs, r, a, v, c)\ si \\ &(TCSOwner(tcs) = ROwner(r) = AOwner(a) = VOwner(v) = \\ &COwner(c)) = t \vee \\ &(TCSOwner(tcs) = AOwner(a) = VOwner(v) = COwner(c) = t) \wedge \\ &(r \in trustRoles(ROwner(r), t) \vee \\ &(TCSOwner(tcs) = ROwner(r) = COwner(c)) = t) \wedge \\ &(\exists\, t1\, un\, tenant\, tel\, que,\ AOwner(a) = VOwner(v) = t1 \wedge \\ &(a, v) \in trustViews(t1, t)) \end{aligned} \tag{5.8}$$

Pour dériver les permissions d'accès du niveau abstrait au niveau concret, nous redéfinissons la dérivation des permissions en utilisant le formalisme suivant :

$\forall$ $te, tcs, r, a, v, c, cs, u, ac$ et o un tenant, un type de session collaborative, un rôle, une activité, une vue, un contexte, une session collaborative, un utilisateur,

une action et un objet respectivement.

$$
\begin{aligned}
&Si \quad Permission(te, tcs, r, a, v, c) \wedge Type(te, cs, tcs) \quad \wedge \\
&Empower(te, u, r, cs) \wedge \\
&((use(te, o, v, cs) \wedge Consider(te, ac, a)) \vee \\
&((\exists\, tr \in T\,tel\,que,\ AOwner(a) = VOwner(v) = tr \wedge \\
&(a, v) \in trustViews(tr, te)) \wedge \\
&use(tr, o, v, cs) \wedge Consider(tr, ac, a))) \wedge \\
&hold(te, cs, u, ac, o, c) \\
&\rightarrow IsPermitted(cs, u, a, o)
\end{aligned}
\tag{5.9}
$$

Ce formalisme s'interprète de la façon suivante : l'utilisateur u est autorisé à exécuter l'action a sur l'objet o, si seulement si : (1) si dans le tenant te, le rôle r est autorisé à effectuer l'activité a sur la vue v dans le contexte C est vrai, et pour des sessions collaboratives de type CST. (2) la session collaborative cs est de type tcs ; (3) dans le tenant te, le rôle r est attribué à l'utilisateur u ; (4) Dans le tenant te, l'objet o fait partie de la vue v ; (5) dans le tenant te l'action ac fait partie de l'activité a ; (6) Ou bien il existe un tenant tr qui le propriétaire des entités a et v, tel que tr fait confiance à te pour que ce dernier utilise les deux entités a et v ; (7) dans le tenant tr, l'objet o fait partie de la vue v ; (8) dans le tenant tr l'action a fait partie de l'activité a ; (9) dans le tenant te, si le contexte c est vrai pour le quintuplet (te, cs, u, a, o).

5.4 Implémentation

Dans cette section, nous allons décrire l'implémentation de notre approche sur le cloud open source Openstack [2]. Ce cloud comme il a été décrit dans le chapitre précédent, est un ensemble des composants open source permettant de déployer des infrastructures de cloud computing (le service d'infrastructure : IaaS). Openstack possède une architecture modulaire composée de plusieurs projets corrélés (Nova, Swift, Glance, Keystone...) qui permettent de fournir et contrôler les différents services tels que les machines virtuelles, la puissance de calcul, le stockage ou encore le réseau.

Dans notre implémentation, nous nous intéressons plus particulièrement à la composante de stockage d'Openstack Swift [3]. Cette composante est un système

de stockage de fichiers, vidéos, données, données analytiques, contenus web, sauvegardes, images, instances de machines virtuelles et autres données non structurées. Swift utilise pour les sauvegardes plusieurs serveurs. Si un serveur ou un disque dur tombe en panne, Swift réplique ses données depuis des nœuds actifs du cluster dans de nouveaux emplacements.

Swift utilise le modèle de contrôle d'accès à base de listes (ACL) pour définir la politique de d'accès au niveau des objets stockés dans Swift. Les règles ACL de swift sont définies au niveau de la couche conteneur et supportent la notion des listes pour l'accès en lecture et en écriture à une ressource donnée. Cependant, le contrôle d'accès à base des listes (ACL) de Swift [11] ne permet pas de spécifier des règles d'accès plus contextuelles telles que celles définies dans notre approche à savoir : les relations de confiance entre les tenants, la spécification de la dérivation des autorisations, les relations entre les entités du modèle et les contraintes liées à la collaboration.

Afin de surmonter cette limitation, nous proposons d'introduire un nouveau module "Module OrBAC" à l'environnement Swift (voir la figure 5.4). Ce module OrBAC se compose de quatre composants : le composant de la politique de décision, le composant de la politique d'information, le composant du management session et le composant des règles d'accès.

Dans ce module, l'administrateur peut spécifier des permissions abstraites dans le composant des règles d'accès, affecter les utilisateurs, les actions et les objets aux rôles, aux activités et à la vue, respectivement, dans le composant de la politique d'information. De plus, dans ce composant, l'administrateur peut définir d'autres relations essentielles dans notre approche par exemple la relation entre le tenant et les différentes autres entités (les rôles, les activités, les vues, les types de session collaborative, les utilisateurs et les objets). En outre, l'administrateur spécifie les paramètres dynamiques et les contraintes liées à la politique de collaboration dans le composant du management de la session afin de supporter le concept de la session collaborative.

Lorsqu'un utilisateur envoie une requête demandant l'accès à une ressource stockée dans le cloud Swift, le composant de la politique de décision évalue cette requête conformément aux règles de la politique d'accès et les paramètres des entités afin de décider si l'utilisateur est autorisé ou non à accéder à cette ressource. Si l'utilisateur est autorisé à effectuer cette action, le composant de la politique de

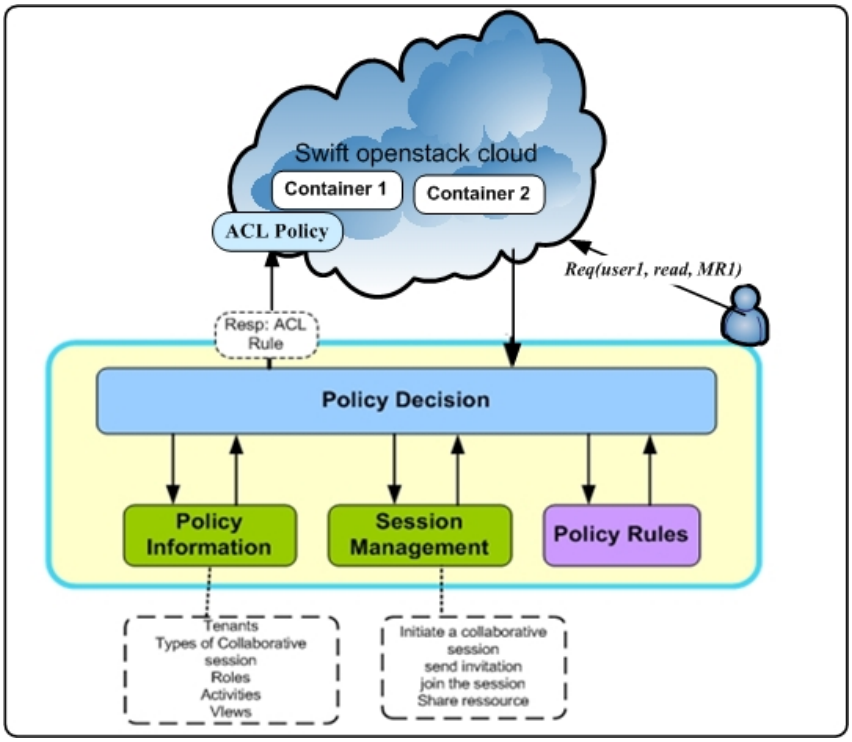

FIGURE 5.4: Implémentation du modèle MTCS-OrBAC

décision exécutera une commande ACL (comme ci-dessous) pour attribuer cette autorisation à l'utilisateur dans l'environnement Swift :

- *swift post -w '<ACL>' <container> [-A AUTH_ URL] [-U user] [-K password]*
- *curl -X <PUT|POST> -i -H "X-Auth-Token : <TOKEN>" -H "X-Container-Write : <ACL>" <STORAGE_ URL>/<container>*
- *curl -H "X-Auth-Token : <TOKEN>" -X PUT <STORAGE_ URL>/<container>/<object> –data-binary @<filename>*

5.5 Conclusion

Dans ce chapitre, nous avons proposé une extension du modèle OrBAC pour : (1) supporter l'accès inter-tenant et la collaboration entre plusieurs tenants ; (2) Introduire au modèle OrBAC la notion de session collaborative. Le principal objectif du modèle MTCS-OrBAC est d'étendre le modèle OrBAC pour qu'il prenne en compte le concept de la "multi-tenancy" et l'aspect dynamique de la

session collaborative. Dans le chapitre suivant, nous allons présenter un autre modèle MTCS-TRBAC basé sur RBAC pour contrôler l'accès aux ressources partagées lors d'une session collaborative se déroulant dans un environnement multi-tenant.

Chapitre 6

Modèle session collaborative multi-tenant pour RBAC et TRBAC : MTCS-RBAC et MTCS-TRBAC

Dans le chapitre précédent, nous avons proposé MTCS-OrBAC [35], pour assurer le contrôle d'accès aux ressources échangées lors d'une session collaborative se déroulant sur un environnement multi-tenants. Cette approche est basée sur le modèle de contrôle d'accès OrBAC : (A Organization Based Access control Model) en introduisant de nouvelles entités comme le tenant, la session collaborative et le type de la session collaborative.

De même, ce chapitre présente un autre modèle MTCS-RBAC (Modèle session collaborative multi-tenant pour RBAC) pour contrôler l'accès aux ressources partagées lors d'une session collaborative se déroulant dans un environnement multi-tenant [36]. Ce modèle est une extension du modèle RBAC (Role Based Access Control) en intégrant de nouvelles entités comme le tenant, la session collaborative et le template de la session collaborative. Ceci est dans le but de répondre aux exigences de contrôle d'accès liées à la fois à la multi-tenacy et à la session collaborative. De plus, cette approche introduit des relations de "trust" pour établir la confiance entre les différents tenants participant à la collaboration, afin d'assurer une collaboration sécurisée dans un système multi-tenant.

Puis, nous étendions ce modèle pour qu'il puisse supporter le concept du «

task » et « workflow » dans un environment collaboratif multi-Tenant [37]. Le modèle étendu est appelé MTCS-TRBAC : Modèle session collaborative multi-tenant pour TRBAC [42]. Enfin, nous présentons une spécification formelle de notre approche pour une application collaborative pour le télé-diagnostic dans le domaine de la neuroscience. Cette spécification a été implémentée et vérifiée sur la composante de stockage swiftstack de la plateforme Openstack.

Le modèle RBAC a été choisi dans ce travail en tant que modèle de base pour sa capacité d'abstraction permettant de définir de politiques de contrôle d'accès centrées sur le rôle. Dans ce chapitre, tout d'abord, nous commençons par présenter les motivations qui nous ont permis de proposer ces modèles. Ensuite, nous proposons le modèle MTCS-RBAC : une extension du modèle RBAC en prenant en compte les concepts "Session collaborative" et la multi-tenancy. Par la suite, nous étendions le modèle MTCS-RBAC en introduisant le concept "Tâche et workflow" pour supporter les processus collaboratifs s'exécutant sur un environnement multi-tenant. C'est-à-dire que les tâches d'un processus collaboratif sont accomplies par les différents tenants participants à la collaboration. Dans ce qui suit, nous présentons les concepts de base du modèle MTCS-RBAC dans la première section. Dans la deuxième section, nous allons voir le modèle MTCS-RBAC.

6.1 Concepts de base du modèle MTCS-RBAC

6.1.1 Entités de base du modèle MTCS-RBAC

Dans cette section, nous présentons les entités de base du modèle MTCS-RBAC (Multi Tenant Collaborative Session Role Based Access Control) [36] est un modèle pour contrôler l'accès aux ressources partagées lors d'une session collaborative se déroulant sur un environnement multi-tenant. Il est basé sur le modèle de contrôle d'accès RBAC : Role Based Access Control en introduisant de nouvelles entités comme le tenant, la session collaborative, le Template de la session collaborative et le type objet. De plus, cette approche introduit des relations pour établir la confiance entre les différents tenants participant à la collaboration, afin d'assurer la collaboration dans un environnement multi-tenant. L'objectif principal du modèle MTCS-RBAC est d'étendre le modèle RBAC pour qu'il supporte le

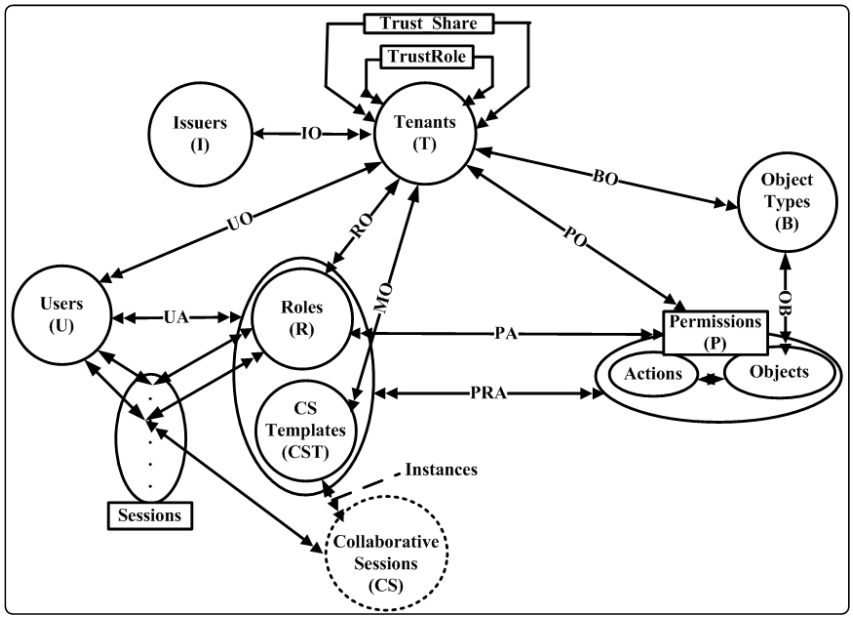

FIGURE 6.1: Modèle MTCS-RBAC

concept de la "multi-tenancy" et l'aspect dynamique de la session collaborative.

Dans ce modèle, comme il est schématisé dans la figure 6.1, nous avons plusieurs entités : clients (Issuers) (I), tenants (T, sessions collaboratives (CS), temples de sessions collaboratives (CST), rôles (R), permissions (P), utilisateurs (U), sessions (S), des actions (C), des objets (O) et des types d'objets (B). Un client a un ou plusieurs tenants. Un tenant est associé à un seul client par une relation un-à-plusieurs. En outre, il existe des relations un-à-plusieurs entre les différentes entités et leur tenant propriétaire. Dans ce qui suit, nous allons décrire les différentes entités du modèle.

a. Clients du Cloud (Issuers)

Un client du cloud est une organisation ou un individu qui utilise des services offerts par un fournisseur cloud. C'est un client qui est capable d'administrer ses propres tenants dans le cloud. Le client du Cloud consulte la liste des services fournis par le fournisseur du Cloud ; puis il choisit les services appropriés. Ainsi, le client établit un contrat $SLA´ServiceLevelAgreement˘$ des services avec le fournisseur. On note I est l'ensemble des issuers possibles.

b. Tenants

Un tenant est une partition virtuelle d'un service cloud ou d'une instance d'application logicielle fournie par le fournisseur des services cloud à l'issuer. Un

issuer est associé à plusieurs tenants alors qu'un tenant appartient à un seul issuer. La relation entre le tenant et l'issuer est définie comme suit : $(t, i) \in TO$ signifie que l'issuer i est le propriétaire du tenant t. Par exemple, $(CHU, clientCHU) \in TO$ représente la relation entre le tenant CHU et son issuer propriétaire *clientCHU*.

Un tenant peut avoir ses propres utilisateurs et objets. De plus, un tenant peut définir ses propres entités à savoir : le template de la session collaborative, le rôle et le type d'objet. L'administrateur du tenant affecte les utilisateurs et les objets aux rôles et types d'objet respectivement. Cet administrateur définit la politique de contrôle d'accès locale en se basant sur ces entités. On note T est l'ensemble des tenants possibles.

c. Sessions collaboratives

Une session collaborative est l'entité de base de notre modèle. Il s'agit d'une entité abstraite, comprenant un ensemble d'utilisateurs (appelés membres de la session), effectuant des tâches spécifiques, en accédant à des ressources partagées pour atteindre un objectif commun. Une session collaborative se caractérise par un ensemble de paramètres à savoir : son identifiant, son type, les membres de la session, les objets partagés dans la session et les états de la session. Une session collaborative est une instance d'une activité de collaboration.

