Modélisation et vérification de protocoles pour des communications sécurisées de groupes

Sara del Socorro Mota Gonzalez

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE TOULOUSE
délivré par l'**INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE**

École Doctorale : Systèmes
Discipline : Informatique

présentée et soutenue

par

Sara del Socorro MOTA GONZALEZ

le 5 juin 2008

Modélisation et vérification de protocoles pour des communications sécurisées de groupes

Directeurs de thèse :
Thierry VILLEMUR et Michel DIAZ

JURY
Jean-Charles FABRE, Président
Ana CAVALLI
Luciano CHIRINOS GAMBOA
Isabelle CHRISMENT
Michel DIAZ
Refik MOLVA
Thierry VILLEMUR

Rapport LAAS n°

A Oscar

A Jared, Mauricio et Diego

A ma famille

Remerciements

Mon séjour au Laboratoire d'Analyse et d'Architecture des Systèmes, lieu du développement de mes travaux de thèse, me laisse un excellent souvenir. Je remercie donc MM. Ghallab et Chatila, directeurs successifs du LAAS-CNRS, pour m'avoir accueillie dans leur laboratoire. Je tiens également à remercier François Vernadat, responsable du groupe OLC du LAAS-CNRS, pour m'avoir accueillie et appuyée dans mes travaux.

J'exprime ma très sincère reconnaissance à Messieurs Michel Diaz et Thierry Villemur, respectivement Chargé de Recherche CNRS et Maître de Conférences, pour m'avoir encadrée, soutenue et encouragée pendant le déroulement des travaux exposés dans ce mémoire. Je les remercie tous deux pour ce soutien constant et pour la confiance qu'ils m'ont toujours témoignée. Sans leur patience, sans les conseils qu'ils m'ont prodigués tout au long de ces quatre ans, sans leurs mots d'encouragement, ce travail n'aurait pas pu être mené a son terme. Leur encadrement restera pour moi un très bon souvenir et m'aura grandement appris.

J'adresse une mention spéciale à M. Diaz, car il possède une grande compétente et de grandes capacités, au contact desquelles les autres peuvent avancer et s'enrichir. Je me sens honorée d'avoir travaillé avec lui.

Je remercie Ana Cavalli et Refik Molva qui ont accepté les charges de rapporteurs et de membre du jury. Je remercie également Jean-Charles Fabre, Isabelle Chrisment et Luciano Chirinos de m'avoir fait l'honneur d'examiner mes travaux. Leur présence dans le jury fut pour moi un grand plaisir.

Les personnes qui font partie du groupe OLC ont rendu mon séjour très agréable, grâce à la gentillesse et la convivialité dont ils ont su faire preuve. Grâce à leurs grandes compétences et à leurs remarquables qualités humaines, ils sont su rendre chaleureuses et fructueuses ces quelques années passées parmi eux. Je garde de très bons souvenirs des nombreuses discussions qui m'ont changée, qui m'ont construite et qui m'ont poussée jusqu'à la conclusion de mes travaux. Enfin, j'ai une pensée particulière pour Pierre de Saqui-Sannes et Philippe Owezarski pour m'avoir soutenue et pour avoir partagé leur expérience tout au long de mon séjour dans ce laboratoire.

Mes remerciements vont également à tous les membres du projet SAFECAST. J'exprime ma reconnaissance à Benjamin, avec qui j'ai partagé mes connaissances, mon expérience et l'envie de faire avancer les choses. Je tiens également à remercier Philippe, Pierre et Thierry pour nous avoir encadré, Benjamin et moi-même, dans ce projet.

J'exprime une grande reconnaissance envers mon université, l'ITESM Campus Toluca. Plus particulièrement, le témoignage de ma reconnaissance s'adresse à M. Rueda, Mme. Ortiz et M. Enriquez, directeurs successifs de l'ITESM Campus Toluca ; à MM. Chirinos et Miranda, directeurs successifs de l'Ecole d'Ingénieurs et d'Architecture de l'ITESM ; et à M. Gutierrez, directeur du département d'Informatique.

Je remercie mon pays, le Mexique, dont le soutien financier, à travers l'organisme CONACYT, m'a permis de vivre en France et d'effectuer mon doctorat à Toulouse. Ce soutien contribue aussi à la découverte d'autres cultures, ce qui devient aujourd'hui essentiel.

Je n'oublie pas de saluer les doctorants actuels et anciens que j'ai pu côtoyer au cours de ces années. Je salue donc mes amis Luis, Felipe et Fabiana, avec lesquels nous avons partagé nos solitudes, et j'adresse une pensée spéciale pour mes amis Jean-François, German, Ismail et Riadh. Parmi les vieux docteurs et les jeunes maîtres de

conférences, mes pensées vont vers Francisco, Guillermo, David, Idalia, Roberta, Magnos, Carmen, Fernando et Hector desquels j'ai beaucoup appris. Merci à mes amis Brésiliens Olga, Eraldo, René et Marcos, qui m'ont transmis une petite partie de leur univers.

Je remercie chaleureusement toute ma famille de m'avoir épaulée et soutenue. Merci à mes parents d'avoir cru en moi, de m'avoir accompagnée dans toutes les circonstances, et de m'avoir encouragée jusqu'au bout, y compris le jour de la soutenance. Merci aussi à mes enfants, Diego, Mauricio et Jared, qui m'ont accompagné en France, avec lesquels nous avons partagé tant de choses, de grands moments mais aussi parfois de difficultés.

Grâce à nos voisins et amis, nous ne nous sommes jamais sentis seuls en France, notamment avec les membres de la famille Viguier : Sandrine, Philippe, Nico et Alex ; de la famille Fourré : Pascale, Pascal, Lili et Simon; de la famille Mas : Sophie, Pierre, Virgie, Caro et Bruno, et de la famille Labaune : Marie-Hélène, Eric, Robin, Miléna et Naomie dont nous avons vu grandir ensemble les enfants. Merci à mes frères et à mes sœurs pour leurs encouragements.

J'ai un sentiment spécial pour Patricia, Dalia et Cristina, trois grandes dames qui ont fait aussi partie de ma famille durant le déroulement de mon travail de doctorat.

Je veux dire à mon mari Oscar et à mes enfants Jared, Mauricio et Diego, que je ne concevais pas cette expérience sans eux. Votre compagnie a donné un sens à cet effort si bien récompensé. Vos mots d'encouragement, les difficultés vécues ensemble et autant des réussites à fêter ensemble me comblent. Sachez que cette complicité si positive et si pleine d'énergie ne peut que laisser une famille unie et prête pour l'avenir. Je vous aime de tout mon cœur. Jared, ma fille, tu as été pour moi, en plus, une amie, un pilier… Comment te remercier ?

Oscar, mon mari, je me demande toujours comment j'ai fait pour te trouver… Quelle joie de vivre avec toi et quelle chance d'avancer à tes côtés ! Je t'admire car, dans mon cœur, j'ai conscience que seule ta compagnie fait ce que suis. Ces dix-huit ans de vie commune m'ont semblé passer très vite. Et après toutes ces années je réalise que nos pensées ne sont jamais loin les unes des autres. Je ne suis jamais toute seule, je compte toujours sur un grand homme, je compte sur toi Oscar …

Table des matières

Index des Figures

Index des Tableaux

1 Introduction

1.1. Des systèmes de communication de groupes sécurisés

Les nouvelles générations d'utilisateurs intègrent aujourd'hui dans leur vie quotidienne l'informatique comme un moyen pour accéder à autres services et comme un outil pour communiquer avec d'autres personnes. La vente et l'achat de biens par internet se popularisent. Les outils et les forums de dialogue dans des espaces réels ou virtuels sont de plus en plus couramment utilisés.

Avec cette évolution, la communication de groupes est un moyen d'échange dont les applications ont de plus en plus besoin. Dans notre monde actuel, l'évolution des systèmes informatiques a fait de la communication de groupes une technique nécessaire et très utilisée dans différents domaines.

Par exemple, dans le monde des jeux informatiques, il existe une gamme des jeux basée sur la participation d'un ensemble d'utilisateurs distribués physiquement dans des endroits différents. L'idée de mettre « ensemble » un groupe de personnes de façon virtuelle pour partager un temps d'amusement informatique implique, en arrière plan, la mise à disposition d'une plateforme de communication multipoint fiable. Avec ceci, l'industrie du jeu induit une évolution et une modernisation des autres technologies, y compris la communication de groupes.