Dans ce modèle, on considère qu'une session collaborative est associée à un ou plusieurs sessions individuelles alors qu'une session individuelle peut être connectée à une ou plusieurs sessions collaboratives. Une session individuelle (telle qu'elle est définie dans le modèle RBAC [46]) est une instance d'activité établie par un seul utilisateur. Chaque session individuelle est un mappage d'un utilisateur à un sous-ensemble des rôles qui lui sont attribués. On note CS est l'ensemble des sessions collaboratives possibles. La relation entre la session collaborative et le tenant est définie par la fonction : $CSOwner(cs) = t$, signifiant que le tenant t est le propriétaire de la session cs.

d. Templates de la session collaborative

Un template de session collaborative représente une activité de collaboration [4]. Une session collaborative est créée et instanciée à partir d'un template donnée. Nous pouvons avoir différents templates de sessions collaboratives : (template de session publique, template de session privée et urgence dans la neurologie). Un template de session collaborative appartient à un seul tenant, tandis qu'un tenant

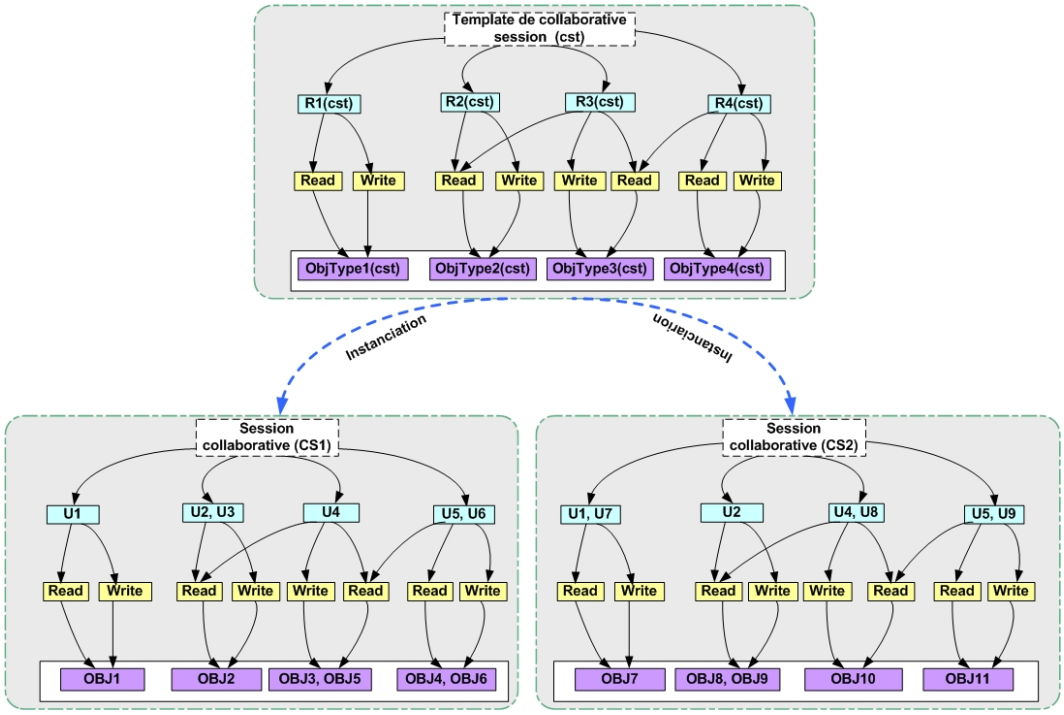

FIGURE 6.2: Template d'une session collaborative

peut définir plusieurs templates. Un template de session collaborative est défini par le 3-uplet (cst, R, B) :

- cst : Identifiant du template ;
- R : ensemble de rôles liés au template ;
- B : ensemble de types d'objets relatifs au template ;

Dans la figure 6.2, nous avons un template de session collaborative cst à partir duquel les sessions collaboratives $cs1$ et $cs2$ sont instanciées. Dans ce template cst, nous avons défini un ensemble de rôles $R1$, $R2$ et $R3$. Pour chaque rôle, nous associons un ensemble de permissions. Une permission est définie dans le template comme une action sur un type d'objet.

Lorsqu'un utilisateur instancie une session collaborative $cs1$ à partir du template cst, les entités utilisateurs et objets de la session $cs1$ seront instanciées à partir des entités rôles et types d'objet respectivement dans le template cst, lors du déroulement de la session. Par exemple, les utilisateurs $u2$ et $u3$ de la session collaborative $cs1$ sont instanciés à partir du rôle $R2$, ce qui signifie que ces utilisateurs jouent le rôle $R2$ dans cette session $cs1$. De même pour l'objet $obj1$ est instancié à partir du type d'objet $ObjType1$. On note M est l'ensemble des

templates possibles. La relation entre le template de session collaborative et le tenant est définie comme suit : $CSTOwner(cst) = t$ signifie que le template cst est défini par le tenant t.

e. Rôles

Un rôle est une collection d'utilisateurs ayant la même fonction dans le tenant. Un tenant peut définir plusieurs rôles alors qu'un rôle est associé à un seul tenant. La relation entre le tenant et le rôle est définie comme suit : $ROwner(r : R) \rightarrow T$, est une fonction mappant le rôle r à son propriétaire t.

f. Utilisateurs

Un utilisateur est l'entité qui peut effectuer des actions sur l'objet dans le tenant. Dans ce modèle, on considère qu'un utilisateur peut jouer différents rôles dans différents tenants. La relation entre l'utilisateur et la session collaborative est définie par la fonction : $membre(u, cs)$, signifie que l'utilisateur u est membre de la session collaborative cs. La relation entre le tenant et l'utilisateur u est définie ainsi : $UOwner(u : U) \rightarrow T$, est une fonction mappant l'utilisateur u à son tenant propriétaire.

g. Sessions

Une session individuelle (telle qu'elle est définie dans le modèle RBAC [46]) est une instance d'activité établie par un utilisateur.

h. Objets

Un objet est la ressource à laquelle la politique de sécurité contrôle son accès à partir des utilisateurs non autorisés. Un objet est peut être des données, des fichiers, des répertoires, des périphériques, des ports, etc. Notez que O est l'ensemble des objets possibles. La relation entre l'objet et la session collaborative : $shared(o, cs)$, signifiant que l'objet o est partagé dans la session collaborative cs. La relation entre le tenant et l'objet est définie ainsi : $OOwner(o : O) \rightarrow T$, est une fonction mappant le type d'objets ob à son tenant.

j. Types d'objet

MTCS-RBAC regroupe les objets en catégories d'objets. Un type d'objet représente un type de ressources ayant une propriété commune telles que les images de scanner. Chaque tenant définit ces propres types d'objets. De même, on note que OB est l'ensemble de type d'objet possible. La relation entre le tenant et le type d'objet est définie comme suit : $TOOwner(ob : OB) \rightarrow T$, est une fonction mappant le type d'objets ob à son tenant.

k. Actions

Les actions sont des opérations autorisées dans le système. Ces opérations incluent généralement créer, lire, modifier et supprimer. Une action est exécutée sur un objet par un utilisateur.

l. Permissions

Une permission est l'autorisation d'effectuer une action donnée par un utilisateur sur un objet donné. Les permissions définissent les droits d'accès accordés à un utilisateur ou à un rôle sur un objet ou une de ses propriétés. Une permission p est définie par le couple $p = (action, objet)$. Chaque administrateur du tenant définit ses propres permissions de manière autonome. De même, on note que P est l'ensemble des permissions possibles. La relation entre le tenant et la permission est définie comme suit : $POwner(p : P) \rightarrow T$, est une fonction mappant la permission p à son propriétaire.

Dans ce modèle, nous considérons un environnement multi tenant, ce qui signifie que les utilisateurs collaborant via des sessions collaboratives et les ressources partagées appartiennent au même ou bien aux différents tenants. Dans la figure 6.3, l'utilisateur $user5$ du tenant hôpital d'accueil (HA) rejoint une session collaborative se déroulant dans le tenant $SAMU$. Durant la collaboration, cet utilisateur pourrait partager certaines ressources du tenant HA dans cette session collaborative. Afin de prendre en compte la collaboration et le partage des ressources entre les tenants, nous réutilisons la relation de confiance $TrustRole()$ définie dans [54] et nous définissons une nouvelle relation de confiance $TrustShare()$.

6.1.2 Relation TrustRole()

La relation $TrustRole()$ est une relation de confiance établie entre les tenants $(TT) \subseteq T \times T$. Cette relation est réflexive plusieurs-à-plusieurs entre le tenant qui établit la confiance (Truster) tr et celui qui reçoit la confiance (Trustee) Te, $TrustRole(tr, te : T) \rightarrow 2^R$, tels que R est l'ensemble des rôles possibles et T est l'ensemble des tenants possibles. Cette relation signifie que le truster tr autorise le trustee te à utiliser certains rôles définis par tr pour que le trustee te puisse affecter ses utilisateurs aux rôles de Tr.

Par exemple (figure 6.3) : $TrustRole(SAMU, CHU) = \{neuroSamu, radioSamu\}$, signifie que le tenant $SAMU$ autorise le tenant CHU pour qu'il utilise les deux rôles $neuroSamu$ et $radioSamu$ définis par le $SAMU$. Dans ce cas, le CHU

peut attribuer ces rôles à ses propres utilisateurs qui peuvent ainsi accéder aux services fournis par $SAMU$. En utilisant cette relation de confiance, l'affectation d'un utilisateur u à un rôle r sera faite de la façon suivante : $\forall\ te, r$ et u tenant, rôle et utilisateur respectivement ;

$$\begin{aligned}&(u, r) \in UA\, si\\&(\,UOwner(u) \,=\, ROwner(r) \,=\, te \vee\\&(\,\exists tr \in T\, tel\, que,\, T \,=\, ROwner(r) \wedge\\&UOwner(u) \,=\, Te \,\wedge\, r \in TrustRole(T, Te)\,))\end{aligned} \tag{6.1}$$

Dans ce qui suit, nous allons définir une nouvelle relation de confiance $TrustShare()$

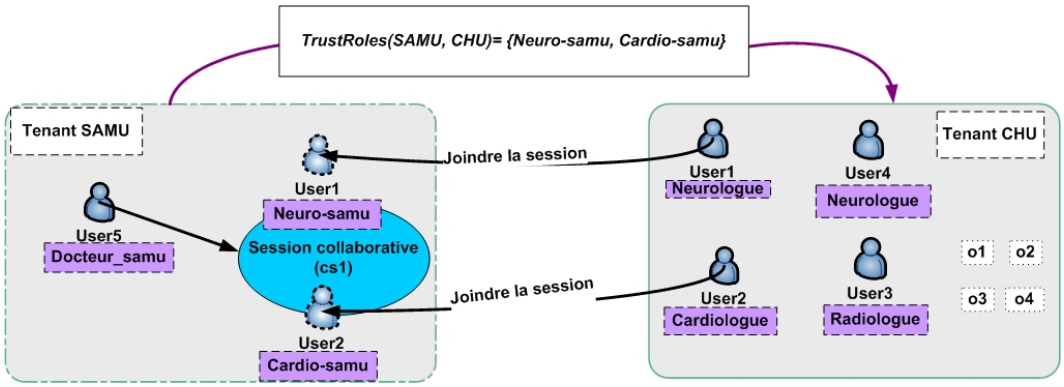

FIGURE 6.3: Relation trustRole()

établie en les collaborateurs (tenants). Cette relation permet à un utilisateur d'un tenant de partager sa propre ressource, pour une durée limitée, dans une session collaborative se déroulant dans un autre tenant.

6.1.3 Relation TrustShare()

La relation de confiance $TrustSahre$ entre les tenants $(TT) \subseteq T \times T$ est une relation plusieurs-à-plusieurs établie le truster tr et le trustee te. Cette relation est définie comme suit :
$TrustShare(tr, te\ :\ T) \,=\, \cup_k(perm_k(cs)) \,=\, \cup_{ij}(a_i, objType_j(cs))$, tels que a_i est une action, $objType_j$ est un type d'objet défini par le tenant tr et cs est une session collaborative. Cette relation signifie que les objets du type $objType_j$ peuvent être partagés dans des sessions collaboratives cs se déroulant dans le tenant te, uniquement pour l'action ai. La confiance est toujours établie par le

truster permettant au trustee d'utiliser les permissions dynamiques du truster. Par conséquent, le trustee peut affecter ces permissions dans la session $p(cs)$ à ses propres rôles dans la session $r(cs)$.

Dans cette exemple illustré dans la figure 19, La relation $TrustShare()$ établie entre le tenant HA et le tenant $SAMU$ a été définie comme suit :
$TrustShare(HA, EMS) = \{perm1(cs), perm2(cs), perm3(cs), perm4(cs)\}$, tels que :

- *perm1(cs)=(read, MR(cs))*
- *perm2(cs)=(write, MR(cs))*
- *perm3(cs)=(read, scan(cs))*
- *perm4(cs)=(write, scan(cs))*

Cette relation signifie que toutes les ressources de HA de type MR et *scan* peuvent être partagées dans des sessions collaboratives se déroulant dans le tenant $SAMU$, respectivement pour les actions $(Read, Write)$. L'administrateur du tenant $SAMU$ peut affecter ces permissions aux rôles dans la session $doctor\_samu(c\text{-}s)$ et $neurologue(cs)$. Durant la collaboration, un utilisateur autorisé du tenant HA peut partager les ressources (détenues par HA) $mr1$ et $scan1$ de type MR et *scan* respectivement dans la session collaborative cs se déroulant dans $SAMU$. Dans ce cas, une fois ces ressources sont partagées dans la session, elles seront soumises aux règles de contrôle d'accès définies par le tenant $SAMU$, au moment de déroulement de la session.

6.1.4 Modèle MTCS-RBAC

Le modèle MTCS-RBAC comporte les composants suivants :

- I, T, U, R, P, S, CS, M, O, B et TT sont un ensemble fini des issuers, tenants, utilisateurs, rôles, permissions, sessions, sessions collaboratives, templates de sessions collaboratives, objets, types d'objets et relations de confiance entre les tenants respectivement ;
- $TO \subseteq T \times I$, est une relation un-à-plusieurs mappant chaque tenant à son issuer propriétaire, $TOwner(t : T) \rightarrow I$, est une fonction mappant un tenant donné à son propriétaire telle que $TOwner(t) = i$ si $(t, i) \in TO$;
- $MO \subseteq M \times T$, est une relation un-à-plusieurs mappant chaque template de session collaborative à son tenant créateur, $CSTOwner(m : M) \rightarrow T$,

est une fonction mappant chaque template de session collaborative à son créateur t telle que $CSTOwner(m) = t$ si $(m, t) \in MO$;

— $UO \subseteq U \times T$, une relation un-à-plusieurs mappant chaque utilisateur à son tenant propriétaire, $UOwner(u : U) \rightarrow T$, est une fonction de mappage d'un utilisateur u à son propriétaire t telle que $userOwner(u) = t$ si $(u, t) \in UO$;

— $RO \subseteq R \times T$, une relation un-à-plusieurs mappant chaque rôle à son tenant, $ROwner(r : R) \rightarrow T$, est une fonction de mappage du rôle r à son tenant propriétaire t telle que $ROwner(r) = t$ si $(r, t) \in RO$;

— $PO \subseteq P \times T$, une relation un-à-plusieurs mappant chaque permission à son tenant, $POwner(p : P) \rightarrow T$, est une fonction qui prend en paramètre la permission p et retourne le tenant qui la définit t telle que $POwner(p) = t$ si $(p, t) \in PO$;

— $OT \subseteq O \times T$, une relation un-à-plusieurs mappant chaque objet à son tenant propriétaire, $ObjectOwner(o : O) \rightarrow T$, est une fonction de mappage d'un objet o à son tenant t telle que $ObjectOwner(o) = t$ si $(o, t) \in OO$;

— $BO \subseteq B \times T$, une relation un-à-plusieurs mappant chaque type d'objet à son tenant créateur, $ObjectTypeOwner(b : B) \rightarrow T$, une fonction de mappage d'un type d'objet b au tenant qui le définit t telle que $ObjectTypeOwner(b) = t$ si $(b, t) \in BO$;

— $OB \subseteq O \times B$, une relation un-à-plusieurs mappant chaque objet à son type, $objectType(o : O) \rightarrow B$, est une fonction de mappage de l'objet o à son type b telle que $objectType(o : O) = b$ si $(o, b) \in OB$;

— $Instances(cst : CST) \rightarrow 2^{CS}$, est une relation un-à-plusieurs mappant chaque template de session collaborative à ses instances de sessions collaboratives. Par exemple, la relation $Instance(NeuroEmergency) = \{cs1, cs2\}$ signifie que les sessions collaboratives $cs1$ et $cs2$ sont des instances du template $neuroEmergency$.

— $UA \subseteq U \times R$, une relation plusieurs-à-plusieurs mappant les utilisateurs à des rôles, telle que $(u, r) \in UA$ si seulement si $UOwner(p) = UOwner(r)$. D'autre part $(u, r(cs)) \in UA$ si $(u, r) \in UA \wedge member(u, cs)$;

— $PA \subseteq P \times R$ relation plusieurs-à-plusieurs mappant les rôles à des permissions, telle que $(p, r) \in PA$ si seulement si $POwner(p) = ROwner(r)$;

— $Users(cs : C) \rightarrow 2^U$, est une fonction de mappage de chaque session collaborative cs à un sous ensemble des utilisateurs qui sont membres de cette session, $Users(cs) = \cup_{u \in U}\{u|Member(u, cs)\}$;

— $Sessions(cs : C) \rightarrow 2^S$, est une fonction de mappage de chaque session collaborative cs à un sous ensemble de sessions :
$Sessions(cs) = \cup_{s \in S}\{s|Member(user(s), cs)\}$;

— Rôle dans la session $r(cs)$: est le rôle qui sera activé pour un utilisateur lorsqu'il rejoint la session collaborative cs. Dans la même session, nous pouvons avoir un ensemble de membres qui jouent les mêmes rôles ou de différents rôles : $roles : (u, r(cs)) \in UA$ si seulement si $(u, r) \in UA \wedge member(u, cs)$;

— Type d'objets dans la session $ObjType(cs)$: sont les objets de type $objType$ qui sont partagés dans la session collaborative cs ; $cs : obj \in objType(cs)$ if $(obj, objType) \in OB \wedge shared(obj, cs)$;

— Permission dans la session $p(cs)$: est une permission relative à l'accès à un objet partagé dans une session collaborative donnée cs.
$p(cs) = (a, objType(cs)) : (u, p(cs)) \in UP$ si $(u, r(cs)) \subseteq UA \wedge$
$(p(cs), r(cs), cst) \in PRA$;

— $PRA \subseteq P \times R \times M$, est une relation plusieurs-à-plusieurs-à-plusieurs entre la permission le rôle et le template, tels que $(p(cs), r(cs), m) \in PRA$ si seulement si $POwner(p) = ROwner(r) = CSTOwner(m) \wedge cs \in instances(m)$;

— $TrustRole(tr, te : T) \rightarrow 2^R$, est une nouvelle relation de confiance établie entre les tenants, mappant le truster et le trustee à un sous ensemble des rôles définis par le truster ;

— $TrustShare(tr, te : T) \rightarrow 2^P$, est une nouvelle relation de confiance établie entre les tenants, mappant le couple (truster, trustee) à un ensemble permissions dans la session (relatives aux ressources partagées dans la session).

Les règles de contrôle d'accès dans le modèle MTCS-RBAC sont formellement exprimées comme suit :
$\forall u \in U, p \in P, r \in R, cs \in si, cst \in CST, ta \in TA, obj \in O, objType \in B$ telles

que :

$$\begin{aligned} & si\, cs \in Instance(cst) \wedge \\ & (u, r(cs)) \in UA \wedge \\ & (r, ta) \in RTA \wedge \\ & (p(cs), ta(cs), cst) \in PRA \\ & \rightarrow (u, p(cs)) \in UP \\ & (p(cs) = (a, objType(cs))) \\ & \rightarrow (u, (a, obj)) \in UP \end{aligned} \tag{6.2}$$

— $[(u, r(cs)) \in UA$ si $(u, r) \in UA \wedge member(u, cs)]$

— $[obj \in objType(cs)$ si $(obj, objType) \in OB \wedge Shared(obj, cs)]$

Cette règle signifie que l'utilisateur u est autorisé à effectuer l'action a sur l'objet obj, si seulement si : (1) il existe une session collaborative cs qui est une instance du template cst ; (2) L'utilisateur u joue le rôle r dans la session cs ; (3) La permission $p(cs) = (a, objType(cs))$ est affectée au rôle dans la session $r(cs)$; (4) l'objet obj est de type $ObjType1$ et partagé dans la session collaborative cs.

Par exemple, nous spécifions la permission de contrôle d'accès liée à la règle : "seul le neurologue membre d'une session collaborative instanciée à partir du template $NeuroEmergency$ est autorisé à accéder à tous les objets (du type MR : dossier médical) partagés dans cette session" ; $cs1$ est une session collaborative telle que :

$$\begin{aligned} & cs1 \in Instance(neuroEmergency) \wedge \\ & (user1, neurologist(cs1)) \in UA \wedge \\ & (neurologist(cs1), (read, MR(CS1)), neuroEmergency) \in PRA \\ & \rightarrow (user1, (read, MR(cs1)) \in UP \\ & (shared(mr1, cs1) \wedge (mr1, MR) \in OB) \\ & \rightarrow (user1, (read, mr1)) \in UP \end{aligned} \tag{6.3}$$

Le modèle MTCS-RBAC fournit une solution intéressante pour prendre en compte le partage des ressources dans une session collaborative se déroulant entre plusieurs tenants. Cependant, ce modèle se limite au contrôle d'accès aux ressources partagées entre les tenants lors d'une session collaborative. Il ne permet pas de supporter les processus collaboratifs composés d'une suite de tâches, telle

que chaque tâche est accomplie par un tenant. Pour cette raison, nous proposons le modèle MTCS-TRBAC pour supporter les workflows collaboratifs.

6.2 Concepts de base du modèle MTCS-TRBAC

Dans cette section, nous présentons le modèle MTCS-TRBAC [37] Multi-Tenant Collaborative Session for TRBAC, est une extension du modèle TRBAC. Ce modèle MTCS-TRBAC est similaire au modèle précédent MTCS-RBAC, la seule différence est que le modèle MTCS-TRBAC prenne en compte le concept du « task » et « workflow » dans un environnement collaboratif Multi-Tenant. Ceci est dans le but de supporter les processus collaboratifs composés d'une suite de tâches et s'exécutant sur un système multi-tenant.

Ce modèle doit permettre l'activation et la désactivation des permissions d'une manière automatique durant la progression et l'exécution des tâches. Ainsi, le modèle prend en considération à la fois le contrôle d'accès actif et passif. Dans ce qui suit, nous rappelons de l'étude de cas d'une application collaborative distribuée décrit précédemment en intégrant un scénario d'un workflow composé d'un ensemble des tâches. Ceci est dans le but de montrer l'utilité du modèle proposé MTCS-TRBAC pour les collaboration définies sous forme d'un workflow des tâches.

6.2.1 Exemple d'un processus collaboratif

Dans cette section, nous rappelons de l'étude de cas décrit précédemment, d'une application collaborative de télé-médecine, pour le diagnostic dans le domaine de la neurologie. Dans cette étude de cas, les trois entités CHU de rabat, $SAMU$ et HA de Khemisset collaborent en partageant un cloud privé commun. Ce dernier fournit un service de stockage aux différentes entités pour stocker leurs données et leurs objets. Chaque entité utilise une instance de ce service de stockage. Chaque instance est dédiée à une entité représentant un locataire (tenant).

Dans cet exemple, nous supposons que chaque client a son propre administrateur qui pourrait définir ses propres politiques de sécurité indépendamment des autres collaborateurs clients. Cet administrateur pourrait effectuer de nombreuses opérations d'administration, telles que :

— Ajouter un nouveau tenant ;

— Supprimer un tenant ;

— Ajouter / supprimer des utilisateurs au / du tenant ;

— Créer des rôles, des templates de sessions collaboratives ;

— Définir des workflows et des tâches ;

— Affecter des utilisateurs aux rôles ;

— Affecter des tâches aux rôles ;

— Affecter les autorisations aux tâches ;

— Affecter les objets aux types d'objets

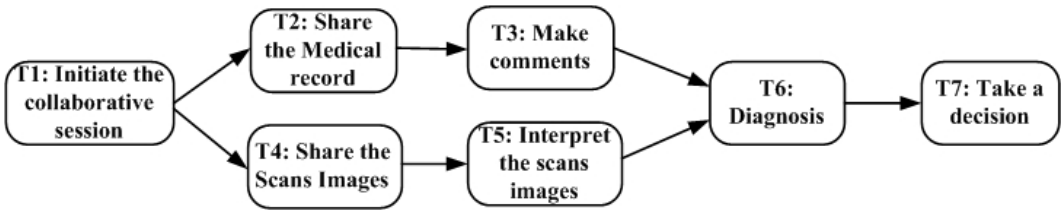

FIGURE 6.4: Workflow Schéma pour le diagnostic en neurologie

L'exemple illustré dans la figure 6.4 montre le workflow schéma pour le diagnostic en neurologie d'un patient en situation d'urgence. Ce workflow est défini comme étant une suite de tâches à accomplir afin de prendre la décision sur le type de soins que va suivre le patient. Dans ce cas d'utilisation, nous considérons que chaque tâche de la session collaborative sera activée uniquement si les tâches précédentes sont accomplies. Le tableau des tâches (6.1) décrit l'affectation des tâches aux rôles dans la session collaborative. Cette affectation est définie par l'administrateur du tenant. De plus, ce dernier associe un ensemble des permissions à chaque tâche. Par exemple, les permissions $(Read, scan\_imag$ et $(write, scan\_imag)$ sont affectées à la tâche « Ta6 : Interpréter les images scans ». Le scénario décrit implique différentes règles de contrôle d'accès inter-tenants. Parmi ces règles, nous trouvons :

— Seul l'utilisateur jouant le rôle $Docteur\_samu$ a l'autorisation de créer une session collaborative et d'inviter les utilisateurs à la rejoindre ;

— Seul l'utilisateur jouant le rôle $Docteur\_ha$ a le droit de partager le dossier médical du patient dans la session collaborative ;

Rôle	***Tâche affectée***
Docteur_samu	*Ta1 : Initier une session collaborative*
Docteur_ha	*Ta2 : Faire des observations*
Docteur_ha	*Ta3 : Partage du dossier médical*
Neurologue CHU	*Ta4 : Déterminer le type des images de scan à effectuer*
Docteur_ha	*Ta5 : Partage les images de scan*
Radiologue CHU	*Ta6 : Interpréter les images de scan*
Neurologue CHU	*Ta7 : Prise de décision*

TABLE 6.1: Tableau d'affectations des tâches

— Tous les membres de la session a le droit d'accéder au dossier médical partagé afin de faire le diagnostic médical ;

— Les non-membres de la session collaborative $CS1$ n'ont pas le droit de participer à cette collaboration ;

— Le radiologue de la session collaborative a le droit d'interpréter les images de scanner partagées dans cette session ;

— Seul le neurologue membre de cette session de collaboration a l'autorisation de décider sur le type de soins que va suivre le patient ;

— Les autorisations associées à chaque tâche ne seront activées que si les tâches précédentes sont accomplies.

Dans ce qui suit, nous présentons le modèle MTCS-TRBAC qui va nous permettre de spécifier formellement ce genre de règles.

6.2.2 Entités de base du modèle MTCS-TRBAC

Dans cette section, nous présentons le concept de base du modèle MTCS-TRBAC (Multi Tenant Collaborative Session for TRBAC) [37] est un modèle pour contrôler l'accès aux ressources partagées lors d'une session collaborative se déroulant sur un environnement multi-tenant. Il est basé sur le modèle de contrôle d'accès TRBAC (Task Role Based Access Control) en intégrant des entités à savoir le tenant, la session collaborative, la Template de la session collaborative et le type objet.

Le choix du modèle TRBAC pour définir les politiques de contrôle d'accès de chaque tenant est motivé par sa capacité d'abstraction qui facilite la spécification des règles d'accès adaptées pour les systèmes de workflow. L'objectif principal du

modèle MTCS-TRBAC est d'étendre le modèle TRBAC pour qu'il prenne en compte les aspects de la "multi-tenancy" et de la session collaborative. Dans ce sens, nous utilisons les deux relations de trust $TrustRole()$ et $TrustShare()$ définies dans le modèle MTCS-RBAC afin d'établir la confiance entre les différents tenants participant à la collaboration.

Dans le modèle MTCS-TRBAC(voir la figure 6.5), nous avons plusieurs entités : clients (I), tenants (T), sessions collaboratives (CS), temples de sessions collaboratives (CST), rôles (R), permissions (P), utilisateurs (U), sessions (S), actions (C), objets (O), types d'objets (B), tâches (TA), instances de tâche (TI), workflows (W) et instances de workflow (WI). De même comme dans MTCS-RBAC, un client a un ou plusieurs tenants. Un tenant est associé à un seul client par une relation un-à-plusieurs. En outre, il existe des relations un-à-plusieurs entre les différentes entités et leur tenant propriétaire. Dans ce qui suit, nous allons décrire les nouvelles entités du modèle les tâches, les instances de tâche, les workflows et les instances de workflow. De plus, nous allons redéfinir l'entité template de session collaborative dans ce modèle. Tandis que les autres entités sont décrites précédemment dans le modèle MTCS-RBAC (la section 6.2).

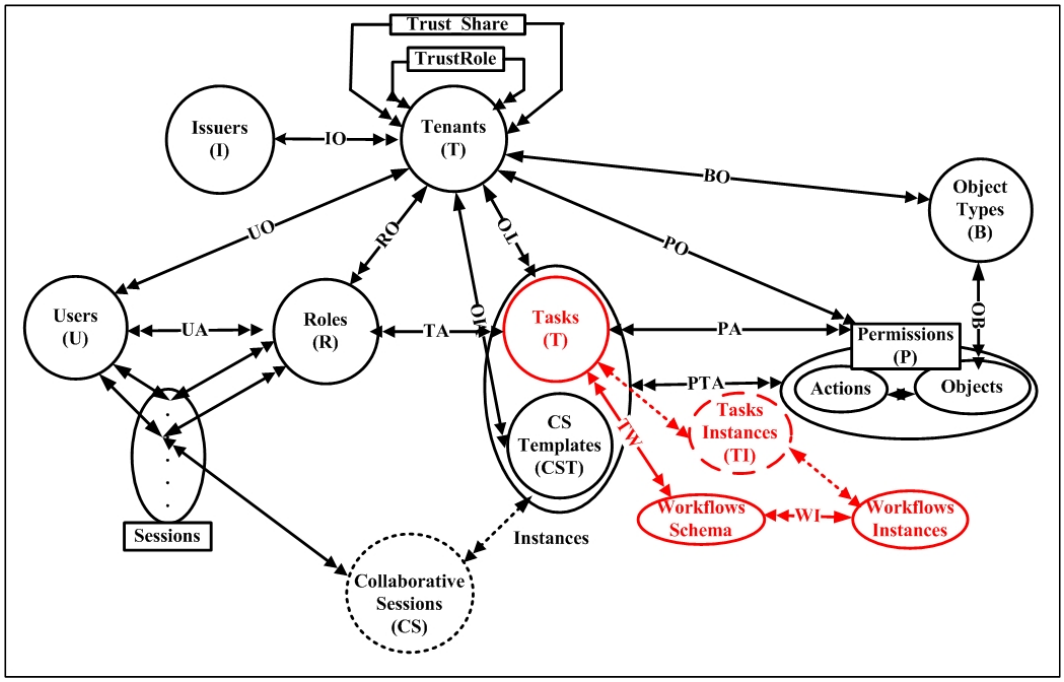

FIGURE 6.5: Modèle MTCS-TRBAC

a. Tâches

La tâche est l'unité fondamentale du travail ou de l'activité métier. « Faire des observations», « partage du dossier médical » et « Interpréter les images de scan » sont des exemples de tâches. Les tâches sont attribuées aux utilisateurs en fonction de leur poste ou de leur rôle dans l'organisation. D'un point de vue contrôle d'accès, un utilisateur n'est autorisée d'accéder à des ressources qu'au moment d'exécution des tâches qui lui sont attribuées. Les droits d'accès sont activés uniquement pendant l'exécution des tâches affectées. La relation entre le tenant et une tâche est définie comme suit : $TAOwner(ta : TA) \rightarrow T$, est une fonction mappant la tâche ta à son propriétaire tenant. Les tâches sont classées en deux catégories :

— La 1ère classe : la tâche qui n'appartient à aucun workflow ($ta \notin WFS$) ;

— La 2ème classe : la tâche appartenant à un workflow donné ($ta \in WFS$) ;

b. Instances de tâche

Une instance de tâche est une instanciation d'une tâche donnée qui est définie par le système. Par exemple «$Ta1$: $initierlasessioncollaborativecs1$ » est une instance de la tâche «$Ta1$: $Initierunesessioncollaborative$ ». La relation entre la tâche et l'instance de tâche est définie comme suit : $taskInstance(ta : TA) \rightarrow 2^{TI}$, est une fonction mappant chaque tâche à un ensemble d'instances de tâche.

c. Workflows

Un workflow est défini comme étant une suite de tâches ou opérations effectuées par une personne, un groupe de personnes ou un organisme dans le but d'atteindre un objectif commun. « Le diagnostic dans le neuroloigie » est un exemple d'un workflow schéma défini dans le scénario de l'application de télémédecine). La relation entre le tenant et le workflow est définie comme suit : $WOwner(w : W) \rightarrow T$, est une fonction mappant un workflow donné à son tenant créateur.

d. Instances de workflow

Une instance de workflow est une instanciation d'un workflow donné modélisé par le concepteur du système. Par exemple, l'instance du workflow s'exécutant lors de la session collaborative $cs1$, est une instanciation du workflow schéma « Diagnostic en neurologie ». La relation entre le workflow et l'instance de workflow est exprimée comme suit : $workflowInstance(w : W) \rightarrow 2^{WI}$, est une fonction mappant chaque workflow à un ensemble d'instances de workflow.

e. Template de la session collaborative

Un template de session collaborative représente une activité de collaboration [4]. Une session collaborative est créée et instanciée à partir d'un template donnée. Un template de session collaborative appartient à un seul tenant, tandis qu'un tenant peut définir plusieurs templates. Un template de session collaborative est défini par le 5-uplet (cst, R, B, TA, w)

- cst : Identifiant du template ;
- R : sous ensemble de rôles liés au template ;
- B : sous ensemble de types d'objets relatifs au template ;
- TA : ensemble de tâches effectuées dans la session ;
- w : un workflow schéma relatif à la session.