Nous pouvons trouver un autre exemple de l'utilisation des communications de groupes dans le monde du spectacle, où l'on entend souvent parler de la diffusion des concerts par internet. Des diffusions multicast sont envoyées, seulement vers les membres qui ont payé l'accès au concert et qui auront le droit de regarder l'émission.

Le dernier exemple, l'utilisation de ce type de communication pour des dialogues protégés entre des groupes, donnera le cadre d'étude de ce mémoire. Il s'agit d'installer dans un ensemble de groupes une communication qui assure, entre autres, la propriété de confidentialité de l'information échangée à l'intérieur de chaque groupe. Les groupes doivent être aussi protégés vis-à-vis des possibles insertions de messages par des personnes extérieures au système. Seuls les membres autorisés dans la session peuvent apporter de l'information lire ou modifier celle échangée au sein du groupe.

Dans le monde des systèmes qui utilisent des communications sous forme de diffusion de groupes, le critère de performance devient un facteur aussi important que la sécurité. La difficulté d'assurer cette communication augmente avec la prise en compte des différentes exigences temporelles qui interviennent lorsque cette communication s'effectue dans des systèmes temps réel. La dynamique naturelle des membres actifs augmente aussi la complexité dans l'assurance des échanges entre eux.

Différents facteurs interviennent lors du développement d'un système de communication pour des groupes.

Nous proposons ainsi, dans ce mémoire, d'identifier les besoins généraux de ce type de systèmes, de concevoir une architecture de modélisation et de vérification des fonctions qui interviennent dans la communication de groupes vis-à-vis des attentes du système en termes de sécurité et de garanties temporelles.

1.2. Eléments d'un système de communication de groupes

Dans une application de groupes où la communication entre ses membres prend un rôle important, nous pouvons observer qu'il existe communément deux types d'information à échanger [LIV 2.5] : 1). L'information concernant l'application et 2). L'information concernant le bon déroulement de cette communication. Puisque les messages à échanger par l'application, dits « messages de données », doivent être sécurisés, nous avons besoin des éléments, dits « éléments secrets », qui serviront de base pour sécuriser cette information. Et puisque ces éléments risquent d'être changés, afin de maintenir leur sécurité tout au long d'une session active, nous aurons aussi besoin de messages, dits « messages de contrôle » qui serviront principalement à l'actualisation des éléments secrets.

En effet, dans la communication de groupes, nous devons à la fois assurer les tâches de communication intrinsèque de l'application, et effectuer le contrôle du protocole de communication.

Si nous considérons qu'il existe un seul canal de communication dans le groupe (ce qui peut se passer dans le pire des cas) nous aurons donc intérêt à optimiser l'utilisation du médium pour le transport des messages de données. Pour cela, plusieurs travaux de recherche [BAN 02], [WAN 05] portent leurs efforts sur la diminution du trafic de contrôle nécessaire au fonctionnement du protocole de communication, de façon à ne pas affecter les temps de réponse attendus par l'application. Cette première exigence, qui intervient très fortement dans la définition d'un système des groupes, correspond aux contraintes temporelles ou bien de performance établies lors de l'étape d'analyse de ce système.

Souvent, dans un ensemble de groupes, les supports réseaux dont les nœuds disposent deviennent aussi un facteur important. La disponibilité et la robustes des ressources réseaux sont deux paramètres qui permettent de restreindre plus ou moins les processus de contrôle impliqués dans la communication.

La composition des groupes liés par cette communication, ainsi que l'évolution de cette composition, est un paramètre déterminant pour les choix des mécanismes de protection de l'information échangée et des techniques d'actualisation des éléments secrets dont les mécanismes de protection ont besoin. Le choix des techniques pour la gestion des groupes se verra aussi affecté par cette composition et cette dynamique des membres.

En ce qui concerne l'application, le terme sécurité est pris en compte comme une personnalisation des communications de groupes. Protéger contre tout type d'attaque la communication d'un groupe de membres n'est pas forcément synonyme d'efficacité. Offrir un système plus rapide et plus performant que les autres n'est pas, non plus la meilleure solution si aucune sécurité n'est garantie. Il est évident que la sécurité dans la communication joue aujourd'hui un rôle

autour duquel les autres paramètres doivent souvent s'ajuster. Nous devons aussi prendre en compte que le niveau de sécurité peut varier selon l'application concernée.

Nous pouvons ainsi voir que plusieurs éléments participent à la détermination d'un système de communication de groupes sécurisés. Quelques axes de référence ont été déterminés et publiés lors des travaux faits dans ce domaine. Vous les trouverez dans la section suivante.

1.3. Des solutions pour la communication des groupes

C'est en 1976 que l'équipe composée par Diffie, Hellman et Merkle [DIF 76], [MER 78], [HEL 02] établit la technique de chiffrement de donnés « Diffie-Hellman » (DH), qui révolutionnera le concept des anciens systèmes pour la protection des données. Cette technique est maintenant la base de plusieurs systèmes, notamment dans le domaine de la communication de groupes.

L'idée de cette technique est de faire contribuer totalement ou partiellement chacun des membres dans une session active pour la génération des éléments de base qui seront ensuite utilisés par les techniques de chiffrement nécessaires à la communication.

A l'origine « Diffie-Hellman » établit un secret utilisé pour chiffrer et déchiffrer l'information échangée entre deux membres. En reprenant cet algorithme, [ING 82] l'adapte à des groupes, puis l'utilise dans des conférences multiutilisateurs (Conference Key Distribution System CKDS). D'autres extensions pour des groupes sont présentées dans [STE 90], [BUR 94], [STE 96], [STE 97], [STE 00], [ASO 00].

Avec l'apparition de grands réseaux et en ayant à l'esprit l'idée de minimiser le temps de traitement de la gestion du groupe, dans [MIT 97], [MOL 00] et [DU 99] les auteurs se basent sur la structure hiérarchique du groupe pour proposer un algorithme qui délègue le calcul de la clé aux nœuds composant le réseau.

En étudiant la même problématique de gestion des éléments secrets, nous trouvons des travaux qui utilisent des arbres pour la génération de clés : [WAL 98], [CAR 98], [NOU 98], [WON 98], [SHE 03], [CAN 99a], [DON 99a], [DON 99b], [DON 00], [KIM 00], [KIM 04a] et [ZOU 04]. Dans [BAN 02], l'utilisation de clusters représente aussi une bonne solution pour diminuer la charge de gestion de ces éléments secrets.

Les certificats sont aussi des éléments très utilisés dans le domaine de la communication de groupes. Dans [LIV 2.2], [LIV 2.3] et [LIV 2.4], les auteurs font une étude très complète qui concerne les certificats appliqués à la communication des groupes.

En suivant une autre direction, plus focalisés sur les services de gestion des groupes, les auteurs [BIR 90], [WHE 94], [DOL 96], et [MOS 95] cherchent à garantir la fiabilité des systèmes distribués en considérant principalement la tolérance aux fautes, l'ordonnancement des messages, la synchronisation dans la communication, etc. Des services pour la gestion des utilisateurs comme le *join* ou le *leave* sont ici inclus.

[REI 95], [KIH 98], [REN 96], [BIR 97] traitent, eux de leur côté, deux sujets innovants et importants, le premier traitant la sécurité dans la communication et le deuxième référant l'utilisation d'un système complet.

Pour finir, nous avons identifié un ensemble de travaux dont l'intérêt a été de répondre aux besoins des nouvelles technologies et des nouvelles techniques de développement des systèmes logiciels. Ainsi dans [HIL 97], [HIL 00], [GON

96], [McD 00], [IRR 03] et [AMI 05], les auteurs proposent des architectures liées à des plates-formes middleware, mises à la disposition des applications sous la forme d'entités micro protocolaires pour la communication des groupes avec des interfaces faciles d'utilisation. La possibilité que les applications puissent fabriquer leurs propres services, l'évolution dans la proposition d'architectures globales, et l'utilisation de nouvelles techniques dans le domaine de la sécurisation de la communication font partie des avancées apportées au domaine de la communication des groupes sécurisés. De façon plus précise, leurs travaux traitent la problématique du bon compromis entre le niveau de sécurité choisi (authentification, confidentialité et intégrité de la communication, protection contre le rejeu et la collusion, etc.) et la performance du système de communication.