6.2.3 Modèle MTCS-TRBAC

Le modèle MTCS-TRBAC comporte les composants suivants :

- I , T , U , R, P , S, CS, M, TA, TI, WS, WI, O, B and TT sont un ensemble fini des issuers, tenants, utilisateurs, rôles, permissions, sessions, sessions collaboratives, templates de sessions collaboratives, tâches, instances de tâche, workflows, instances de workflow, objets, types d'objets et les relations de confiance entre les tenants respectivement ;
- $TO \subseteq T \times I$, est une relation un-à-plusieurs mappant chaque tenant à son issuer propriétaire, $tenantOwner(t : T) \rightarrow I$, est une fonction mappant un tenant donné à son propriétaire telle que $issuerOwner(t) = i \; si \; (t, i) \in TO$;
- $MO \subseteq M \times T$, est une relation un-à-plusieurs mappant chaque template de session collaborative à son tenant créateur, $CSTOwner(m : M) \rightarrow T$, est une fonction de mappage chaque template de session collaborative à son créateur t telle que $CSTOwner(m) = t \; si \; (m, t) \in MO$;
- $UO \subseteq U \times T$, une relation un-à-plusieurs mappant chaque utilisateur à son tenant propriétaire, $userOwner(u : U) \rightarrow T$, est une fonction de mappage d'un utilisateur u à son propriétaire t telle que $userOwner(u) = t \; iff \; (u, t) \in UO$;
- $RO \subseteq R \times T$, une relation un-à-plusieurs mappant chaque rôle à son tenant propriétaire, $ROwner(r : R) \rightarrow T$, est une fonction de mappage d'un rôle r à son propriétaire t telle que $roleOwner(r) = t \; si \; (r, t) \in RO$;

— $TAO \subseteq TA \times T$, une relation un-à-plusieurs mappant chaque tâche à son tenant, $TOwner(ta : TA) \rightarrow T$, est une fonction de mappage d'une tâche ta à son propriétaire t telle que $TAOwner(ta) = t \; si \; (ta, t) \in TAO$;

— $WSO \subseteq WS \times T$, une relation un-à-plusieurs mappant chaque schéma de workflow à son tenant créateur, $WOwner(ws : WS) \rightarrow T$, une fonction de mappage d'un schéma de workflow ws au tenant qui le défini t telle que $WorkflowOwner(ws) = t \; si \; (ws, t) \in WSO$;

— $PO \subseteq P \times T$, une relation un-à-plusieurs mappant chaque permission à son tenant créateur, $POwner(p : P) \rightarrow T$, est une fonction mappant une permission p au tenant qui la définit t telle que $permOwner(p) = t \; si \; (p, t) \in PO$;

— $OT \subseteq O \times T$, une relation un-à-plusieurs mappant chaque objet à son propriétaire, $OOwner(o : O) \rightarrow T$, est une fonction de mappage d'un objet o à son tenant propriétaire t telle que $ObjectOwner(o) = t \; si \; (o, t) \in OO$;

— $BO \subseteq B \times T$, une relation un-à-plusieurs mappant chaque type d'objet à son tenant créateur, $OTOwner(b : B) \rightarrow T$, une fonction de mappage d'un type d'objet b à son créateur t telle que $OTOwner(b) = t \; si \; (b, t) \in BO$;

— $OB \subseteq O \times B$ relation un-à-plusieurs mappant chaque objet à son type, $objectType(o : O) \rightarrow B$, est une fonction de mappage d'un objet o à son type b telle que $objectType(o : O) = b$ si $(o, b) \in OB$;

— $Instances(cst : CST) \rightarrow 2^{CS}$, est une relation un-à-plusieurs mappant chaque template de session collaborative à ses instances de sessions collaboratives. Par exemple, la relation $Instance(NeuroEmergency) = \{cs1, cs2\}$ signifie que les sessions collaboratives $cs1$ et $cs2$ sont des instances du template $neuroEmergency$;

— $TaskInstances(ta : TA) \rightarrow 2^{TI}$, est une relation un-à-plusieurs mappant chaque tâche à un ensemble des instances de tâche. Par exemple, la fonction $TaskInstances(observation) = \{obs(cs1), obs(cs2)\}$ signifie que les instances de tâche obs(cs1) et obs(cs2) sont instanciées à partir de la tâche « $Ta2 : Faire\,des\,observations$ » ;

— $WorkflowInstances(ws : WS) \rightarrow 2^{wi}$, est une relation un-à-plusieurs mappant chaque schéma workflow à un ensemble des instances de workflow. Par exemple, la relation $workflowInstances(neuroDiagnostic) =$

$\{neuroDiagnostic(cs1), neuroDiagnostic(cs2)\}$, signifie que les instances de workflow $neuroDiagnostic(cs1)$ et $neuroDiagnostic(cs2)$ sont instanciées à partir du schéma de workflow « *diagnosticenneurologie* » ;

— $UA \subseteq U \times R$, une relation plusieurs-à-plusieurs mappant les utilisateurs à des rôles, telle que $(u, r) \in UA$ si seulement si $userOwner(u) = roleOwner(r)$;

— $RTA \subseteq R \times TA$, est une relation plusieurs-à-plusieurs mappant chaque rôle à un ensemble de tâches qui lui sont associées, telle que $(r, ta) \in RTA$ si seulement si $ROwner(r) = TAOwner(ta)$;

— $PTA \subseteq P \times TA$, est une relation plusieurs-à-plusieurs mappant chaque tâche à un ensemble des permissions qui lui sont attribuées, telle que $(p, ta) \in PTA$ si seulement si $POwner(p) = TAOwner(ta)$;

— $TAWS \subseteq TA \times WS$, est une relation plusieurs-à-plusieurs mappant chaque tâche à un schéma de workflow auquel elle appartient, telle que $(ta, ws) \in TAWS$ si seulement si $TAOwner(ta) = WOwner(ws)$;

— $Users(cs : C) \rightarrow 2^U$, est une fonction de mappage de chaque session collaborative cs à un sous ensemble des utilisateurs qui sont membres de cette session, $Users(cs) = \cup_{u \in U}\{u | Member(u, cs)\}$;

— $Sessions(cs : C) \rightarrow 2^S$, est une fonction de mappage de chaque session collaborative cs à un sous ensemble de sessions individuelles :
$Sessions(cs) = \cup_{s \in S}\{s | Member(user(s), cs)\}$;

— Rôle dans la session $r(cs)$: est le rôle qui sera activé pour un utilisateur, lorsqu'il rejoint une session collaborative cs. Dans la même session, nous pouvons avoir un ensemble de membres qui jouent les mêmes rôles ou de différents rôles : $(u, r(cs)) \in UA$ si $(u, r) \in UA \wedge member(u, cs)$

— Type d'objets dans la session $ObjType(cs)$: sont les objets de type $objType$ qui sont partagés dans la session collaborative cs. $obj \in objType(cs)$ si seulement si : $(obj, objType) \in OB \wedge shared(obj, cs)$;

— Permission dans la session $p(cs)$: est une permission relative à l'accès à un objet partagé dans une session collaborative donnée cs.
$p(cs) = (a, objType(cs))$;

— $PTAM \subseteq P \times TA \times M$, est une relation plusieurs-à-plusieurs-à-plusieurs entre la permission, la tâche et le template, telle que $(p(cs), ta(cs), m) \in$

$PTAM$ seulement si :
$POwner(p) = TAOwner(ta) = CSTOwner(m) \wedge cs \in instances(m)$;

— $\forall\, tij \in TI$ est une instance de la tâche taj telle que, $tij \in taskInstances(taj)$, tij est active, on note $active(tij)$ si seulement si :

  — $taj \notin WFS$ or ;

  — $taj \in WFS \wedge (ta1, ta2, ..., tan \rightarrow taj) \wedge Completed(ti1, ti2, ..., tin)$.

$Completed(ti1, ti2, ..., tin)$, signifie que les instances de tâche qui précédent l'instance tij sont complètement accomplies ;

— $TrustRole(tr, te : T) \rightarrow 2^R$, est une nouvelle relation de confiance établie entre les tenants, mappant le truster et le trustee à un sous ensemble des rôles définis par le truster ;

— $TrustShare(tr, te : T) \rightarrow 2^P$, est une nouvelle relation de confiance établie entre les tenants, mappant le couple (truster, trustee) à un ensemble permissions dans la session (relatives aux ressources partagées dans la session).

Les règles de contrôle d'accès dans le modèle MTCS-TRBAC sont formellement exprimées comme suit : $u \in U, p \in P, r \in R, cs \in CS, cst \in CST, ta \in TA, obj \in O, objType \in B$ telles que :

$$\begin{array}{l} si\, \exists cs \in Instance(cst) \wedge \\ (u, r(cs)) \in UA \wedge \\ (r, ta) \in RTA \wedge \\ (p(cs), ta(cs), cst) \in PTAM \wedge \\ ta(cs) \in taskInstances(ta) \wedge \\ Active(ta(cs)) \rightarrow (u, p(cs)) \in UP \\ (p(cs) = (a, objType(cs))) \\ \rightarrow (u, (a, obj)) \in UP \end{array} \tag{6.4}$$

  — $[(u, r(cs)) \in UA$ si $(u, r) \in UA \wedge member(u, cs)]$

  — $[obj \in objType(cs)$ si $obj, objType) \in OB \wedge Shared(obj, cs)]$

Cette règle signifie que l'utilisateur u est autorisé à exécuter l'action a sur l'objet obj, si seulement si : (1) il existe une session collaborative cs qui est une instance du template cst ; (2) L'utilisateur u joue le rôle r dans la session cs ; (3) La tâche ta est affectée au rôle r ; (4) La permission $p(cs) = (a, objType(cs))$ est affectée à l'instance de tâche $ta(cs)$ pour le

template de session collaborative cst ; (5) $ta(cs)$ est une instance de la tâche ta ; (6) $ta(cs)$ est une tâche active ; (7) l'objet obj est de type $ObjType$ et partagé dans la session collaborative cs.

A titre d'exemple, nous spécifions la permission de contrôle d'accès liée à la règle : le radiologue membre d'une session collaborative instanciée à partir du Template $NeuroEmergency$ est autorisé à accéder à tous les objets (du type $scan$) partagés dans cette session collaborative $cs1$, uniquement pendant l'exécution de la tâche « $Ta6 : Interprter\ les\ images\ de\ scan$ » ;

$$\begin{aligned}
&cs1 \in Instance(neuroEmergency) \wedge \\
&(user1, radiologist(cs1)) \in UA \wedge \\
&(radiologist, ta6) \in RTA \wedge \\
&((read, scan(cs1)), ta6(cs1), neuroEmergency) \in PTAM \wedge \\
&ta6(cs) \in taskInstances(ta6) \wedge \\
&Active(ta6(cs)) \\
&\rightarrow (user1, (read, scan(cs1)) \in UP \\
&(shared(scan1, cs1) \wedge (scan1, scan) \in OB) \\
&\rightarrow (user1, (read, scan1)) \in UP
\end{aligned} \tag{6.5}$$

— *Active(ta6(cs1)) si :*
Completed(ta1(cs1), ta2(cs1), ta3(cs1), ta4(cs1), ta5(cs1)

6.2.4 Modèle d'administration

Dans cette section, nous proposons le modèle administratif de notre modèle MTCS-TRBAC. Ce modèle permet aux administrateurs d'effectuer certaines opérations administratives. Chaque opération administrative nécessite certaines conditions préalables. Dans ce qui suit, pour un seul client cloud (Issuer), nous proposons la spécification formelle de chaque opération administrative, en mentionnant ses conditions préalables :
$\forall i \in I$, $\forall t \in T$, $\forall u \in U$, $\forall r \in R$, $\forall ta \in TA$, $\forall p \in P$, $\forall cs \in CS$ et $\forall m \in M$

— AssignUser(t, u, r)
Precondition : $(t, i) \in TO \wedge (u, t) \in UO \wedge [(r, t) \in RO \vee \exists tx \in T,\ r \in TrustRole(tx, t),\ where(r, tx) \in RO] \wedge (u, r) \notin UA$

— RevokeUser(t, u, r)

Precondition : $(t,i) \in TO \wedge (u,t) \in UO \wedge (r,t) \in RO \vee \exists tx \in T,\ r \in TrustRole(tx,t),\ where(r,tx) \in RO] \wedge (u,r) \in UA$

— AssignRoleTask(t, r, ta)
Precondition : $(t,i) \in TO \wedge (ta,t) \in TAO \wedge [(r,t) \in RO \vee \exists tx \in T,\ r \in TrustRole(tx,t),\ where(r,tx) \in RO] \wedge (r,ta) \notin RTA$

— revokeRoleTask(t, r, ta)
Precondition : $(t,i) \in TO \wedge (ta,t) \in TAO \wedge [(r,t) \in RO \vee \exists tx \in T,\ \ r \in TrustRole(tx,t),\ where(r,tx) \in RO] \wedge (r,ta) \in RTA$

— AssignPerm(t, ta, p)
Precondition : $(t,i) \in TO \wedge (ta,t) \in TAO \wedge (p,t) \in PO \wedge (p,ta) \notin PTA$

— revokePerm(t, ta, p)
Precondition : $(t,i) \in TO \wedge (ta,t) \in TAO \wedge (p,t) \in PO \wedge (p,ta) \in PTA$

— AssignPermCS(t, p(cs), ta(cs), m)
Precondition : $(t,i) \in TO \wedge (m,t) \in MO \wedge \forall cs \in Instances(m) \wedge (ta,t) \in TAO \wedge ta(cs) \in taskInstances(ta) \wedge [(p(cs),t) \in PO \vee \exists ty \in T, p(cs) \in TrustShare(ty,t)] \wedge (p(cs),ta(cs),m) \notin PTAM$

— RevokePermCS(t, p(cs), ta(cs), m)
Precondition : $(t,i) \in TO \wedge (m,t) \in MO \wedge \forall cs \in Instances(m) \wedge (ta,t) \in TAO \wedge ta(cs) \in taskInstances(ta) \wedge [(p(cs),t) \in PO \vee \exists ty \in T \wedge P(cs) \in TrustShare(ty,t)] \wedge (p(cs),ta(cs),m) \in PTAM$

— AssignObject(t, obj, objType)
Precondition : $(t,i) \in TO \wedge (obj,t) \in OO \wedge (obTypej,t) \in BO$

— addTenant(t)
Precondition : $i \in I \wedge t \notin T$

— deleteTenant(t)
Precondition : $(t,i) \in TO \wedge t \in T$

— addCSTemplate(t,m)
Precondition : $(t,i) \in TO \wedge m \notin M$

6.3 Implémentation

Dans cette section, nous allons décrire l'implémentation de notre approche sur le cloud open source Openstack [2]. Ce cloud comme il a été décrit dans le

premier chapitre, comporte un ensemble des composants open source permettant de déployer des infrastructures de cloud computing (infrastructure as a service : IaaS). Openstack possède une architecture modulaire composée de plusieurs projets corrélés (Nova, Swift, Glance, Keystone...) qui permettent de fournir et gérer les différents services tels que les machines virtuelles, la puissance de calcul, le stockage ou encore le réseau.

Dans notre implémentation, nous nous intéressons plus particulièrement à la composante de stockage d'Openstack Swift. Cette composante est un système de stockage de fichiers, vidéos, données, données analytiques, contenus web, sauvegardes, images, instances de machines virtuelles et autres données non structurées. Swift utilise pour les sauvegardes plusieurs serveurs. Si un serveur ou un disque dur tombe en panne, Swift réplique ses données depuis des nœuds actifs du cluster dans de nouveaux emplacements.

Dans notre scénario, nous considérons que nous avons trois tenants collaborateurs : le tenant « centre hospitalier universitaire (CHU) », le tenant « service d'aide médicale urgente (SAMU) » et le tenant « hôpital d'accueil (HA) ». Nous implémentons les interactions entre ces tenants sur la composante de stockage Swift dans la plate-forme de cloud computing open source OpenStack. Nous considérons que chaque tenant correspond à un compte de swift ($ACCOUNT$) (par exemple, les comptes ACC_CHU, ACC_SAMU et ACC_HA représentent respectivement les tenants CHU, $SAMU$ et HA).

Un conteneur (Container) est un mécanisme qui stocke des objets de données. Un compte peut avoir plusieurs conteneurs, tandis qu'un conteneur doit être associé à un seul compte. (voir 6.6). Un utilisateur est l'entité qui peut effectuer des actions sur l'objet dans le compte. Chaque utilisateur est détenu par un seul tenant (propriétaire). Le compte ACC_CHU comporte quatre utilisateurs : ACC_user1, ACC_user2, ACC_user3 et ACC_user4. Dans notre exemple, chaque compte comporte plusieurs conteneurs et chaque conteneur contient un ensemble d'objets. Par exemple, le compte ACC_HA comporte quatre conteneurs : MR, DEC, $SCAN$ et $DIAG$. (Voir figure 6.7). Swift utilise le modèle d'accès à base de listes (ACL) pour définir les politiques d'accès des utilisateurs aux objets stockés dans le conteneur. Le modèle ACL permet de définir des autorisations relatives à l'utilisation des objets d'un conteneur d'une manière intuitive et statique. Cependant, ce modèle ne permet pas de supporter un ensemble des

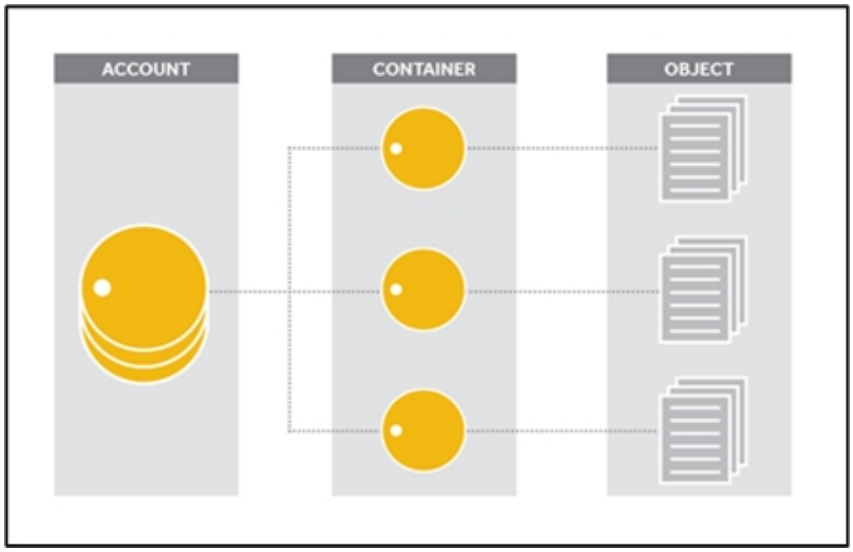

FIGURE 6.6: Architecture de Swift [3]

exigences liées à la collaboration dans les environnements Cloud (citées dans le chapitre précédent) à savoir :

— Les ACL de Swift se basent sur les listes afin de donner l'autorisation à un utilisateur pour accéder à un fichier. Cependant, Swift ne permet pas de spécifier les opérations d'accès et d'administration définies dans notre approche telles que : l'affectation d'un utilisateur à un rôle, l'attribution des droits à un rôle et les relations relatives aux sessions de collaboratives.