L'ensemble de problématiques traitées dans la communication des groupes est tellement vaste qu'il a été impossible d'identifier un groupe ou un travail qui traite, dans son ensemble et au même niveau, l'ensemble des axes auxquels nous nous sommes intéressés.

1.4. Contributions apportées dans ce mémoire

La proposition d'une architecture pour un système de communication de groupes sécurisés, sa modélisation et sa vérification représentent une tâche hautement complexe. Notre intérêt a été d'instancier dans un cas réel une architecture proposée dans le cadre de ce mémoire et de montrer l'intérêt de l'application d'une méthodologie incrémentale de conception dans le cas d'architectures aussi complexes. L'objet final a été de montrer, avec la vérification de ce système, que le protocole de communication choisi satisfait les exigences établies à l'origine de sa définition.

Le travail présenté dans cette thèse intègre dans son architecture un ensemble représentatif des problématiques qui se présentent dans la communication de groupes sécurisés.

Nous présentons ci-dessous l'ensemble des apports de ce mémoire.

- Analyser des systèmes de communication de groupes en utilisant le profil UML 2.0 pour la définition des fonctionnalités intrinsèques de tels systèmes

Nous avons choisi de suivre une méthodologie [MOT 05], [MOT 06], [FON 06] qui nous aide à l'identification des fonctionnalités d'un protocole de communication de groupes mais aussi à l'implantation du modèle. Ainsi, en se basant sur certains diagrammes de la notation Unified Modeling Language (UML) 2.0, nous analysons et documentons les composants d'un protocole de communication de groupes. Les résultats de cette analyse peuvent servir à l'établissement de standards pour l'analyse et la définition de tels systèmes.

- Proposer une architecture de modélisation et de vérification de protocoles pour la communication sécurisée de groupes

Nous proposons l'architecture générique appelée SGCva (Secure Group Communication verification architecture) [MOT 06] [MOT 07], qui peut être utilisée dans différents contextes pour des protocoles de communication de groupes. Sa définition s'appuie sur les fonctionnalités précédemment identifiées dans ce type de communications. Le modèle de référence OSI (Open System Interconnection) [TAN 96] que nous considérons comme un cadre conceptuel adéquat pour concevoir les services et les protocoles de communications des systèmes distribués, a été la référence pour la structuration de notre architecture en couches. Les niveaux de déploiement de notre architecture s'inscrivent dans

les couches hautes du modèle OSI, c.à.d. dans les couches Application, Session et Transport du modèle OSI.

Globalement, nous proposons d'implémenter dans notre architecture les services concernés par : 1). Le transport sécurisé de l'information échangée entre les membres d'une session. 2). La gestion de la communication des groupes sécurisés. 3). La gestion du groupe. 4). L'application finale.

- Définir des primitives et des services pour des communications sécurisées de groupes

  L'identification des primitives de communication entre les couches d'une entité réseau implique un travail d'abstraction complexe. L'objet est de réussir à énumérer de façon exhaustive toutes les possibilités de dialogues entre les services des couches dans l'architecture. Nous avons identifié, tout d'abord, toutes les interactions existantes entre des couches. Puis, nous avons défini l'ensemble des échanges qui se déroulent entre les couches pour arriver, à la fin, à l'identification des primitives pour chaque couche. Un travail équivalent a été fait pour l'identification des « Unités de Données de Protocole » (PDUs), unités de données échangées entre entités équivalentes au sein d'une même couche réseau.

- Instancier notre architecture à travers une méthodologie incrémentale dans un projet industriel

  Le projet RNRT SAFECAST, dédié à l'étude, à la conception et à la validation de fonctionnalités de mécanismes garants de communications sécurisées à l'intérieur de groupes dynamiques [EAD 04], [SAFECAST], représente un cas réel et complexe de l'application d'un protocole sécurisé dans des communications de groupes. A travers l'application d'une méthodologie incrémentale [MOT 05], [MOT 06] de modélisation et de vérification, nous avons instancié notre architecture dans une modélisation du protocole Safecast [LIV 2.5] sous un environnement TURTLE [TURTLE]. Des résultats concernant la satisfaction ou la non satisfaction des exigences temporelles ont été fournis aux partenaires du projet.

  Notre contribution aux biens livrables de projet se trouve dans [LIV 2.5], [LIV 3.4], [LIV 4.1], [LIV 4.2], [LIV 4.3] et [LIV 4.4].

- Définir une nouvelle technique d'analyse de protocoles de groupes sécurisés

  Une contribution originale dans un tel domaine a été le fait de proposer une solution dont les qualités (fonctionnelles et de performances) auront été démontrées par des techniques de validation formelle. Le grand problème qui s'est posé, et qui à notre connaissance n'avait jamais été abordé, a porté sur la nécessité de traiter en même temps des problématiques qui, du point de vue de la modélisation et de la vérification formelle, avaient été adressées séparément dans la littérature, à savoir :

  (1) la sécurité dans la communication ;

  (2) la gestion de groupes dynamiques ;

  (3) les contraintes temporelles.

  Par rapport à ces trois problèmes de base, nous ajoutons plusieurs spécificités parmi lesquelles :

  (4) des groupes régis selon une organisation hiérarchique de ses membres ;

  (5) la combinaison étroite entre les exigences de sécurité et les exigences temporelles.

Le travail développé dans cette thèse ouvre des possibilités pour une nouvelle technique d'analyse des performances des protocoles de communication de groupes à travers une étude temporelle de ses fonctionnalités et de l'impact du renforcement de la sécurité vis-à-vis des performances effectives du protocole.

1.5. Organisation du mémoire

Le chapitre 2 présente l'état de l'art lié aux principaux axes d'étude dans le domaine de la communication de groupes. Le lien de cette présentation avec l'évolution historique, expose de façon progressive les sujets auxquels sont confrontés les protocoles de communication pour des groupes. Ce chapitre contient aussi une liste d'auteurs et d'architectures les plus représentatives du domaine traité.

Par la suite, les principaux éléments pour la sécurisation des communications des groupes sont présentés. Une combinaison de ces éléments est nécessaire pour garantir différentes propriétés de sécurité. Un bon équilibre dans cette combinaison est essentiel pour que la performance du système ne soit pas dégradée.

Nous nous focalisons ensuite sur les systèmes et sur les travaux qui s'approchent le plus des problématiques de recherche auxquelles nous nous sommes intéressés. Un tableau final fait une synthèse des problèmes à traiter ainsi que des solutions données pour la communication sécurisée des groupes.

Le chapitre 3 est dédié à l'analyse des fonctionnalités pour la communication de groupes sécurisés. Cette analyse a été conduite avec le profil UML 2.0. Comprendre le comportement de tels systèmes va nous aider à classifier et à énumérer les fonctionnalités nécessaires pour répondre aux besoins généraux des applications. Certaines fonctionnalités relèvent plus de la communication, d'autres de la sécurisation, et les dernières de la gestion des groupes. L'objectif que nous poursuivons est de définir un guide générique que l'on pourra appliquer à une large gamme d'applications. A la fin de cette description, nous définissons les ensembles d'acteurs nécessaires.

Le chapitre 4 propose une architecture générique en forme de couches qui peut être utilisée dans le cadre de l'analyse, de la modélisation et de la vérification de protocoles de communication de groupes sécurisés. La définition de cette architecture s'appuie sur les fonctionnalités identifiées dans le chapitre 3.

Nous avons cherché à proposer un canevas standard qui guide l'analyse et la conception de protocoles de groupes. Ce canevas s'est traduit sous la forme de l'architecture SGCva, qui organise les fonctionnalités que l'on peut trouver pendant le déroulement d'une session active pour des communications de groupes.

Le chapitre 5 montre l'instanciation de l'architecture présentée dans le chapitre 4 dans le cadre d'un système de communication de groupes spécifique. De façon plus précise, nous proposons une méthodologie de développement et de vérification des protocoles de communication que nous appliquons à notre architecture [MOT 05] dans le cadre du projet SAFECAST [SAFECAST].

Dans ce chapitre, nous effectuons aussi une analyse temporelle par couches. Nous extrayons pour chaque couche les résultats qui concernent la satisfaction ou la non-satisfaction temporelle des exigences énoncées par la spécification du

système SAFECAST. Nous avons cherché à établir, lors de cette modélisation et lors de la présentation des résultats, des standards généraux qui servent de base pour l'étude générique des protocoles de communication pour des groupes sécurisés. Comme résultat de cette vérification effectuée, nous donnons des précisions sur les points faibles du protocole proposé, ainsi que pour les points qui interviennent dans la non satisfaction des exigences temporelles établies pendant la phase d'analyse du système.

Le chapitre 6 conclut notre mémoire. Un bilan de nos contributions est présenté. Les perspectives de notre travail sont aussi exposées dans ce chapitre.

2 Etat de l'art

2.1. Introduction

Ce chapitre se consacre à une analyse du domaine très vaste de la communication de groupes. De ce fait, nous considérons en largeur une gamme de systèmes développés qui intègrent la notion de communication de groupes. L'identification des différentes problématiques traitées dans cette communication ainsi que les solutions proposées font aussi partie de nos objectifs.

Ce chapitre se compose de quatre sous-sections. Les sous-sections 2.1 et 2.4 correspondent à l'introduction et à la conclusion du chapitre. Nous exposons et nous expliquons les principaux éléments nécessaires à la sécurisation de la communication de groupes dans la sous-section 2.2. Ensuite, dans la sous-section 2.3, nous effectuons un historique des principaux systèmes existants en se focalisant sur leur évolution.

En ce qui concerne les principaux éléments de sécurisation, la sous-section 2.2 fait une étude des éléments les plus utilisés dans les communication de groupes, afin de : protéger les échanges des messages, veiller à l'authenticité des membres participant dans des groupes, assurer la non répudiation des membres face aux responsabilités qu'ils prennent au sein du groupe et finalement donner certaines capacités aux membres participants au travers de rôles.

La sous-section 2.3 est organisée selon deux grandes parties. La première partie expose les travaux visant à fiabiliser la communication de groupes. Des points comme l'ordonnancement, l'atomicité d'envoi des messages, la cohérence entre les vues des membres en communication sont abordés. Nous résumons dans cette sous-section les travaux qui sont devenus, pour la communication distribuée, les références essentielles.

La deuxième partie montre, à travers différents systèmes plus complets, comment un système de communication de groupes se compose de sous-systèmes qui, eux-mêmes, peuvent être assez complexes. La plateforme physique, la composition des groupes et l'application utilisatrice de cette communication sont trois facteurs importants qui personnaliseront le système, et qui engendreront une certaine complexité.

Finalement, nous synthétisons dans un tableau les principaux problèmes soulevés dans la communication de groupes, ainsi que les systèmes qui les ont traités.

2.2. Principaux éléments pour la sécurisation des communications

Dans la littérature, les travaux qui s'occupent de communications de groupes induisent souvent des besoins en sécurité. En effet, chaque fois qu'un membre communique dans un groupe, il peut s'avérer nécessaire de garder confidentielle l'information échangée au sein de ce groupe. L'authentification et l'intégrité des messages sont aussi des propriétés qu'un système de communication pour des groupes doit garantir aux applications qui l'utilisent. En effet, assurer qu'un

message dans le réseau vient bien d'un utilisateur autorisé (authentification) est, en général, une propriété importante à garantir [ZOU 07]. De la même façon, garantir que chaque message dans le réseau n'a pas été modifié malicieusement est essentiel (intégrité).

Le niveau de sécurité à garantir pendant le déroulement d'une session peut varier fortement d'une application à l'autre, mais aussi peut varier pendant le déroulement d'une même application. Par exemple, dans certains contextes, la tâche d'authentifier la source du message peut être la tâche la plus importante. Pour d'autres, la confidentialité des messages est la priorité.

Aujourd'hui, la cryptographie répond bien aux besoins des utilisateurs en matière de sécurité. Elle représente une bonne solution pour la protection des données [GOL 01], [KER 06]. En partant de cette base, des techniques diversifiées sont disponibles pour aboutir à une sécurité plus ou moins complète selon les besoins de l'application qui l'utilise [BAG 06], [ÖNE 07].

D'autres sujets connexes sont en liaison avec la communication des groupes sécurisés (CGS) : le réseau de déploiement physique, la taille des groupes et le comportement dans la composition des groupes participants. Ces paramètres personnalisent les systèmes de communication de groupes.

2.2.1. Cryptographie à clé publique

La cryptographie asymétrique, dite aussi à clé publique, a été présentée pour la première fois à la National Computer Conference en 1976, puis publié quelques mois plus tard dans New Directions in Cryptography par Whitfield Diffie, Martin Hellman and Ralph Merkle [DIF 76]. C'est en 1978 que le premier exemple a été publié avec le système RSA, par Ronald Rivest, Adi Shamir et Leonard Adleman [RIV 78]. L'abréviation RSA est tirée des trois noms de ses auteurs. Le principe sommaire de la cryptographie à clé publique est de créer un couple de clé publique et privée. Chaque clé a une relation logique avec la clé duale. La clé privée est gardée confidentielle par l'entité qui a généré le couple de clés. La clé publique est envoyée à l'extérieur et est accessible à tout le monde. La solidité de ce principe vient du fait qu'il est actuellement « impossible », du moins en temps de calcul raisonnable, de déduire la clé privée à partir de la connaissance de la clé publique et de messages codés au travers de ce couple de clés [ELL 03].

Cette découverte a modifié la cryptographie car jusqu'alors, pour envoyer un message chiffré, les deux correspondants devaient se mettre d'accord pour échanger au préalable une information secrète : la clé de sécurisation. La difficulté de cette tâche affaiblissait les algorithmes précédents.

La cryptographie asymétrique est incontournable pour garantir des propriétés de non répudiation. Elle reste utilisée par des algorithmes pour générer des signatures. Une empreinte de taille fixe est obtenue [LIV 1.4] par l'application d'une fonction de hachage à sens unique (avec faible probabilité de collision) au message à envoyer. Cette empreinte est ensuite chiffrée par un algorithme asymétrique en utilisant la clé privée de l'émetteur. Le résultat obtenu correspond à la signature. Elle est ajoutée au message à envoyer. Seule la personne possédant la clé privée pourra signer le message. Ceci permettra de garantir la non répudiation de la source.

En utilisant la clé publique de la personne qui présume avoir envoyé le

message, le récepteur déchiffre la signature, calcule l'empreinte du message avec une fonction de hachage. Il effectue ainsi, la vérification de l'intégrité du message reçu ainsi que de sa provenance. L'émetteur ne peut pas nier qu'il a bien émis ce message.

La cryptographie à clé publique a permis de résoudre des nombreux problèmes pratiques mais elle reste moins utilisée que la cryptographie à clé secrète pour faire communiquer des groupes. La mise en place d'infrastructures à clé publique (PKI) ne s'applique pas bien au cas d'un ensemble d'utilisateurs qui souhaitent faire une même communication dans un groupe de N membres. Utiliser une paire de clés (la clé publique et la clé privée) pour chaque couple de membres qui échangent de l'information introduit une complexité combinatoire exponentielle en 2^n dans le nombre de clés, ce qui est difficile à gérer pour des groupes de grande taille. L'utilisation de cette technique devient coûteuse dans les communications de groupes car elle oblige à effectuer N transmissions point-à-point en lieu d'une seule émission multipoint et broadcast.

Au final, les algorithmes de cryptographie asymétrique ne supportent pas le passage à l'échelle [ZOU 05] pour des émissions multipoint et broadcast. Les algorithmes basés sur la cryptographie symétrique supportent ce passage à l'échelle. Ils offrent de bien meilleures performances et ne requièrent pas l'utilisation d'une infrastructure spécifique.

2.2.2. Cryptographie à clé secrète

La cryptographie à clé secrète, dite aussi chiffrement symétrique, fonctionne selon le principe suivant : pour effectuer une communication de groupe de manière sécurisée, l'information échangée doit être protégée en utilisant une clé symétrique qui intervient dans le chiffrement et le déchiffrement des messages. Cette technique représente une solution dès lors qu'une clé commune (la clé secrète) à tous les utilisateurs d'un groupe en communication, est générée et partagée pour chiffrer et déchiffrer l'information échangée entre eux.