— Il est nécessaire d'avoir un composant pour gérer les sessions collaboratives (Créer des sessions collaboratives, Rejoindre / quitter la session, Ajouter des utilisateurs dans des sessions et partager des objets dans les sessions).

— Les relations de confiance entre les tenants, le truster expose certains rôles au qui va affecter ses utilisateurs aux rôles de truster. Donc, dans Swift, il faut avoir un mécanisme permettant de prendre en charge ces relations de confiance.

Afin de surmonter les problèmes liés à l'utilisation des ACL, nous proposons d'ajouter, dans l'environnement Swift, un nouveau module "Policy module" développé en langage shell. Ce module (6.7) est composé de plusieurs composants à savoir : "SHELL engine", "Tenant Trust Relations", "Session managment", etc. Dans ce qui suit, nous décrivons chaque composant de ce module :

— Composant URT (User Role Relations) : Dans ce composant, l'administrate-

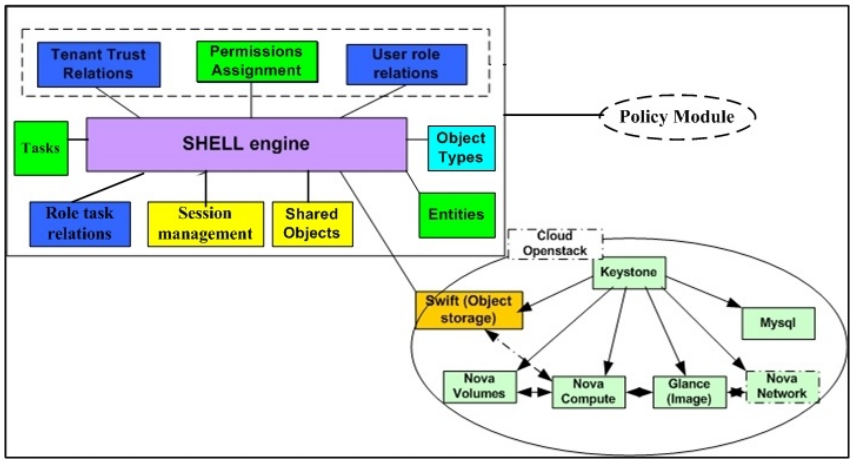

FIGURE 6.7: Implémentation du modèle MTCS-TRBAC [37]

ur de sécurité définit la relation entre le tenant, l'utilisateur et le rôle. Par exemple (ACC_CHU : ACC_user1 : $neurologue$) signifie que dans le tenant ACC_CHU, l'utilisateur ACC_user1 joue le rôle de neurologue.

— Composant PA (Permissions Assignement) : Dans ce composant, l'administrateur spécifie l'attribution des permissions aux rôles. Dans notre scénario, nous considérons que le tenant $SAMU$ est celui qui définit les règles d'accès. Notez qu'à ce niveau, nous supposons que les règles de politique de sécurité sont valides et sans conflit.

— Composant Entités : Dans ce composant, l'administrateur définit toutes les entités du modèle (Template de sessions collaboratives, rôles, utilisateurs, objets et types d'objets). De plus, il définit dans ce composant, les relations reliant chaque entité à propriétaire tenant.

— Composant Types d'objets : Dans ce composant, l'administrateur spécifie toutes les relations reliant les objets à leurs types. Ces relations sont spécifiées comme suit : ($MR : mr1$), ce qui signifie que l'objet $mr1$ est de type MR.

— Composant de membre Sessions : ce composant est responsable de la gestion des utilisateurs membres dans la session, à savoir : rejoindre / quitter une session collaborative. Un membre d'une session collaborative est exprimé dans ce composant comme suit : ($CS1 : ACC\_user1$).

— Relations de confiance entre les tenants : Dans ce composant, l'administrateur définit les relations de confiance établies entre deux tenants. Ces re-

lations sont spécifiées dans ce composant comme suit : (ACC_SAMU : ACC_CHU : $neurologue$), ce qui signifie que le tenant ACC_SAMU fait confiance au tenant ACC_CHU pour que ce dernier utilise le rôle neurologue défini par ACC_SAMU. De même, l'administrateur spécifie la relation de confiance $TrustShare()$ en utilisant le formalisme suivant : ($ACC\_HA : ACC\_SAMU : perm1(cs)$) signifie que le tenant ACC_HA autorise le tenant ACC_SAMU à utiliser la permission $perm1(cs)$.

— Composant d'objets partagés : ce composant est responsable de la gestion des ressources partagées dans la session. Par exemple, ($CS1 : MR1$) signifie que le dossier médical $MR1$ est partagé dans $CS1$.

— Composant des tâches : dans ce composant, l'administrateur définit les entités tâche, instance de tâche, workflow et instance de workflow. De plus, l'administrateur représente chaque schéma de workflow par une suite des tâches. Par exemple, (ACC_CHU : $Take\_decision$: $NeuroDiagnostic$) signifie que dans le tenant ACC_CHU, la tâche $Take\_decision$ appartient au workflow $NeuroDiagnostic$.

Ces composants sont utilisés par le composant "Shell Engine" pour évaluer une requête d'accès d'un utilisateur à une ressource partagée lors d'une session collaborative. Lorsqu'un utilisateur envoie cette requête au cloud swift, le composant "Shell Engine" évalue cette requête en fonction des règles définies dans la politique d'accès afin de décider si l'utilisateur est autorisé à accéder à cette ressource ou non. Si l'utilisateur est autorisé à effectuer cette action, le module "Policy module" exécutera une commande ACL pour attribuer cette autorisation à cet utilisateur dans l'environnement Swift.

6.4 Conclusion

Dans ce chapitre, nous avons défini formellement une nouvelle approche MTCS-TRBAC pour assurer le contrôle d'accès aux ressources partagées lors d'une session collaborative se déroulant dans un environnement multi-tenant. Dans cette approche, nous avons introduit de nouvelles entités et des relations de confiance pour prendre en compte la "multitenancy" et la session collaborative. Ensuite, nous avons étendu cette approche pour qu'elle prenne en compte le concept du « task » et « workflow » dans un environnement collaboratif Multi-Tenant. Enfin,

nous avons démontré la faisabilité de notre approche par une implémentation dans l'environnement SwiftStack.

Chapitre 7

Modèle session collaborative multi-tenant pour ABAC : MTCS-ABAC

Dans le chapitre précédent, nous avons présenté les deux modèles MTCS-RBAC et MTCS-TRBAC pour assurer le contrôle d'accès des ressources échangées entre les tenants, lors d'une session collaborative. Le modèle MTCS-TRBAC est similaire au modèle MTCS-RBAC, la seule différence est que le modèle MTCS-TRBAC prenne en compte le concept du « task et workflow » dans un environnement collaboratif Multi-Tenant. Ceci permet ainsi de supporter les processus collaboratifs.

Dans ce chapitre, nous présentons un nouveau modèle de contrôle d'accès basé sur le modèle ABAC pour assurer la collaboration entre multiples tenants [39]. Ce modèle permet aux tenants participant à la collaboration de définir la politique globale de contrôle d'accès d'une manière collaborative sur un point central tout en gardant l'autonomie et la confidentialité de chaque tenant. De plus cette approche supporte la notion des taches actives et passives. Ce travail a été implémenté et vérifié sur la composante de stockage swiftstack de la plateforme Openstack.

Le modèle ABAC a été choisi dans ce travail en tant que modèle de base pour sa flexibilité et sa capacité d'abstraction permettant de définir de politiques de contrôle d'accès centrées sur les attributs [12]. Dans ce chapitre, tout d'abord, nous commençons par présenter la motivation qui nous a permis de proposer

ce modèle MTCS-ABAC. Ensuite, nous introduisons la notion du tenant collaboratif. Puis, nous décrivons comment ce modèle peut supporter les processus collaboratifs s'exécutant sur un environnement multi-tenant. De plus, nous proposons de nouvelles fonctions permettant aux tenants de garder leur autonomie et leur confidentialité durant la collaboration. Ensuite, nous présentons le modèle formel s'appuyant sur la logique du premier ordre. Enfin, nous décrivons l'architecture d'implémentation.

7.1 Motivation

Durant une session collaborative, les tenants ont besoin d'accéder et d'utiliser des ressources partagées par d'autres tenants. Certaines ressources contiennent souvent des données sensibles. Le contrôle d'accès à ces ressources constitue un défi majeur auquel il faire face. Dans ce sens, les tenants participant à la collaboration doivent adopter un modèle de contrôle d'accès flexible et solide pour prendre en compte les exigences liées à la collaboration entre les tenants. Cette collaboration doit prendre en considération aussi bien que le partage des ressources entre des tenants que les workflows collaboratifs s'exécutant sur un système multi-tenant.

Dans notre modèle, nous nous intéressons plus particulièrement à définir un modèle de contrôle d'accès pour sécuriser la collaboration entre les tenants. En effet, on considère que cette collaboration englobe à la fois le partage des ressources et les workflows collaboratifs. Les modèles de contrôle d'accès pour les environnements collaboratifs multi tenans sont classés en deux catégories : les modèles de contrôle d'accès centralisés et décentralisés (pair à pair).

Dans les modèles centralisés, la politique globale de contrôle d'accès est spécifiée sur un point central. Ceci va permettre de prendre en compte les processus collaboratifs. Tel que chaque processus est composé d'une suite des tâches et chaque tâche sera achevée par un tenant participant à la collaboration. D'un point du vue technique, ces modèles permettent ainsi l'activation et la révocation automatique des permissions en fonction de la progression des tâches.

Dans les approches décentralisées, chaque tenant est responsable de la définition de sa propre politique de contrôle d'accès. Dans ce cas, les tenants sont faiblement couplés. Généralement, l'administrateur de chaque tenant intègre une

partie de la politique globale de contrôle d'accès dans sa politique. En adoptant cette approche, chaque tenant garde le contrôle total de sa politique d'accès et de ses propres ressources.

Dans ce travail, nous proposons le modèle de contrôle d'accès MTCS-ABAC (Multi Tenants Collaborative Session for ABAC) [39] une extension du modèle ABAC pour définir des politiques de contrôle d'accès plus complexes et fines. Ce modèle est spécialement adapté pour les processus collaboratifs s'exécutant sur un environnement multi-tenant. Le modèle MTCS-ABAC s'appuie sur une approche centralisée qui permet au collaborateurs (tenants) de spécifier leur politique globale d'autorisations, de manière collaborative sur un point central.

MTCS-ABAC surmonte les limitations des modèles multi-tenants de contrôle d'accès en prenant en compte les exigences suivantes (décrites dans le chapitre état de l'art) : L'agilité de la session collaborative ; le partage des ressources dans la session entre plusieurs tenants ; les processus collaboratifs ; la flexibilité ; l'autonomie et la confidentialité. Dans ce qui suit, nous allons décrire en détail le modèle MTCS-ABAC.

7.2 Concepts de base du modèle MTCS-ABAC

Dans cette section, nous présentons les concepts de base du modèle MTCS-ABAC (Multi Tenants Collaborative Session for ABAC) [39] : un nouveau modèle basé sur ABAC pour supporter la collaboration dans des environnements multiples tenants. Le modèle ABAC a été défini dans la littérature de différentes manières, généralement pour répondre à des besoins spécifiques. Dans notre approche, nous nous appuyons sur le modèle ABAC0 qui a été défini dans [45]. Le modèle ABAC0 (voir la figure 7.1) qui comporte les deux entités utilisateur U et objet O. Chaque entité est associée à un ensemble des attributs. Dans ce sens, une autorisation pour une action A dans modèle ABAC0, est définie comme étant une combinaison des attributs utilisateurs et objets. Dans cette section, tout d'abord, nous commençons par la définition du concept "tenant collaboratif". Ensuite, nous décrivons le processus métier de la collaboration. Après cela, nous présentons la définition formelle du modèle MTCS-ABAC. Enfin, nous terminons cette section par un exemple illustrant l'utilisation du modèle MTCS-ABAC pour le scénario de l'application collaborative de télémédecine décrit précédemment.

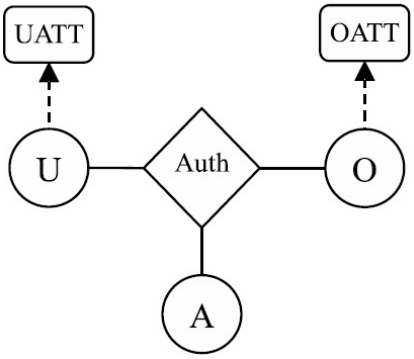

FIGURE 7.1: Modèle ABAC0

7.2.1 Tenant collaboratif

Le tenant collaboratif est le tenant responsable d'assurer la collaboration entre les différents tenants. Il fournit la collaboration en tant que service (Collaboration as a service) pour les clients du cloud. Ce tenant collaboratif permet à un groupe d'utilisateurs de différents tenants de collaborer via des plateformes distribuées afin d'accomplir un processus commun. En faite, si l'on considère qu'un ensemble de tenants envisagent de collaborer. Tout d'abord, ils doivent créer ce tenant collaboratif. Ensuite, ils définissent dans ce tenant collaboratif le processus de collaboration qui est défini sous forme d'un workflow composé d'un ensemble de tâches dans lequel chaque tâche sera achevée par un tenant donné, tandis qu'un tenant peut accomplir une ou plusieurs tâches. Par exemple, (Figure 7.2), les tenants CHU, $SAMU$ et HA sont trois tenants collaborent à travers le tenant collaboratif $CTE1$ qui leur fournit un service de collaboration.

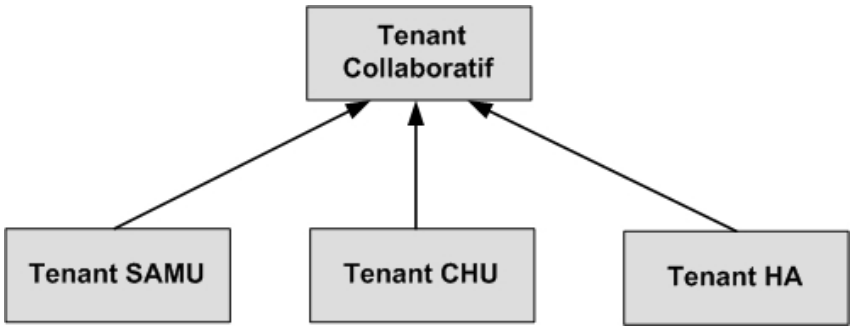

FIGURE 7.2: Tenant collaboratif

7.2.2 Processus métier de collaboration

Dans cette approche, on considère que chaque collaboration est définie sous forme d'un processus composé d'une suite de tâches connectées pour atteindre un objectif commun. Par exemple, la figure 7.3 présente le processus métier de diagnostic lié au scénario télé-médecine décrit précédemment. Dans cette collaboration, chaque tâche sera accomplie par un tenant donné en utilisant des ressources fournies par d'autres tenants. Le tableau 7.1 montre l'affectation des tenants aux tâches du processus métier (diagnostic) ainsi que la détermination de type des ressources nécessaires pour accomplir chaque tâche. Par exemple, le CHU est le tenant responsable d'exécuter la tâche $T7$. De plus, pour accomplir cette tâche, le tenant HA doit donner au tenant CHU l'accès aux ressources le dossier médical MR, l'image de scanner $Scan$ et le vidéo du patient concerné.

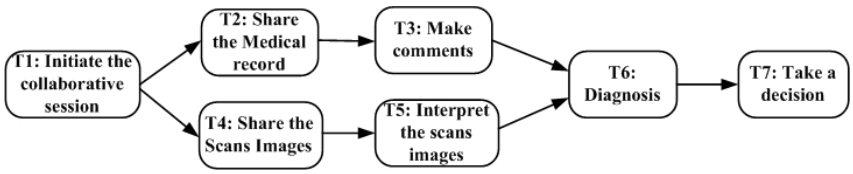

FIGURE 7.3: Workflow de collaboration

Dans ce travail, nous étendions le modèle ABAC0 par l'ajout de l'entité "Tâche" aux autres entités du modèle. La tâche est une unité fondamentale du travail ou de l'activité métier. Les tâches sont attribuées aux tenants en fonction de leur rôle dans la collaboration. La tâche est définie par le triplet (Libellé de la tâche, tenant responsable de la tâche, ensemble de ressources (appartenant à d'autres tenants) utilisés pour l'accomplissement de la tâche). Par exemple dans le tableau (7.1), pour que le tenant CHU accomplisse la tâche $T7$, il a besoin d'accéder aux ressources fournies par le tenant HA.