L'algorithme le plus utilisé pour procéder à cette génération et ce partage de clé secrète est l'algorithme de Diffie Hellman [DIF 76], [MER 78]. Son principe de fonctionnement est le suivant : deux membres *m1* et *m2* voulant communiquer des informations secrètes, génèrent chacun de leur côté, une pair d'éléments (*a1* et *b1* pour *m1*, *a2* et *b2* pour *m2*). Les deux membres échangent entre eux les éléments *b1* et *b2* qui fonctionnement comme des clés publiques. *m1* garde secret l'élément *a1*, *m2* garde secret l'élément *a2* qui fonctionnement comme des clés privées. Les deux membres peuvent ensuite générer une clé commune *cm* issue de la combinaison de *b1* et *b2*. *a1* et *a2* restent privées pour chaque membre.

Dans le cas de communications de groupes, la clé secrète commune *cm* sert à chiffrer et à déchiffrer les messages échangés.

L'algorithme initial de Diffie Hellman s'est avéré sensible aux attaques de type « Man-in-the-middle », dans lesquelles un intrus se place entre les deux membres concernés. Cet intrus se fait passer pour l'autre membre auprès de chacun des membres. En entrelaçant ses comportements de façon astucieuse auprès des deux membres et en ajoutant des informations dans la partie non chiffrée du message, l'intrus se rend maître des communications entre les membres initiaux qu'il peut alors déchiffrer. Pour pallier ce problème, cet

algorithme a été rectifié par Gavin Lowe [LOW 96], en introduisant l'identité des membres participants aux contributions pour la génération de la clé. Ce nouvel algorithme est détaillé dans [HEL 02].

L'algorithme de Diffie Hellman a été étendu pour s'appliquer à N membres. Cette extension est référencée dans la section 2.2.2.2

Le tableau 2.1 montre de façon chronologique et synthétique les travaux qui ont abouti aux protocoles actuels utilisés dans le domaine de la communication de groupes sécurisées. Ces travaux sont détaillés dans cette sous-section. Les différentes problématiques qui apparaissent au fur et à mesure que les applications des groupes évoluent, traités au travers d'améliorations des algorithmes de sécurité, sont les suivantes. Certaines cherchent à diminuer la charge en temps de calcul ou en nombre de messages échangés que l'algorithme proposé nécessite. Le calcul de la clé de session peut être plus ou moins compliqué en fonction des différents paramètres intervenants pendant ce processus de calcul. La façon d'envoyer les contributions des membres impliqués dans le processus peut aussi varier. Ces derniers peuvent, par exemple, diffuser les informations selon une logique broadcast ou bien en établissant un anneau logique circulaire de distribution séquentielle.

Auteur	**Ingemarsson**	**Steer et al.**	**Fiat and Naor**	**Burmester and Desmedt**	**Steiner et al.**	**Steiner et al.**	**Steiner et al.**
Date de la proposition	**1982**	**1990**	**1994**	**1996**	**1996**	**1997**	**2000**
Centralisée			**x**		**x**	**x**	
Distributive / Contributive	**x**	**X**		**x**	**x**	**x**	**x**
Variation de "n-partyDiffie-Hellman key exchange"	**x**	**X**		**x**	**x**	**x**	**x**

Tableau 2.1. Les protocoles symétriques de base pour la communication de groupes sécurisés

La majorité des références dans le tableau 2.1 basent leur technique sur le principe de contribution et de distribution des secrets entre les membres lors du processus de génération de la clé de session. Ce principe cherche à impliquer les membres actifs dans les processus de génération et de distribution des clés. Fiat et Naor, par contre, proposent un algorithme de nature centralisée. D'autres auteurs (Ingemarsson, Steer et al., Burmester et Desmedt) proposent des méthodes distributifs / contributifs. En 1996 et 1997, Steiner et al. proposent une méthode hybride dans le sens où ils combinent l'utilisation d'entités centrales pour effectuer certaines opérations de calcul avec la délégation d'une autre partie du calcul chez les membres de façon distribuée. Une amélioration des travaux de Steiner et al., en 2000, propose un algorithme distributif.

Les principaux apports de ces algorithmes se focalisent sur les axes suivants :

- La structure du groupe pour la génération et la distribution des clés
- La technique de génération des clés

- La distribution d'information (1-vers-n ou n-vers-n)
- Le type de sécurité offert (inconditionnel secure, computationnel secure ou résistant jusqu'à k adversaires k-resilient schemes)

2.2.2.1 Gestion centralisée de la clé de session

Les méthodes centralisées répondent bien aux besoins des applications avec un grand nombre de membres.

Dans [FIA 94], Fiat et Naor proposent le traitement de la communication de groupes en utilisant une entité centrale pour la gestion de la clé. Cette entité doit donner la clé à tout membre qui rejoint le groupe. Elle se charge aussi d'actualiser cette clé, si besoin est. Leur étude montre que le système proposé résiste bien aux attaquants passifs. L'intérêt de mettre cette entité centrale est de diminuer la charge de gestion de la clé chez les membres du groupe. Le principal inconvénient de cette technique est bien entendu l'installation de cette entité centrale à laquelle on doit faire confiance pour la gestion de la clé du groupe.

2.2.2.2 Gestion contributive de la clé de session

La génération de la clé sécrète à travers un algorithme contributif apporte un degré de sécurité bien supérieur. L'algorithme Diffie-Hellman, grâce à son principe de fonctionnement sur des contributions et sur l'accord final d'une clé partagé entre deux membres participants, à été étendu pour son utilisation dans des sessions multiutilisateurs sous le nom de « n-party Diffie-Hellman key exchange ». Au cours du temps, de nombreuses modifications ont été faites au protocole de base ; des extensions sont présentées en [STE 96], [ASO 00].

Cette clé est générée avec l'intervention des membres appartenant au groupe. Elle reste seulement accessible par ces membres pour chiffrer et déchiffrer les messages échangés. C'est la raison de l'importance de cette gestion de la clé de session de façon contributive, sujet cœur dans la Communication de Groupes Sécurisée.

Dans la gestion des clés, deux problématiques doivent être traitées. La première correspond à la génération de la clé de session et la deuxième à la distribution de cette clé. Ayant une étroite relation entre eux, les deux problèmes sont, en général, traités dans une même proposition d'algorithme de groupes. Dans certains protocoles de génération de clés, la distribution de la clé de session est faite pendant sa génération. Dans d'autres, la clé est générée en premier, sa distribution s'effectue après.

Ingemarsson et al. [ING 82] est l'un des premiers auteurs qui a contribué aux travaux de recherche dans la communication des groupes. Dans leur article, ils font référence à la technique « Diffie-Hellman-Merkle key exchange » comme un système intelligent mais qui sert juste à deux utilisateurs. Dans [ING 82], les auteurs font une généralisation de ce système à un groupe pour la génération d'une clé pour le cas d'une conférence (Conference Key Distribution System. CKDS). Dans CKDS un ensemble de stations peut s'accorder sur une clé secrète pour crypter et décrypter leurs informations échangées. Les membres participants sont connectés selon un anneau logique pour composer la nouvelle clé avec la contribution de chacun des membres. A son tour, le membre *i* reçoit la clé envoyée par le membre *i-1.*Il fait sa contribution dans la clé reçue, puis il envoie cette clé au membre *i+1*. Le tour commence avec l'utilisateur *0*. Il finit

avec l'utilisateur *M-1, M* étant égal au nombre de stations qui sont membres de la conférence.

Ingemarsson et al. montrent que leur système est sûr vis-à-vis d'un attaquant appelé « Multitap », qui, à l'écoute des messages pendant plusieurs cycles d'établissement de la clé, ne peut pas deviner la clé de session à un moment donné. Leur système offre la possibilité de choisir le niveau de résistance au « Multitap ». Ils montrent aussi, de la même façon que [DIF 76], que leur système sert, à travers l'utilisation des signatures, à authentifier la source des messages.

Des travaux postérieurs à Ingemarsson continuent à proposer des améliorations à « Diffie-Hellman-Merkle key exchange ». En suivant la philosophie de faire contribuer les membres participants et de distribuer la tâche de génération de la clé de communication dans les membres du groupe, nous trouvons notamment les travaux de Steer et al. [STE 90], Burmester et Desmedt [BUR 94] ainsi que Steiner et al. [STE 96], [STE 97], [STE 00], détaillées par la suite. Leurs études cherchent aussi à rendre les systèmes de communication efficaces pour des applications de groupes.

Steer et al [STE 90] proposent des avancées dans le domaine des téléconférences. Ils proposent une technique pour protéger les participations de chaque membre ainsi que les clés utilisées pendant cette communication. Jusqu'alors, les téléconférences faisant intervenir un ensemble de membres, étaient proposées pour des environnements ouverts. Steer et al utilisent un point de passage de données qu'ils ont appelé « bridge », lequel est connecté à tous les utilisateurs en communication. Ce point coordonne les participations des membres, l'information qui passe est protégée. La contribution de ce travail a été de distribuer les tâches de génération de la clé du groupe ainsi que le chiffrement et le déchiffrement des participations des membres. Ils offrent, avec cette distribution, la possibilité de choisir entre différents algorithmes de chiffrement.

Burmester et Desmedt critiquent les protocoles [ING 82], [FIS 92], [BLU 93] et [KOY 87] car soit ils ne sont pas efficaces au niveau des calculs effectués, soit ils ne sont pas sécurisés. Burmester et Desmedt [BUR 94] présentent une extension de l'algorithme Diffie-Hellman dans un protocole d'échange de clé de groupe appelé BD. Ils traitent la problématique de passage à l'échelle en proposant un protocole qui prend juste deux tours pour effectuer le calcul de la clé, le nombre des opérations de calcul des exponentielles est constant. Au niveau sécurité, Burmester et Desmedt testent la robustesse de leur protocole contre des adversaires passifs. Leur algorithme supporte aussi la confidentialité post-résiliation.

Steiner, Tsudik et Waidner se basent sur le protocole Diffie-Hellman. Ils publient deux nouveaux algorithmes [STE 96] dans un système appelé Cliques [STE 97] pour considérer la dynamique des groupes. Dans [STE 96], Steiner et al. présentent une extension du protocole Diffie-Hellman (deux personnes) à *n* utilisateurs en incluant la démonstration sur la sécurité de cette généralisation. Dans leur système, ils proposent des améliorations importantes aux travaux publiés autour de Diffie-Hellman. De ce fait, ils proposent des algorithmes qui n'ont besoin ni d'ordre ni de synchronisation des messages pour l'échange de la clé du groupe. Des sujets comme la diminution des étapes de calcul pour les fonctions de calcul exponentielles et du nombre de messages nécessaires, sont aussi traités. Une bonne synthèse et analyse de la problématique et des

solutions des algorithmes contributifs et distributifs est abordée. Dans [STE 97], les études sont étendues à la problématique de la dynamique des groupes avec le système Cliques. Par conséquent, la garantie des propriétés est faite en relation avec cette composition dynamique. Déjà ils spécifient la confidentialité, l'intégrité, l'authentification de la source, la non répudiation de la source et le contrôle d'accès comme des sujets d'intérêt pour les nouvelles applications. Cliques divise les opérations de gestion de la clé en deux : celles pour la génération initiale de la clé, et celles pour l'actualisation vis-à-vis des mouvements des membres dans le groupe. Puisque la génération initiale se fait avant toute communication de groupe, le choix de l'algorithme initial générateur de la clé du groupe est moins restreint. On peut donc utiliser, un algorithme contributif et distribué qui consomme plus de tours de calcul que dans l'actualisation en ligne. Pour l'actualisation en ligne, en utilisant un contrôleur central, Cliques diminue la charge de calcul de la clé du groupe. Ce contrôleur offre des services de gestion de clés qui fonctionnent selon le même principe contributif que ceux du protocole Diffie-Hellman, en réponse aux changements de membres dans un groupe. Des opérations de fusion et de séparation sont aussi prises en compte par le système Cliques.

Dans [STE 00], Steiner et al. proposent l'application du protocole Cliques pour des groupes de pairs qui répondent bien aux besoins sécuritaires vis-à-vis de la composition changeante du groupe. Cliques n'est pas applicable à des grands groupes à cause de sa technique de génération contributive de la clé de groupe : les algorithmes proposés sont de complexité exponentielle par rapport à la taille du groupe. L'algorithme OFT, par exemple, basé sur des fonctions à sens unique appliquées à des arbres, requiert un nombre logarithmique d'opérations de calcul. OFT proposé dans [SHE 03] a bien fonctionné pour des groupes de plus de 10 000 000 participants.

Le Tableau 2.2 synthétise les caractéristiques des travaux énoncés dans cette sous-section, travaux tous basés sur Diffie-Hellman. Les critères énumérés détaillent l'efficacité du calcul et de la distribution de la clé de session dans les systèmes proposés.

Extension du protocole « n-party Diffie-Hellman key exchange »						
Nom du Protocole	**ING**	**STR**	**BD**	**GDH.1**	**GDH.2**	**GDH.3**
Auteurs	**Ingemarsson et al**	**Steer et al.**	**Burmester and Desmedt**	**Steiner et al.**	**Steiner et al.**	**Steiner et al.**
Année	1982	1990	1996	1996	1996	2000
Tours	n-1	2	2	2(n-1)	n	n+1
Nombre total de messages	n(n-1)	2(n-1)	2n	2(n-1)	n	2n
Synchronisation	V	F	V	F	F	F
Sérialisation	V	V	V	V	V	V

Tableau 2.2. Evolution des algorithmes basés sur Diffie-Hellman

La troisième ligne du Tableau 2.2 énumère les auteurs des travaux. La quatrième ligne indique l'année de parution de ces travaux. Les lignes suivantes donnent les critères de classification retenus [ZOU 05]. Les protocoles de ce tableau font contribuer de façon différente les membres du groupe qui sont impliqués dans le processus de génération de la clé du groupe. La ligne *Tours* donne le nombre de tours que l'algorithme doit effectuer pour effectuer le calcul. La ligne *Nombre total de messages* est le nombre total de messages échangés pendant tout le calcul. La grande majorité d'entre eux utilisent une technique de sérialisation qui s'appuie sur le numéro de série des membres participants dans la génération de la clé. Certains d'entre eux demandent de la synchronisation des messages pour que les opérations pour le calcul de cette clé s'exécutent dans un ordre spécifique, ce qui est coûteux en temps et en volume de messages dans le cas de grands réseaux.

L'inconvénient de ces types de protocoles est l'augmentation du nombre d'opérations effectuées. Ceci implique qu'il est difficile d'appliquer ces protocoles à des groupes très grands.

Pour lutter contre ce problème, en se basant sur la structure du groupe, trois publications apportent des solutions algorithmiques :

Mittra [MIT 97], Molva et Pannetrat [MOL 00] et Du et al. [DU 99] présentent respectivement les systèmes IOLUS, RPS et STB. Ils répartissent les membres dans des sous-groupes organisés selon une structure hiérarchique [ZOU 05]. Dans [MOL 00], chaque sous-groupe a un administrateur désigné et une clé du sous-groupe administrée par cet administrateur, lequel est aussi en communication avec son groupe père à travers la clé de ce sous-groupe. Le/les administrateur(s) des sous-groupes peut/peuvent être ou non membre du groupe. Mittra [MIT 97] et Du et al. [DU 99] font confiance à leurs administrateurs, Molva et Pannetrat [MOL 00] non. Faire confiance à un certain administrateur veut dire qu'il détient toutes les clés du groupe et qu'il comprend l'information échangée. Dans le cas contraire, il ne possède que les clés qui servent à transmettre les messages du protocole dans lesquels il intervient. Il ne peut pas comprendre toute l'information qui transite.

Les systèmes RPS et STB se basent sur l'utilisation de clés publiques alors qu'IOLUS se base sur l'utilisation de clés secrètes.

Le système RPS [MOL 00] fait confiance aux nœuds intermédiaires dans l'arbre des sous-groupes, nœuds qui ont pour but d'assister en enregistrant, en contribuant et en faisant transiter les contributions des membres pendant la gestion des clés des sous-groupes. Dans [MOL 00], Molva et Pannetrat proposent un protocole qui intègre la dynamique des groupes, la gestion des clés et la confidentialité des échanges. Un des principaux apports de ce protocole est la délégation du calcul de la clé vers les nœuds intermédiaires du réseau. Une dépendance est créée entre l'appartenance aux groupes et la topologie du réseau. Ils assurent ainsi le passage à l'échelle d'un protocole pour des groupes dynamiques mais aussi la sécurité des membres en communication. Pour chaque changement d'utilisateur dans les groupes, seules les modifications locales qui correspondent au sous-groupe du membre, sont effectuées. Si une clé est compromise pendant l'exécution de l'algorithme de renouvellement partiel de cette clé, les autres clés correspondant aux autres parties du groupe ne seront pas compromises. La confidentialité des échanges est faite avec des séquences chiffrées « *Cipher Sequences (CS)* ».