7.2.3 Démarche du modèle

Afin de spécifier l'autorisation MTCS-ABAC tout en prenant en compte les exigences des workflows collaboratifs dans des environnements multi-tenants, nous devons utiliser la démarche illustrée dans le tableau (7.2). Le modèle MTCS-ABAC introduit l'entité tâche aux autres entités utilisateur et objet du modèle ABAC. Dans ce modèle, chaque tâche est définie par un ensemble d'attributs de

Tâches	***Utilisateurs affectés***	***Objets utilisés***
T5 : Interpret the scans images	*Tenant CHU*	*Scan : Tenant HA*
T7 : Take a decision	*Tenant CHU*	*MR : Tenant HA* *Scan : Tenant HA* *Video : Tenant HA*

TABLE 7.1: Affectations des tâches aux tenants

tâches tels que : le nom de la tâche, le workflow de la tâche, les tâches précédentes et la session collaborative dans laquelle la tâche s'exécute. En outre, le tenant chargé d'accomplir une tâche peut avoir une ou plusieurs autorisations relatives à cette tâche. Par exemple, pour « T7 : Take-a-decision », l'utilisateur du tenant *CHU* chargé de cette tâche nécessite avoir droit de quatre autorisations : (1) lire le dossier médical $mr1$ du patient ; (2) lire l'image de scan $Scan1$ du patient ; (3) accéder au fichier vidéo du patient ; (4) écrire la décision finale sur le type de soins que va suivre le patient. Dans le modèle MTCS-ABAC (Voir

Tâches	***Utilisateurs affectés***	***Objets utilisés***	***Actions***
$TATT1(ti) = Vali$ $TATT2(ti) = Vali$ ..... $TATTn(ti) = Vali$	$UATT1(u) = valj$ $UATT2(u) = valj$ ..... $\mathbf{AssignedUser_{(ta,te)}(u) = True}$	$OATT1(o) = valk$ $OATT2(o) = valk$ ..... $\mathbf{UsedObject_{(OT,a,te)}(o) = True}$	*Read/ Write*

TABLE 7.2: Autorisations du modèle MTCS-ABAC

le tableau 7.2), chaque autorisation liée à une tâche donnée, est définie par : (1) un ensemble d'attributs de tâche (liés à cette tâche) ; (2) un ensemble d'attributs utilisateur représentant l'utilisateur chargé de cette tâche ; (3) un ensemble d'attributs d'objet représentant la ressource utilisée pour accomplir cette tâche ; (4) Une action qui est une opération spécifique sur un objet. En faite, l'utilisateur responsable de chaque tâche obtient de nombreuses autorisations pour l'accomplir. Pour cela, chaque tâche est donc affectée à une ou plusieurs autorisations (autorisation ABAC).

Les autorisations MTCS-ABAC sont définies par la combinaison des attributs

de tâche, d'utilisateur et d'objet. En effet, chaque attribut de tâche aura la même valeur pour toutes les autorisations MTCS-ABAC liées à cette tâche. De même, chaque attribut utilisateur a la même valeur pour toutes les autorisations liées à une tâche vu que l'on considère que l'utilisateur chargé d'accomplir une tâche donnée, est celui qui va effectuer toutes les opérations attribuées à cette tâche.

Le modèle MTCS-ABAC est composé des trois entités de base : utilisateurs (U), objets (O) et tâches (T). Les actions sont des opérations autorisées dans le système. Ces opérations incluent généralement créer, lire, modifier et supprimer. Une action est exécutée par un utilisateur sur un objet. D'un autre coté, dans ce modèle, chaque utilisateur appartient un propriétaire tenant unique. Pour cela, on définit l'attribut $UOwner$ qui est une fonction un-ou-plusieurs entre les utilisateurs U et leur propriétaire tenant TE.

De même, chaque tenant possède un ensemble des objets. De la même manière, on définit l'attribut $OOwner$ qui est une fonction un-ou-plusieurs entre les objets O et leur propriétaire tenant TE. De plus, nous définissons l'attribut $TOwner$ qui est une fonction un-ou-plusieurs entre les tâches T et leur créateur tenant TE. De plus, chaque attribut d'utilisateur, chaque attribut d'objet et chaque attribut de tâche appartiennent également à un tenant unique. Ceci est représenté respectivement par la définition des attributs $UAOwner$, $OAOwner$ et $TAOwner$ dont les valeurs sont atomiques.

Le principe essentiel du modèle MTCS-ABAC est que chaque tenant est le responsable de l'attribution de valeurs d'attributs à ses propres entités (utilisateurs, objets et tâches). Vu l'isolation entre les tenants, l'administrateur d'un tenant donné ne peut attribuer des valeurs d'attributs (pour des attributs définis par ce tenant) qu'à ses propres utilisateurs. Dans notre approche, les autorisations globales MTCS-ABAC liées à une tâche donnée comprennent des attributs globaux qui seront définis au niveau du tenant collaboratif. Ces attributs sont les attributs de tâche, les attributs utilisateur et objet liés à la collaboration. $MemberCS(u)$ (membre de la session collaborative) et $SharedCS(o)$ (partagé dans la session) sont des exemples des attributs globaux.

D'autre part, dans ces autorisations (globales), l'administrateur du tenant collaboratif doit spécifier les attributs liés à l'utilisateur qui est chargé d'accomplir cette tâche. Ces attributs d'utilisateur seront définis par le tenant responsable de la tâche de manière confidentielle, autonome et indépendante des autres tenants

collaborateurs et du tenant collaboratif. Pour faire cela, nous proposons d'utiliser une nouvelle fonction d'attribut $AssignUser()$ qui sera définie au niveau du tenant local et sera utilisée par le tenant collaboratif pour définir les autorisations globales.

De la même manière, dans ces autorisations globales, l'administrateur du tenant collaboratif fait appel aux attributs liés à l'objet utilisé dans la tâche. Ces attributs d'objet seront définis par le propriétaire de l'objet de manière confidentielle et indépendante des autres collaborateurs. Pour ce faire, nous proposons une nouvelle fonction d'attribut $UsedObject()$ qui sera définie par le propriétaire de l'objet et sera utilisée par le tenant collaboratif pour définir les autorisations globales. Dans ce qui suit, nous allons voir la définition des deux fonctions $AssignUser()$ et $UsedObject()$.

7.2.4 Fonction AssignUser()

$AssignedUser_{(ta:TA;te:TE)}(u : U) \rightarrow True; False$, un nouveau attribut composé mappant l'utilisateur u à une valeur booléenne $true$ ou $false$. cela signifie que les attributs de l'utilisateur responsable de la tâche ta seront définis par son propriétaire tenant te de manière confidentielle. Cet attribut composé est défini par le tenant chargé d'accomplir la tâche pour spécifier dans sa politique locale, d'une manière confidentielle et autonome, les attributs de l'utilisateur qui effectuera cette tâche. par la suite, cet attribut sera utilisé par le tenant collaboratif pour définir les autorisations globales. Par exemple, l'attribut composé $AssignedUser_{(T7;CHU)}(u)$ sera défini par le tenant CHU pour spécifier les attributs de l'utilisateur autorisé pour exécuter la tâche $T7$. Les indices utilisés dans cet attribut sont :

- $(ta : T)$: la tâche courante (la tâche active) du workflow de la collaboration.
- $(te : TE)$: Le tenant qui va accomplir cette tâche (l'exécuteur de la tâche).
- Le couple $(ta : TA; te : TE)$, signifie que le tenant te est chargé de l'accomplissement de la tâche ta.

7.2.5 Fonction UsedObject()

$UsedObject_{(ObjType;a:A;te:TE)}(o : O) \rightarrow True; False$, un nouveau attribut composé dont la valeur est booléenne, mappant l'objet o à $true$ ou $false$. Ceci signifie

que les attributs liés aux objets de type $ObjType$, pour l'action a seront définis par leur propriétaire tenant d'une manière confidentielle. Cet attribut composé est défini par le tenant propriétaire de la ressource en spécifiant dans sa politique locale les attributs lié à cette ressource ainsi que l'action autorisée. Cette spécification d'attributs de ressource sera faite d'une manière confidentielle et indépendante du tenant collaboratif et des autres tenants participant à la collaboration. A titre d'exemple, l'attribut composé $UsedObject_{(MR;read;HA)}(u)$ sera défini par le tenant HA pour spécifier les attributs d'objets (de type MR) partagés durant la collaboration. Les indices utilisés dans cette fonction sont :

— $ObjType$: un ensemble d'objets satisfaisant une propriété commune, sont classés dans un type d'objet.

— $(a : A)$: l'action liée à l'autorisation.

— $(te : TE)$: le tenant qui partagera l'objet o dans la collaboration.

7.2.6 Formalisation du modèle MTCS-ABAC

Le modèle MTCS-ABAC est défini par un ensemble des entités de base, les attributs et le langage de politique d'autorisation. Ci-dessous, nous allons définir formellement le modèle MTCS-ABAC :

— U, O, T et TE représentent un ensemble fini d'utilisateurs, d'objets, de tâches et de tenants respectivement.

— A représente un ensemble fini d'actions possibles sur les objets.
$A = \{create; read; update; delete\}$.

— CTE représente un ensemble fini de tenants collaboratifs ($CTE \subseteq TE$).

— UA, OA et TA représentent des ensembles finis d'attributs d'utilisateurs, d'objets et de tâches respectivement.

— Pour chaque attribut $att \in UA \cup OA \cup TA$, $range(att)$ représente le champ d'attribut, qui est un ensemble fini de valeurs atomiques.

— $attType : UA \cup OA \cup TA \rightarrow \{set; atomic\}$, le type d'attribut retourne ensemble set ou une valeur atomique.

— $Collabors : (cte : CTE) \rightarrow TE$, spécifie les tenants qui utilisent ce tenant collaboratif cte.

— Chaque fonction d'attribut mappe des éléments en U à une valeur atomique ou un ensemble des valeurs :
  — $\forall ua \in UA.\ ua : U \rightarrow Range(ua)$ si $attType(ua) = atomic$
  — $\forall ua \in UA.\ ua : U \rightarrow 2^{Range(ua)}$ si $attType(ua) = set$
— Chaque fonction d'attribut mappe des éléments en O à une valeur atomique ou un ensemble des valeurs :
  — $\forall oa \in OA.\ oa : O \rightarrow Range(oa)$ if $attType(oa) = atomic$
  — $\forall oa \in OA.\ oa : O \rightarrow 2^{Range(oa)}$ if $attType(oa) = set$
— Chaque fonction d'attribut mappe des éléments en T à une valeur atomique ou un ensemble ;
  — $\forall ta \in TA.\ ta : T \rightarrow Range(ta)$ if $attType(ta) = atomic$
  — $\forall ta \in TA.\ ta : T \rightarrow 2^{Range(ta)}$ if $attType(ta) = set$
— $UOwner : (u : U) \rightarrow TE$, fonction d'attribut requis mappant l'utilisateur u à son tenant propriétaire te.
— $OOwner : (o : O) \rightarrow TE$, fonction d'attribut requis mappant l'objet o à son propriétaire tenant te ;
— $TOwner : (t : T) \rightarrow TE$, fonction d'attribut requis mappant la tâche t à son propriétaire tenant te ;
— $UAOwner : (uatt : UA) \rightarrow TE$, méta attribut, mappant l'attribut utilisateur ua à son propriétaire tenant te ;
— $OAOwner : (oa : OA) \rightarrow TE$, méta attribut, mappant l'attribut objet oa à son tenant te ;
— $TAOwner : (ta : TA) \rightarrow TE$, méta attribut, mappant l'attribut tâche ta à son tenant te ;
— $ua(u : U)$ est définie seulement si $(UAOwner(ua) = UOwner(u)) \vee (UOwner(u) \in Collaborators(UAOwner(ua)))$;
— $oa(o : O)$ est définie seulement si $(OAOwner(oa) = OOwner(o)) \vee (OOwner(u) \in Collaborators(OAOwner(oa)))$;
— $ta(t : T)$ est définie seulement si $(TAOwner(ta) = TOwner(o))$.
— $AssignedUser_{(t:T;te:TE)}(u : U) \rightarrow \{True; False\}$, un nouveau attribut composé mappant l'utilisateur u à une valeur booléenne $true$ ou $false$, cela

signifie que les attributs de l'utilisateur responsable de la tâche t seront définis par son tenant propriétaire te de manière confidentielle et autonome.

— $UsedObject_{(ObjType;a:A;te:TE)}(o : O) \rightarrow \{True; False\}$, un nouveau attribut composé mappant l'objet o à une valeur booléenne $true$ ou $false$, ceci signifie que les attributs liés aux objets de type $ObjType$ pour l'action a seront définis par leur propriétaire tenant te d'une manière confidentielle et indépendante.

— Une autorisation est une fonction booléenne qui prend en paramètres les entités u, o et t avec l'indice a pour décider si l'utilisateur u peut accéder à l'objet o dans la tâche t pour l'action a.
Règle : $authorization_a(u; o; t) \rightarrow f(UA(u); OA(o); TA(t))$, sous la condition nécessaire : $(UOwner(u) = OOwner(o) = TOwner(t)) \vee (UOwner(u) \in Collaborators(TOwner(t)) \wedge OOwner(o) \in Collaborators(TOwner(t)))$.

7.2.7 Exemple de définition des autorisations dans le modèle MTCS-ABAC

Dans cette section, nous rappelons du scénario décrit précédemment d'une application collaborative distribuée de télé-médecine pour le diagnostic dans le domaine de la neurologie. Dans cette étude de cas, le centre hospitalier universitaire CHU de rabat, Le service d'aide médicale urgente $SAMU$ de rabat et l'hôpital d'accueil HA de khemisset sont trois entités collaborant et partageant un cloud privé commun. Dans cet exemple, nous appliquons notre modèle MTCS-ABAC à cette étude de cas en spécifiant les autorisations liées aux tâches « Interpret-Scan » et « Take-a-decision » mentionnées dans la section 7.2.2 (voir le tableau 7.2).

Tout d'abord, pour chaque tâche, nous définissons un ensemble d'attributs de tâche, un ensemble d'attributs d'utilisateur qui est censé d'accomplir cette tâche et un ensemble d'autorisations d'accès qui lui sont liées. Une autorisation est une action exécutée par un utilisateur sur un objet. Un objet est défini par un ensemble d'attributs d'objet. Par exemple $(write, MR)$, $(read, scan)$ et $(read, video)$ sont trois permissions liées à la tâche « take-a-decision ». Les autorisations de contrôle d'accès dans le modèle MTCS-ABAC sont définies par le formalisme mentionné dans le tableau 7.3.

Tâches	***Utilisateurs affectés***	***Objets utilisés***	***Actions***
$task(ti) = T5$ $workflow(ti) = TM$ $pvstask(ti) = true$ $CSession(ti) = cs1$	$MemberCS(u) = cs1$ $AssignedUser_{(T5,CHU)}(u) = True$	$SharedCS(o) = cs1$ $UsedObject_{(Scan,W,HA)}(o) = True$	$Write$
$task(ti) = T7$ $workflow(ti) = TM$ $pvstask(ti) = true$ $CSession(ti) = cs1$	$MemberCS(u) = cs1$ $AssignedUser_{(T7,CHU)}(u) = True$	$SharedCS(o) = cs1$ $UsedObject_{(MR,W,HA)}(o) = True$	$Write$
		$SharedCS(o) = cs1$ $UsedObject_{(Scan,R,HA)}(o) = True$	$Read$
		$SharedCS(o) = cs1$ $UsedObject_{(Video,R,HA)}(o) = True$	$Read$

TABLE 7.3: Exemple d'autorisations du modèle MTCS-ABAC

$$
\begin{aligned}
&authorization_write(ti, u, o) \leftarrow \\
&task(ti) = T5 \wedge \\
&workflow(ti) = TM \wedge \\
&pvstask(ti) = true \wedge \\
&(\exists cs1 \in CS,\, cs1 \in CSession(ti) \wedge \\
&cs1 \in MemberCS(u) \ \wedge \ cs1 \in SharedCS(o)) \wedge \\
&AssignedUser_{(T5;CHU)}(u) = True \wedge \\
&UsedObject_{(SCAN;Write;HA)}(o) = True
\end{aligned}
\tag{7.1}
$$

$$
\begin{aligned}
&UsedObject_{(SCAN;Write;HA)}(o) = True \leftarrow \\
&ObjectType(o) = SCAN \wedge \\
&Sensitivity(o) <= class2
\end{aligned}
\tag{7.2}
$$

$$
\begin{aligned}
&AssignedUser_{(T5;CHU)}(u) = True \leftarrow \\
&Role(u) = radiologist \wedge \\
&Neurology - level(u) \in \{L1, L2, L3\} \wedge \\
&Radiology - level(u) \in \{L2, L3\} \wedge \\
&Cardiology - level(u) \in \{L0, L1, L2, L3\}
\end{aligned}
\tag{7.3}
$$

L'autorisation $Autorisation_{Write}(ti; u; o)$ qui est mentionnée dans l'équation (7.1) correspond à la règle "le radiologue interprète les images de Scan" spécifiée dans la première ligne du tableau 7.3. Cette autorisation est définie dans le tenant collaboratif $CT1$ et se compose d'un ensemble d'attributs de tâche, d'attributs d'utilisateur et d'attributs d'objet. Cette autorisation est valide pour l'action

write si seulement si :

- (1) l'instance *ti* est instanciée à partir de la tâche « Interpret-Scan » ;
- (2) l'instance de tâche *ti* appartient au workflow *tenemo* ;
- (3) les instances de tâches précédentes du *ti* sont accomplies ;
- (4) Il existe une session collaborative *cs* dans laquelle la tâche s'exécute ;
- (5) L'utilisateur u est membre de la session collaborative *cs* ;
- (6) L'objet o est partagé dans la session collaborative *cs* ;
- (7) Le tenant CHU autorise son utilisateur à effectuer la tâche $T5$;
- (8) Le tenant HA partage l'objet o du type $SCAN$ avec d'autres tenants collaborateurs pour l'action *write*.

Les attributs $AssignedUser$ et $UsedObject$ sont définis et évalués respectivement dans les tenants CHU et HA. Cet attribut (équation 7.2) $AssignedUser_{(T5;CHU)}(u) = True$ si seulement si :

- (1) l'utilisateur u joue le rôle radiologue ;
- (2) l'utilisateur u atteint au moins le niveau 1 d'expertise en neurologie ;
- (3) l'utilisateur u atteint au moins le niveau 2 d'expertise en radiologie ;
- (4) u atteint au moins le niveau 0 d'expertise en cardiologie.

L'attribut (équation 7.3) $UsedObject_{(SCAN;Write;HA)}(o) = True$ si seulement si :

- (1) l'objet o du type $SCAN$;
- (2) la classe de sensibilité de l'objet o est inférieure ou égale à la $classe2$.

7.3 Implémentation

7.3.1 L'architecture d'implémentation

Dans notre implémentation, nous nous intéressons plus particulièrement à la composante de stockage d'Openstack Swift. Cette composante est un système de stockage de fichiers, vidéos, données, données analytiques, contenus web, sauvegardes, images, instances de machines virtuelles et autres données non structurées. Swift utilise pour les sauvegardes plusieurs serveurs. Si un serveur ou un disque dur tombe en panne, Swift réplique ses données depuis des nœuds actifs du cluster dans de nouveaux emplacements.

Le serveur de compte (Account) est responsable de lister les conteneurs, tandis que le serveur de conteneur (Containers) est responsable de lister les objets. Un conteneur (Container) est un mécanisme qui stocke des objets de données. Un compte peut avoir plusieurs conteneurs, tandis qu'un conteneur doit être associé à un seul compte. Swift utilise le modèle d'accès à base de listes (ACL) pour définir les politiques d'accès des utilisateurs aux objets stockés dans le conteneur. Cependant, le modèle ACL définit des règles d'accès statiques et qui ne sont pas fines. Ceci n'est pas adapté pour les systèmes collaboratifs.

Dans ce travail, nous avons implémenté le modèle MTCS-ABAC [38] sur la composante de stockage Swift. Cette composante agit comme un service qui communique avec d'autres composants (Nova volume, nova compute, nova network, glance et keystone) via une file d'attente de messages. Ces composants sont faiblement couplés. Keystone est le service d'identité utilisé par OpenStack pour l'authentification et l'autorisation. Il fournit un jeton signé (Token) comportant la clé privée de chaque utilisateur.

Dans notre scénario, nous considérons que nous avons trois tenants collaborateurs : $SAMU$, CHU et HA. Ces tenants partagent le même cloud privé commun (Openstack). Nous considérons que chaque tenant correspond à un compte de swift (Account) (par exemple, les comptes $ACC\text{-}CHU$, $ACC\text{-}SAMU$ et $ACC\text{-}HA$ représentent respectivement les tenants CHU, $SAMU$ et HA).

De plus, ce cloud comporte un tenant collaboratif $ACC\text{-}CT$ fournissant un service de gestion de collaboration aux autres tenants. Ce tenant collaboratif est responsable pour le déroulement des sessions collaboratives ainsi que la gestion des accès inter-tenants. Dans cet exemple, nous supposons que le compte $ACC\text{-}CHU$ comporte quatre utilisateurs : $ACC\text{-}user1$, $ACC\text{-}user2$, $ACC\text{-}user3$ et $ACC\text{-}user4$. De même, chaque compte a plusieurs conteneurs et chaque conteneur contient un ensemble d'objets. Par exemple, le compte $ACC\text{-}HA$ comporte quatre conteneurs : MR, DEC, $SCAN$ et $DIAG$.

Afin de prendre en compte le modèle MTCS-ABAC dans l'environnement OpenStack Swift et de surmonter les limites de Swift ACL, nous proposons d'étendre la composante Swift en ajoutant un nouveau module "MTCS-ABAC module" (Figure 7.4). Le module MTCS-ABAC est composé de six composants : les attributs d'utilisateur, les attributs d'objet, les attributs de tâche, les attributs composés, les autorisations et la politique de décision. Dans ce qui suit, nous

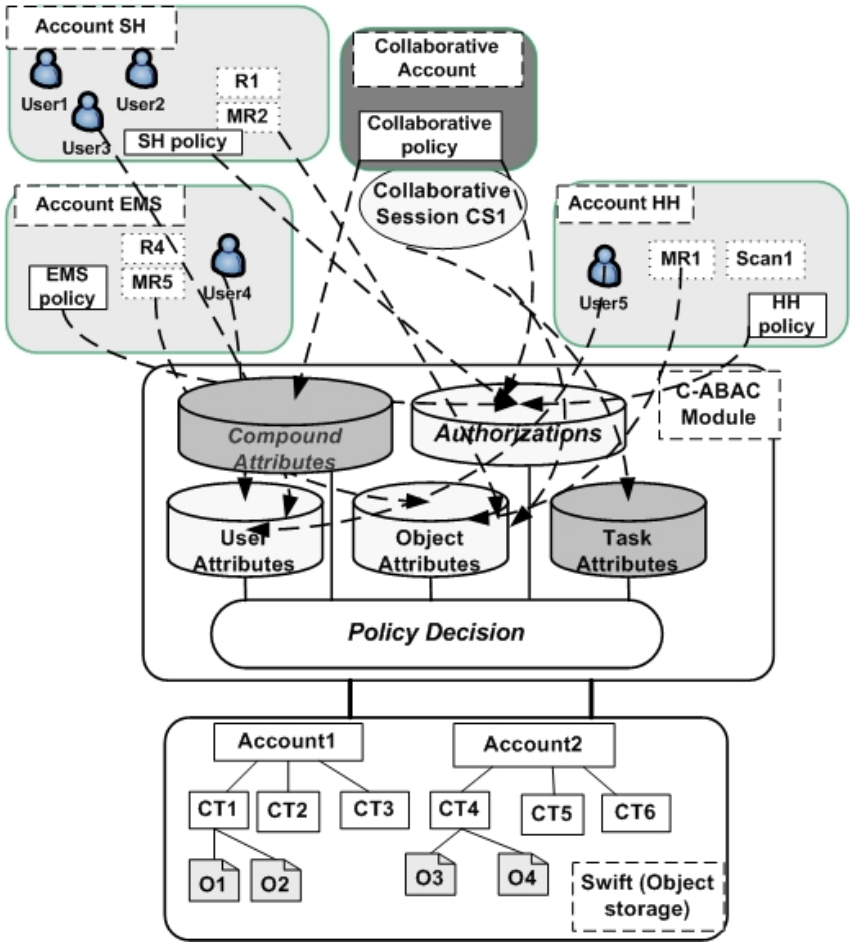

FIGURE 7.4: L'architecture d'implémentation

allons voir en détail chacun de ces composants :

— **Attributs d'utilisateur :** L'administrateur de la politique d'accès définit l'attribut d'utilisateur comme étant une fonction qui prend en paramètre un utilisateur et retourne une valeur ou un ensemble des valeurs. ($user1$: $attr1$: $val1$), signifie que, pour l'utilisateur $user1$ la valeur de l'attribut $attr1$ est $val1$. Par exemple, l'attribut $Role \in UATT$ mappe l'utilisateur $user1 \in U$ à la valeur $neurologist$, en utilisant le formalisme suivant : ($user1 : Role : neurologist$).

De plus, l'administrateur du cloud définit l'attribut $UOwner$ pour spécifier le tenant propriétaire de chaque utilisateur. ($user1 : UOwner : ACC\_SH$) signifie que $user1$ est détenu par le compte ACC_SH. Enfin, l'administrateur définit l'attribut $JoinCS$ pour spécifier les utilisateurs qui peuvent

rejoindre les sessions collaboratives. La valeur de cet attribut est booléen ($true$ ou $false$). ($user1 : JoinCS : true$) signifie que $user1$ peut participer à la collaboration ;

— **Attributs d'objet :** L'administrateur du tenant affecte les attributs d'objet en tant que fonction qui prend en paramètre un objet et retourne une valeur appartenant à l'ensemble des valeurs possibles. ($obj1 : attr1 : val1$) signifie que pour l'objet $obj1$ la valeur de l'attribut $attr1$ est $val1$. De plus, l'administrateur du cloud définit l'attribut $OOwner$ pour spécifier le propriétaire de l'objet. ($MR1 : UOwner : ACC\text{-}HH$) signifie que l'objet $MR1$ est détenu par le compte $ACC\text{-}HH$. Enfin, l'administrateur définit la fonction d'attribut $SharedCS$ pour déterminer les objets qui pourraient être partagés dans la session collaborative. Par exemple, ($Per\text{-}info1 : SharedCS : false$) signifie que l'objet $Per\text{-}info1$ (informations personnelles) ne doit pas être partagé dans la session collaborative.

— **Attributs de tâche :** L'administrateur affecte les attributs de tâche en tant que fonction qui prend en paramètre une instance de tâche et retourne une valeur atomique ou un ensemble des valeurs. ($ti1 : attr1 : val1$) signifie que pour l'instance de tâche $ti1$ la valeur de l'attribut $attr1$ est $val1$. En outre, l'administrateur du cloud définit l'attribut $TOwner$ pour spécifier le propriétaire de la tâche. Par exemple ($t1 : TOwner : ACC\text{-}EMS$) signifie que la tâche $t1$ est définie par le compte $ACC\text{-}EMS$;

— **Attributs composés :** Dans ce composant, l'administrateur du tenant définit les attributs composés $AssignedUser$ et $UsedObject$ d'une manière confidentielle et autonome. Ces attributs composés sont spécifiés ici comme suit : $UsedObject|SCAN|Write|HA : -o : ObjectType : SCAN \wedge o : sensitivity : class0|class1|class2$, ce qui signifie que cet attribut est vrai si seulement si : (1) l'objet o est de type $Scan$; (2) la classe de sensibilité de l'objet o est inférieure ou égale à $class2$;

— **Autorisations :** L'administrateur définit ici les autorisations de la politique MTCS-ABAC. Dans notre scénario, nous considérons que chaque tenant spécifie ses règles de politique de manière autonome. Notez qu'à ce niveau, nous supposons que les règles de politique d'accès sont valides et sans conflits. Les autorisations de la politique sont spécifiées ici comme suit :

$write - ti : task : interpret\_scan \wedge$
$ti : workflow : tenemo \wedge$
$ti : previoustask : true \wedge$
$ti : CSession : cs1 \wedge u : memberCS : cs1 \wedge$
$o : sharedCS : cs1 \wedge$
$u : AssignedUser|T5|CHU : True \wedge$
$o : UsedObject|SCAN|Write|HA : True$
ce qui signifie que pour l'action *write*', cette autorisation est valide si seulement si : (1) ti est une instance de la tâche $interpret\_scan$; (2) l'instance de tâche ti appartient au workflow $tenemo$; (3) les instances de tâche précédentes du ti sont accomplies ; (4) il y a une session collaborative $cs1$ dans laquelle l'instance de tâche ti s'exécute ; (5) l'utilisateur u est membre de la session collaborative $cs1$; (6) l'objet o est partagé dans la session $cs1$; (7) le tenant CHU autorise son utilisateur u à effectuer cette tâche $T5$; (8) le tenant HA partage l'objet o du type $SCAN$ avec d'autres tenants collaborateurs pour l'action $Write$.

— **Politique de décision :** Ce composant est responsable d'évaluer les requêtes d'accès d'un utilisateur à une ressource partagée dans la session collaborative. L'évaluation des requêtes se fait en fonction des valeurs attributs et des autorisations collectées. Lorsqu'un utilisateur envoie une requête pour accéder à une ressource stockée dans le cloud swift. Ce composant de politique de décision évalue cette requête afin de décider si l'utilisateur est autorisé à accéder à cette ressource ou non.

7.3.2 Modèle d'application

Dans ce travail, nous présentons le processus d'autorisation pour le composant Swift en utilisant le module MTCS-ABAC (voir la figure 7.5). Lorsque l'utilisateur $user1$ tente d'accéder à la ressource $MR1$ stockée dans swift. Premièrement, (1) L'utilisateur demande à Keystone d'obtenir son jeton (token). (2) Keystone génère un jeton et l'envoie à l'utilisateur. (3) L'utilisateur envoie une requête au module MTCS-ABAC en utilisant son jeton pour accéder à la ressource $MR1$. Le composant "policy decision" reçoit cette requête pour l'évaluer.
(4) Durant le processus d'évaluation, le composant "policy decision" demande aux autres composants : attributs d'utilisateur, attributs d'objet et attributs de

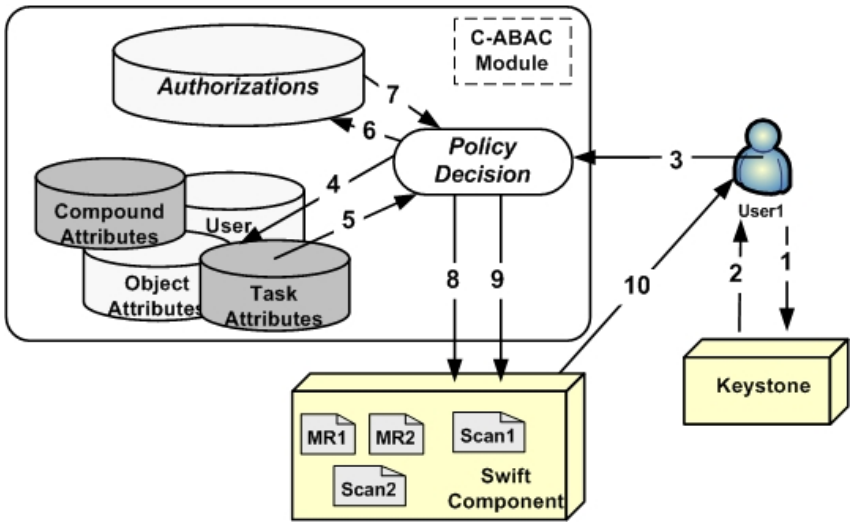

FIGURE 7.5: Le modèle d'application

tâche pour qu'ils lui envoient les valeurs d'attributs utilisées pour l'évaluation. (5) Puis, il reçoit les attributs de $user1$, les attributs de $MR1$ et les attributs de la tâche relative à cette demande d'accès.
(6) Le composant "policy decision" demande au composant d'autorisation de lui envoyer les autorisations de la politique. Ces attributs et ces autorisations collectés seront utilisés par le composant "policy decision" pour évaluer la requête d'accès afin de décider si l'utilisateur est autorisé à accéder à cette ressource ou non.

(8) "policy decision" exécutera une commande ACL pour attribuer la décision d'autorisation (accès ou refus) à l'utilisateur dans l'environnement swift. (9) Le composant "policy decision" exécute une commande d'API Swift dans l'environnement swift en utilisant le jeton $user1$ afin d'envoyer la requête d'accès de $user1$ à swift. (10) $user1$ accède à la ressource $MR1$ si la décision d'accès lui autorise.

7.3.3 Mesures de performance

Dans ce travail, nous implémentons ABAC et MTCS-ABAC sur la composante de stockage Swift d'openstack. Nos simulations ont été exécutées sur une machine virtuelle avec les caractéristiques suivantes (mémoire 1024 Mo, processeur 2 cœurs, disque dur 30 Go). Nous considérons le temps d'accès à un objet Swift en utilisant le modèle ABAC et le modèle MTCS-ABAC. Nous observons que la performance de l'application de notre approche dépend de nombreux fac-

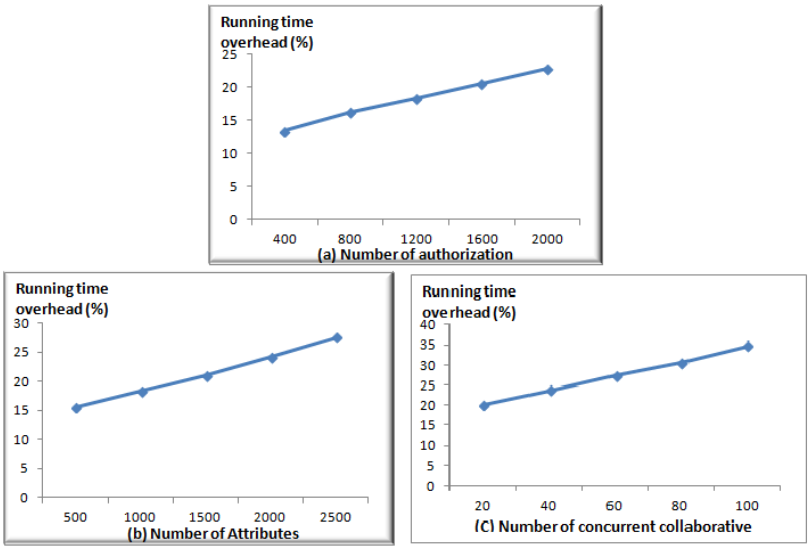

FIGURE 7.6: Délai d'exécution pour les décisions d'accès / refus

teurs, tels que le nombre de règles, le nombre d'attributs et le nombre de sessions collaboratives actives. Dans notre analyse, nous avons utilisé un jeu de données (dataset) synthétique contenant jusqu'à 2000 règles, 2500 attributs et 100 sessions collaboratives concourantes.

La figure 7.6 (a) montre que la durée moyenne pour autoriser l'accès à un objet Swift en utilisant le modèle ABAC, augmente de 13.4% et 22.7% pour les politiques comportant de 400 et 2000 règles respectivement en utilisant le modèle MTCS-ABAC. La durée de décision pour décider si un accès est autorisé ou pas, devient plus important lorsqu'il y a trop d'autorisations à collecter. Nous constatons que notre implémentation fonctionne bien pour un nombre important d'autorisations.

De plus, nous calculons le temps de prise de décisions d'accès à un objet Swift en utilisant les modèles ABAC et MTCS-ABAC pour 400 règles et pour 500 à 2500 attributs. La figure 7.6 (b) montre que la durée moyenne pour l'accès à une ressource Swift en utilisant le modèle MTCS-ABAC augmente de 15.6% et 27.6% pour 500 et 2500 attributs utilisateurs respectivement par rapport au modèle ABAC. Nous constatons que notre implémentation fonctionne bien pour un nombre important d'attributs.