Le problème de RPS est que chaque émetteur dans la structure doit avoir

connaissance du chemin à suivre pour envoyer des messages aux nœuds intermédiaires et aux récepteurs finals. Ces émetteurs supposent que ce canal est sûr. Le schéma suivi s'applique à des applications 1-vers-tous, et le schéma tous-vers-tous dans RPS n'est pas possible.

STB traite les envois unicast et multicast. La communication s'effectue au travers de routeurs qui utilisent des clés publiques et privées. Son problème principal est dû au besoin d'un routeur final qui doit connaître toute la configuration du groupe et des sous-groupes afin de pouvoir chiffrer, déchiffrer et bien renvoyer les messages qui circulent dans le réseau.

IOLUS, fait partie des protocoles à chiffrement symétrique. Il divise aussi le groupe en sous-groupes, afin de gérer une clé par sous-groupe et d'actualiser juste la clé du sous-groupe affectée par des mouvements en son sein. Les sous-groupes sont structurés dans des arbres. Chacun des sous-groupes a un contrôleur pour la gestion de la sécurité. Ce contrôleur est en charge de la communication entre les membres de son groupe et son groupe père. Il gère aussi la clé du sous-groupe.

2.2.2.3 Gestion hybride de la clé de session : centralisée et contributive

Park et al. [PAR 04] font partie des équipes qui proposent l'utilisation des deux techniques dans une même solution pour profiter des avantages mutuels de chacune. Un avantage d'utiliser des algorithmes symétriques est son passage à l'échelle facile. Un avantage des algorithmes asymétriques est qu'ils présentent des garanties de sécurité supérieures à celles des algorithmes symétriques.

Dans [PAR 04], les auteurs présentent une méthode de génération et de distribution participative des clés pour réduire le temps et la charge du réseau. Ils proposent, aussi, l'introduction de deux serveurs, un pour la gestion des clés et l'autre pour l'authentification et la gestion des membres. Ensuite, un mécanisme de proxys est introduit pour réduire le trafic généré entre les membres qui participent à la génération des clés et le serveur de gestion des membres. Leur étude observe le comportement temporel des protocoles, ainsi que la charge réseau pendant le renouvellement de la clé de session.

Park et al. adoptent les hypothèses suivantes :

- Un réseau du type multicast qui utilise le protocole IGMP (Internet Group Management Protocol) de la pile des protocoles IP
- Un maximum de cinq niveaux de routeurs à traverser pour faire communiquer les membres finals dans le réseau
- Seize modèles de distribution des membres dans le réseau ont été utilisés [KAM 99]
- Les valeurs de la fonction de distribution vont de 3700 à 7000 quand le nombre des nœuds est égal à 1500

Les auteurs comparent les méthodes de renouvellement de clés centralisés conventionnelles [HAR 97a], [HAR 97b], [OKA 00] avec leur méthode hybride. Ils montrent une diminution de 1% à 10% du temps utilisé pour la distribution de la clé de session avec l'application de leur méthode. En ce qui concerne le trafic réseau, ils montrent que, avec les méthodes conventionnelles, l'administrateur des groupes transmet/reçoit au total 3060 paquets pendant un renouvellement de clés. Avec leur nouvelle méthode et en fonction du taux d'erreur de transmission, l'administrateur des groupes transmet/reçoit un total de paquets simplement compris entre 9 et 316.

2.2.2.4 Utilisation des arbres pour la génération de clés

Wallner et al. [WAL 98], Caronni et al. [CAR 98], Noubir [NOU 98], Wong et al. [WON 98], Sherman et McGrew [SHE 03] et Canetti et al. [CAN 99a] ont dirigé leurs efforts vers la création d'algorithmes qui génèrent des clés structurées par des arbres. Cette approche semble plus puissante que les algorithmes centralisés, distribués ou hybrides discutés précédemment.

Ce type de schémas peut s'utiliser dans des communications 1-à-tous ou bien tous-vers-tous, au travers d'une gestion centralisée ou bien distribuée et contributive.

En ce qui concerne des algorithmes centralisés, le protocole proposé par Wallner et al. [WAL 98] est la base de cette famille de protocoles. Il se nomme *Logical Key Hierarchy, LKH* [SHE 03]. Des protocoles similaires ont été proposés par Caronni et al. [CAR 98], et par Noubir [NOU 98]. Ces protocoles s'appuient sur un arbre binaire appelé *key tree* ou *Logical Key Hierarchy.* Le groupe multicast est géré au moyen d'un arbre virtuel par un contrôleur de groupe. Les feuilles dans l'arbre correspondent aux membres du groupe, les nœuds intermédiaires sont des clés.

La racine de l'arbre correspond à la clé pour le chiffrement des données du groupe (*Traffic Encryption Key, TEK*). Les autres clés sont des clés de chiffrement des clés (Key Encryption Key, KEK). Un membre dans le groupe connaît toutes les clés qui se trouvent dans le chemin entre la racine de l'arbre et ce membre. La seule clé connue par tous est donc la racine. C'est la clé qui sert à chiffrer les messages dirigés vers tous les membres du groupe. Lors de toute entrée ou sortie d'un membre dans le groupe, les clés connues par lui seront changées.

Afin de diminuer les niveaux possibles dans un arbre, Wong et al. [WON 98] proposent l'utilisation des arbres *d-array,* où un nœud de l'arbre peut avoir *d* nœuds « enfants ». Un arbre dans ce schéma est composé, comme dans les arbres binaires, de clés dans ses nœuds intermédiaires et de membres dans ses feuilles. La racine contient la clé de communication du groupe. Un certain membre connaît les clés dans le chemin qui le lie jusqu'à la racine. Lors de tout mouvement d'un membre dans le groupe, les clés qu'il connaît seront changées. Il existe trois différentes techniques d'actualisation des clés : celle orientée membre, celle orientée clé et celle orientée groupe. Chacune de ces techniques précédentes fait que le nombre d'opérations de chiffrement et/ou le nombre de messages d'actualisation des clés diminue. La dernière technique est celle qui répond le mieux aux besoins de passage à l'échelle.

Sherman et McGrew [SHE 03] proposent le schéma *One-way Function Tree* (OFT), dont l'avantage est de diminuer de moitié le nombre de messages envoyés aux membres du groupe, lors de l'actualisation de la clé du groupe.

La structure d'arbre d'OFT est la même que celle de LKH. L'hypothèse de l'existence des canaux sécurisés entre le contrôleur et chaque membre dans le groupe est prise.

Canetti et al. [CAN 99a] ont réalisé une variante de la technique OFT [SHE 03]. Cette variante s'appelle « One-way function Chain » (OFC). Le mot *Chain* vient des relations qui existent entre la série de clés qui se trouvent sur le chemin qui part de la racine de l'arbre et qui va vers chaque membre du groupe. La relation entre les clés de cette chaine est liée à l'application d'une fonction à sens

unique. La simplicité d'OFC vis-à-vis d'OFT vient de l'association de juste deux valeurs à chaque nœud dans l'arbre. OFC attribue à trois valeurs à chaque nœud : (*a) Un secret du nœud* et (*b) La clé du nœud*. La troisième valeur qu'OFT associe à chaque nœud est *un secret appelé « secret blindé »* Un *secret blindé* est appelé ainsi car, pour être transmis, il est chiffré avec la *clé du nœud.* Ce secret blindé est utilisé pour produire le *secret* du nœud. Ce dernier est utilisé pour que le nœud produise sa *clé du nœud*.

Dans [HOR 02] et [KU 03], OFT a été démontré peu sûr face à des attaques de collusion. Ainsi, dans [KU 03] un nouveau schéma a été proposé ; il empêche d'attaquer le système pour sécuriser les systèmes de communication contre la collusion tant pré-adhésion que post-résiliation. Cependant, la nouvelle modification implique d'effectuer plus d'opérations de chiffrement : l'avantage que OFT avait gagné vis-à-vis de LKH disparaît.