Enfin, nous calculons le temps d'exécution des décisions d'accès à un objet Swift en utilisant le modèle ABAC et MTCS-ABAC pour 400 règles, 500 attributs

et le nombre de sessions collaboratives actives de 10 à 50. La figure 7.6 (c) montre que la durée moyenne des décisions d'accès à Swift ressources en utilisant le modèle MTCS-ABAC augmente respectivement de 20.1% et 34.7% pour 10 et 100 sessions collaboratives actives par rapport à l'utilisation du modèle ABAC. Nous observons que notre implémentation fonctionne d'une manière normale pour un nombre moyen de sessions collaboratives qui sont actives simultanées.

7.4 Conclusion

Dans ce travail, nous avons présenté un nouveau modèle de contrôle d'accès basé sur ABAC appelé (MTCS-ABAC), pour assurer la collaboration entre plusieurs tenants. Ce modèle permet aux tenants participant à la collaboration de définir la politique globale de contrôle d'accès d'une manière collaborative sur un point central tout en conservant l'autonomie et la confidentialité de chaque tenant. De plus, ce modèle prend en compte les workflows collaboratifs s'exécutant sur un environnement multi-tenant ; Enfin, nous avons proposé une architecture d'implémentation permettant d'intégrer le modèle MTCS-ABAC sur la composante de stockage du cloud Openstack. Cette implémentation nous permet de montrer la faisabilité de modèle et de vérifier leur cohérence et leur complétude. Les résultats de simulations ont montré que le temps de décisions d'accès a subi une augmentation limitée en utilisant le modèle MTCS-ABAC par rapport au modèle ABAC.

Chapitre 8

Conclusion et perspectives

Dans ce travail, nous avons considéré le problème de contrôle d'accès dans des sessions collaboratives se déroulant dans un environnement multi-tenant. Les modèles de contrôles d'accès traditionnels (ACL, $RBAC$, $TBAC$, $ABAC$, etc) ne permettent pas de définir des politiques d'accès pour les environnements multi-domaines. Généralement, ils sont destinés pour définir la politique d'accès locale d'une organisation donnée. D'autre part, les modèles de contrôle d'accès existants pour la collaboration et ceux pour les environnements multi-tenants permettent d'assurer l'accès inter-domaines ainsi que l'accès inter-tenants. Cependant, ces modèles ne supportent pas les concepts : de "session collaborative", de "multi-tenancy" et de "workflow".

Pour cela, nous avons contribué à l'extension des modèles de contrôle d'accès les plus répandus à savoir : $RBAC$, $OrBAC$, $TRBAC$ et $ABAC$. Les extensions proposées sont dans le but d'améliorer l'expressivité des politiques de contrôle d'accès à un niveau élevé de granularité ainsi que la flexibilité des modèles. Ces modèles étendus prennent en compte les points suivants : (1) la multi-tenancy ainsi que le partage des ressources entre tenants dans un environnement Cloud ; (2) l'aspect dynamique de la session collaborative ; (3) les processus collaboratifs composés d'une suite de tâches accomplies par différents tenants ; Nous listons ci-dessous nos principales contributions :

— Nous avons proposé MTCS-OrBAC qui est un modèle de contrôle d'accès basé sur le modèle OrBAC pour assurer le contrôle d'accès des ressources partagées lors d'une session collaborative. Le principal objectif du modèle MTCS-OrBAC est d'étendre le modèle OrBAC pour qu'il prenne en compte

le concept de la "multi-tenancy" et l'aspect dynamique de la session collaborative ;

— Nous avons proposé MTCS-RBAC qui est une extension du modèle RBAC en introduisant de nouvelles entités comme le tenant, la session collaborative. De plus, MTCS-RBAC introduit des relations de confiance établies entre les tenants, afin de supporter le partage inter-tenants des ressources lors d'une session collaborative ;

— Modèle MTCS-TRBAC est une extension du modèle TRBAC. Ce modèle MTCS-TRBAC est similaire au modèle précédent MTCS-RBAC, la seule différence est que le modèle MTCS-TRBAC prenne en compte le concept du "workflow" dans un environnement collaboratif Multi-Tenant. Ceci a pour objectif de supporter les processus collaboratifs s'exécutant par les différents tenants ;

— Nous avons suggéré MTCS-ABAC qui est un modèle de contrôle d'accès basé sur le modèle ABAC pour assurer la collaboration entre plusieurs tenants. Ce modèle permet aux tenants participant à la collaboration de définir la politique globale de contrôle d'accès d'une manière collaborative sur un point central tout en conservant l'autonomie et la confidentialité de chaque tenant. De plus, ce modèle prend en compte les workflows collaboratifs s'exécutant sur un environnement multi-tenant ;

— Enfin, nous avons proposé une architecture d'implémentation permettant d'intégrer les modèles étendus sur la composante de stockage du cloud Openstack. Les résultats de simulations ont montré que le temps de décisions d'accès a subi une augmentation limitée en utilisant le modèle MTCS-ABAC par rapport au modèle ABAC.

Perspectives de travail

Nous allons clôturer ce mémoire par la présentation des perspectives souhaitables à notre travail :

— **Prise en compte de la notion "Break-Glass" (en verre brisé) [48]** : étendre les modèles proposés dans cette thèse en intégrant la technique de "Break-Glass" de manière plus fine. Aujourd'hui, les techniques de "Break-Glass" sont généralement ajoutées aux solutions de contrôle d'accès standard de manière ad hoc. En faite, elle permet de prévenir le fonctionnement

d'un système contre un blocage qui peut entrainer des conséquences graves. cette technique est très intéressante surtout si nous souhaitons utiliser ces modèles dans un contexte de télé-médecine. De plus, proposer une architecture d'application permettant d'intégrer la technique de "Break-Glass" sur la plateforme Openstack ;

— **Détection et résolution des conflits** : proposer une solution formelle et générique pour chaque modèle, afin de détecter et résoudre le problème des conflits potentiels qui peuvent survenir durant la collaboration. Un conflit peut se produire de façon inattendue à un système, lorsqu'une action dans le système, pour un utilisateur donnée, est à la fois permise et interdite par la politique de contrôle d'accès. Ce type de problème a fait l'objet de plusieurs travaux de recherche, notamment pour [23] et [22]. Afin de résoudre ce problème, il faut définir le type des priorités à appliquer en cas de conflits entre les règles d'accès.

— **Hétérogénéité des environnements multi-clouds** : nous envisageons d'étendre ces modèles pour assurer le contrôle d'accès lors des sessions de collaboration se déroulant dans des environnements multi-clouds hétérogènes [50]. C'est-à-dire, les utilisateurs membres de la session et les ressources partagées dans la session, appartenant aux différents fournisseurs de cloud. Pour cela, il faut proposer un framework agissant comme médiateur entre les clouds, pour assurer l'interopérabilité entre des clouds hétérogènes. Il est donc souhaitable que ce framework fournis à ces clouds, un standard commun de communication, de collaboration et d'échange de données.

Bibliographie

[1] Description du projet pmars. a secure collaborative platform in the cloud for remote medical diagnosis (secop-cremd) (2013-2015).

[2] Openstack platform. https ://www.openstack.org/.

[3] Openstack swift architecture. https ://swiftstack.com/openstackswift/architecture/.

[4] Tanvir Ahmed and Anand R. Tripathi. Specification and verification of security requirements in a programming model for decentralized cscw systems. *ACM Trans. Inf. Syst. Secur.*, 10(2), May 2007.

[5] Gail-Joon Ahn and Ravi Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3(4) :207–226, November 2000.

[6] A. Almutairi, M. Sarfraz, S. Basalamah, W. Aref, and A. Ghafoor. A distributed access control architecture for cloud computing. *IEEE Software*, 29(2) :36–44, March 2012.

[7] Amine Baïna. *Controle d'accès pour les grandes infrastructures critiques. Application au réseau d'énergie électrique*. PhD thesis, INSA Toulouse, France, 2009.

[8] E. D. Bell and J. L. La Padula. Secure computer system : Unified exposition and multics interpretation, 1976.

[9] E. Bertino, S. Jajodia, and P. Samarati. Supporting multiple access control policies in database systems. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 94–107, May 1996.

[10] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. Trbac : A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3) :191–233, August 2001.

[11] Prosunjit Biswas, Farhan Patwa, and Ravi Sandhu. Content level access control for openstack swift storage. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, pages 123–126, New York, NY, USA, 2015. ACM.

[12] Prosunjit Biswas, Ravi Sandhu, and Ram Krishnan. An attribute-based protection model for json documents. In Jiageng Chen, Vincenzo Piuri, Chunhua Su, and Moti Yung, editors, *Network and System Security*, pages 303–317, Cham, 2016. Springer International Publishing.

[13] J. M. A. Calero, N. Edwards, J. Kirschnick, L. Wilcock, and M. Wray. Toward a multi-tenancy authorization system for cloud services. *IEEE Security Privacy*, 8(6) :48–55, Nov 2010.

[14] Peter H. Carstensen and Kjeld Schmidt. Computer supported cooperative work : New challenges to systems design. In *In K. Itoh (Ed.), Handbook of Human Factors*, pages 619–636, 1999.

[15] Frédéric Cuppens. O2o : Managing security policy interoperability with virtual private organizations. pages 62–68, mai 2006.

[16] Frédéric Cuppens, Nora Cuppens-Boulahia, and Céline Coma. O2o : Virtual private organizations to manage security policy interoperability. In Aditya Bagchi and Vijayalakshmi Atluri, editors, *Information Systems Security*, pages 101–115, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[17] Frédéric Cuppens and Alexandre Miège. Adorbac : an administration model for or-bac. *International Journal of Computer Systems Science and Engineering (CSSE'04)*, 19, 2004.

[18] Sabrina De Capitani di Vimercati and Pierangela Samarati. Access control in federated systems. In *Proceedings of the 1996 Workshop on New Security Paradigms*, NSPW '96, pages 87–99, New York, NY, USA, 1996. ACM.

[19] H. Dommel and J. J. Garcia-Luna-Aceves. Group coordination support for synchronous internet collaboration. *IEEE Internet Computing*, 3(2) :74–80, March 1999.

[20] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3) :224–274, August 2001.

[21] Serban I. Gavrila and John F. Barkley. Formal specification for role based access control user/role and role/role relationship management. In *Proceedings of the Third ACM Workshop on Role-based Access Control*, RBAC '98, pages 81–90, New York, NY, USA, 1998. ACM.

[22] H. Hu, G. Ahn, and K. Kulkarni. Discovery and resolution of anomalies in web access control policies. *IEEE Transactions on Dependable and Secure Computing*, 10(6) :341–354, Nov 2013.

[23] Hongxin Hu, Gail-Joon Ahn, and Jan Jorgensen. Detecting and resolving privacy conflicts for collaborative data sharing in online social networks. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 103–112, New York, NY, USA, 2011. ACM.

[24] X. Jin, R. Krishnan, and R. Sandhu. Role and attribute based collaborative administration of intra-tenant cloud iaas. In *10th IEEE International Conference on Collaborative Computing : Networking, Applications and Worksharing*, pages 261–274, Oct 2014.

[25] Xin Jin, Ram Krishnan, and Ravi Sandhu. A unified attribute-based access control model covering dac, mac and rbac. In *Proceedings of the 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*, DBSec'12, pages 41–55, Berlin, Heidelberg, 2012. Springer-Verlag.

[26] A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 120–131, June 2003.

[27] A. A. El Kalam, Y. Deswarte, A. Baina, and M. Kaaniche. Access control for collaborative systems : A web services based approach. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 1064–1071, July 2007.

[28] Anas Abou El Kalam and Yves Deswarte. Multi-orbac : a new access control model for distributed, heterogeneous and collaborative systems. In *8th IEEE International Symposium on Systems and Information Security*, SSI '06, pages 42–50, Sao Paulo, Brésil, 2006. IEEE Computer Society.

[29] Aymen Kamoun. *Software architectures adaptation access control in ubiquitous collaborative environments.* Theses, Universite de toulouse 1, November 2014.

[30] Angelos D. Keromytis and Jonathan M. Smith. Requirements for scalable access control and security management architectures. *ACM Trans. Internet Technol.*, 7(2), May 2007.

[31] Butler W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8(1) :18–24, January 1974.

[32] Dan Lin, Prathima Rao, Elisa Bertino, Ninghui Li, and Jorge Lobo. Policy decomposition for collaborative access control. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, pages 103–112, New York, NY, USA, 2008. ACM.

[33] M. A. Madani and M. Erradi. Formalisation d'une session collaborative sécurisée. In *2012 National Days of Network Security and Systems*, pages 12–17, April 2012.

[34] M. A. Madani and M. Erradi. Towards a session based orbac model. In *2012 International Conference on Multimedia Computing and Systems*, pages 918–923, May 2012.

[35] M. A. Madani and M. Erradi. How to secure a collaborative session in a single tenant environment. In *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, pages 1–6, July 2015.

[36] M. A. Madani, M. Erradi, and Y. Benkaouz. Access control in a collaborative session in multi tenant environment. In *2015 11th International Conference on Information Assurance and Security (IAS)*, pages 1–6, Dec 2015.

[37] M. A. Madani, M. Erradi, and Y. Benkaouz. A collaborative task role based access control model. *JOURNAL OF INFORMATION ASSURANCE AND SECURITY*, 11(6) :348–358, 2016.

[38] M. A. Madani, M. Erradi, and Y. Benkaouz. Abac based online collaborations in the cloud. In *Emerging Technologies for Developing Countries*, pages 67–76, Cham, 2018. Springer International Publishing.

[39] M. A. Madani, M. Erradi, and Y. Benkaouz. C-abac : An abac based model for collaboration in multi-tenant environment. *EAI Endorsed Transactions on Smart Cities*, 2(8), June 2018.

[40] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States, 2011.

[41] B. Nasser, R. Laborde, A. Benzekri, F. Barrère, and M. Kamel. Access control model for inter-organizational grid virtual organizations. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2005 : OTM 2005 Workshops*, pages 537–551, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[42] Sejong Oh and Seog Park. Task-role-based access control model. *Inf. Syst.*, 28(6) :533–562, September 2003.

[43] Sejong Oh and Ravi Sandhu. A model for role administration using organization structure. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, SACMAT '02, pages 155–162, New York, NY, USA, 2002. ACM.

[44] Navid Pustchi, Ram Krishnan, and Ravi Sandhu. Authorization federation in iaas multi cloud. In *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, SCC '15, pages 63–71, New York, NY, USA, 2015. ACM.

[45] Navid Pustchi and Ravi Sandhu. Mt-abac : A multi-tenant attribute-based access control model with tenant trust. In Meikang Qiu, Shouhuai Xu, Moti Yung, and Haibo Zhang, editors, *Network and System Security*, pages 206–220, Cham, 2015. Springer International Publishing.

[46] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2) :38–47, Feb 1996.

[47] Ravi S. Sandhu. Role hierarchies and constraints for lattice-based access controls. In *Proceedings of the 4th European Symposium on Research in Computer Security : Computer Security*, ESORICS '96, pages 65–79, London, UK, UK, 1996. Springer-Verlag.

[48] Sigrid Schefer-Wenzl and Mark Strembeck. Generic support for rbac break-glass policies in process-aware information systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1441–1446, New York, NY, USA, 2013. ACM.

[49] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Secure collaboration in mediator-free environments. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 58–67, New York, NY, USA, 2005. ACM.

[50] M. Singhal, S. Chandrasekhar, T. Ge, R. Sandhu, R. Krishnan, G. Ahn, and E. Bertino. Collaboration in multicloud computing environments : Framework and security issues. *Computer*, 46(2) :76–84, Feb 2013.

[51] H. Takabi, J. B. D. Joshi, and G. Ahn. Securecloud : Towards a comprehensive security framework for cloud computing environments. In *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*, pages 393–398, July 2010.

[52] H. Takabi, J. B. D. Joshi, and G. Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security Privacy*, 8(6) :24–31, Nov 2010.

[53] B. Tang, Q. Li, and R. Sandhu. A multi-tenant rbac model for collaborative cloud services. In *2013 Eleventh Annual Conference on Privacy, Security and Trust*, pages 229–238, July 2013.

[54] B. Tang and R. Sandhu. Cross-tenant trust models in cloud computing. In *2013 IEEE 14th International Conference on Information Reuse Integration*, pages 129–136, Aug 2013.

[55] B. Tang, R. Sandhu, and Q. Li. Multi-tenancy authorization models for collaborative cloud services. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, pages 132–138, May 2013.

[56] R. K. Thomas and R. S. Sandhu. *Task-based authorization controls (TBAC) : a family of models for active and enterprise-oriented authorization management*, pages 166–181. Springer US, Boston, MA, 1998.

[57] Roshan K. Thomas. Team-based access control (tmac) : A primitive for applying role-based access controls in collaborative environments. In *Proceedings of the Second ACM Workshop on Role-based Access Control*, RBAC '97, pages 13–19, New York, NY, USA, 1997. ACM.

[58] William Tolone, Gail-Joon Ahn, Tanusree Pai, and Seng-Phil Hong. Access control in collaborative systems. *ACM Comput. Surv.*, 37(1) :29–41, March 2005.

[59] Thierry Villemur. *Modèles et services logiciels pour le travail collaboratif.* Habilitation à diriger des recherches, Université Paul Sabatier - Toulouse III, September 2006.

[60] E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*, page 569, July 2005.

[61] Yue Zhang and James B. D. Joshi. *Access Control and Trust Management for Emerging Multidomain Environments*, volume 4 of *Handbooks in Information Systems*. Emerald Group Publishing Limited, June 2009.

[62] Z. Zhang, X. Zhang, and R. Sandhu. Robac : Scalable role and organization based access control models. In *2006 International Conference on Collaborative Computing : Networking, Applications and Worksharing*, pages 1–9, Nov 2006.