En ce qui concerne des algorithmes distribués, des nouvelles solutions pour la gestion de clés ont été proposées [KIM 00], [KIM 04a], [ZOU 04]. Dans [KIM 00] et [KIM 04a], le protocole « 2-party Diffie-Hellman » à été étendu pour être utilisable pour un schéma d'échange de clés multiutilisateurs et structuré dans des arbres : le schéma TGDH. La critique faite à ce schéma vient du fait que, lorsqu'une procédure de renouvellement de clé commence, la communication de l'application doit s'arrêter, et elle recommence quand la clé a été renouvelée. Dans [ZOU 04], les auteurs mettent à disposition des groupes un protocole qui permet de continuer les communications applicatives même si un changement d'un membre se présente et un processus de renouvellement de clés est déclenché. Le protocole DISEC représente aussi une solution distribuée dans laquelle les auteurs proposent d'enlever le contrôleur du groupe [DON 99a], [DON 99b] et [DON 00].

2.2.2.5 Les clusters

Dans [BAN 02], un schéma de gestion de clés à base de clusters a été proposé. Les clusters sont une façon de regrouper des membres dans un groupe. La structure logique qui en résulte comprend des sous-groupes appelés clusters, qui sont aussi rangés dans différents niveaux hiérarchiques, nécessaires pour la génération des clés. Un des principaux apports de l'utilisation des clusters est l'efficacité des processus impliqués dans la gestion de clés. Les clusters représentent une solution pour la diminution de temps d'exécution et l'utilisation de bande passante pendant la génération des clés.

Les membres sont organisés dans des clusters de niveau 0, 0 étant le niveau le plus bas. Chaque cluster élit un leader parmi ses membres. Tous les leaders élus passent au niveau 1, niveau juste au dessus du niveau 0.

Le niveau 1 est aussi divisé en clusters et un nouveau leader est élu pour chaque cluster. Les leaders du niveau 1 passent au niveau 2, niveau juste au dessus du niveau 1.

Le processus continue jusqu'au moment où l'on ne trouve plus qu'un seul leader dans un niveau.

Un ensemble complet de clés est octroyé à chaque membre du groupe. Une clé de niveau, gérée par un contrôleur/serveur, une clé du cluster, générée par le leader et partagée avec les membres du cluster, et une clé secrète, partagée entre chaque membre du cluster et le leader de son cluster. Tout membre qui appartient à deux niveaux possède deux ensembles de clés.

Le support de ce type de structure fait intervenir différents éléments attribués

au traitement des clés mais aussi des clusters. Le « Clustering Protocol » est un des éléments les plus importants. Il sert à maintenir cohérente la structure des clusters dans chaque niveau. Il réagit en fonction de tout mouvement d'un membre. Des opérations de fusion et de séparation de clusters font partie de ce protocole. Une relation des clés entre les différents niveaux existe. Le changement des clés d'un certain niveau génère aussi des actualisations dans des niveaux inférieurs.

Après chaque mouvement de membres dans le groupe, des actualisations sont appliquées aux clés. L'algorithme pour la gestion des clusters est assez complexe car il est basé sur les niveaux existant dans la structure des clusters.

Certains mouvements impliqueront l'actualisation de beaucoup des clés pendant que d'autres obligeront à l'actualisation de moins de clés. Par exemple, à toute entrée d'un membre, la clé du cluster où le membre sera affecté est renouvelée par le leader et envoyée aux membres de ce cluster y compris le nouveau membre. La clé du niveau 0 *CN0*, qui appartient à tous les membres du niveau 0, doit être aussi actualisée par le serveur des clés puisque le nouveau membre entre dans le niveau 0. Cette clé est envoyée à tous les membres à travers les leaders des clusters dans le niveau 0 (chaque membre reçoit la clé *CN0* chiffrée avec sa clé du cluster). L'algorithme exhaustif est détaillé dans [BAN 02].

Un réseau Internet avec 280 000 routeurs est modélisé dans [BAN 02]. En faisant des simulations, ils mesurent l'*overhead* de leur protocole et ils le comparent à celui d'autres schémas. Ils montrent que l'utilisation des clusters donne un meilleur résultat en termes d'utilisation de bande passante, de diminution de processus de calcul impliqué et de stockage intermédiaire utilisé.

2.2.3. Certificats

L'authentification et le contrôle d'accès des membres dans une session seront gérés par des techniques de certificats. Un certificat est un mécanisme basé sur l'utilisation de clés publiques.

D'ailleurs [LIV 1.4], selon la Délégation Interministérielle pour la Sécurité des Systèmes d'Information (DISSI) dissoute en 1996 [REC 901/DISSI/SCSSI], lorsque l'information est échangée, « l'intégrité s'étend à l'authentification du message, c'est-à-dire à la garantie de son origine et de sa destination ».

De plus, d'autres services comme la prévention contre le rejeu et la répudiation des messages, le contrôle d'accès et l'octroi des doits aux utilisateurs, sont souvent nécessaires dans les communications de groupes.

Dans [LIV 1.4], une définition simple est donnée pour les mécanismes de sécurité utilisés :

> *« … Des procédés de sécurité basés sur une distribution de clés publiques et associés parfois à des clés de session semblent résoudre les problèmes d'authentification, d'intégrité et de non répudiation. Cependant, de nouvelles difficultés prennent place et imposent d'autres solutions.*
>
> *La difficulté réside dans l'assurance qu'une clé publique donnée concerne bien la personne correspondante. Une clé isolée n'est qu'un fichier de symboles sans lien significatif avec son propriétaire ou son éditeur s'il est différent. Cette lacune a lieu quand les clés ne sont pas distribuées directement aux utilisateurs par des moyens sûrs (main à main par exemple). Le fait de les distribuer par simple consultation d'un annuaire ou*

d'un serveur Web, vulnérabilise les mécanismes de sécurité décrits précédemment.

Pour empêcher que des parties tierces malveillantes génèrent de fausses clés et pour instaurer ainsi un climat de confiance, une notion de certificat numérique a vu le jour. Au sein de ce système, une entité appelée Autorité de Certification (AC), Certification Authority (CA) jouit de la confiance de tous les utilisateurs. Sa signature apposée est garante de la conformité des données contenues dans un certificat. Cette autorité signe tous les certificats qu'elle émet par sa propre clé privée. Les utilisateurs vérifient l'authentification et l'intégrité du CA et des données qu'il contient en déchiffrant par la clé publique du CA. Sous entendu, CA a généré préalablement une paire de clés pour lui: une privée qu'il garde et une publique qu'il diffuse largement. Là encore il faut être sûr qu'aucune autre entité ne va venir falsifier cette clé pour passer pour CA. Pour cela, plusieurs scénarios sont possibles et font l'objet des différentes études … »

Dans [LIV 2.2], les auteurs expliquent les principaux types de certificats spécifiques aux applications de groupes. Une proposition pour l'application des certificats aux groupes est faite dans [LIV 2.3].

Nous présentons dans ce mémoire un résumé qui illustre l'utilisation des certificats pour des groupes. Vous trouverez plus de détails dans [LIV 2.2] et [LIV 2.3]. Deux certificats sont proposés : les certificats d'identité (CI) et les certificats d'attributs (CAtt). Il est proposé d'utiliser le CI comme équivalent au passeport d'une personne. Le CAtt est équivalent à un visa pour le passeport.

2.2.3.1 Certificats d'Identité

Un certificat d'identité est défini comme une structure de données conçue principalement pour fournir un service d'identification. Ce service peut intervenir dans d'autres services, tels que l'intégrité des données, l'authentification d'entité et la confidentialité. Il permet de lier différents éléments au moyen de la signature d'une autorité de certification. La norme internationale X509 définit les certificats. On retrouve dans leur format les principaux champs suivants :

- Le nom de l'autorité de certification qui a créé le certificat
- Le nom et le prénom de la personne détentrice
- Son entreprise
- Son service
- Son adresse électronique
- Sa clé publique
- La date de validité du certificat
- Des informations optionnelles
- Une signature électronique (d'une autorité reconnue de confiance)

Le standard X509 est le principal format utilisé pour les certificats. La norme qui le définit est : ITU-T X.509 ou ISO/IEC 9594-8.

Afin de considérer des nouveaux besoins, il est possible d'ajouter d'autres informations en introduisant des blocs de données pouvant supporter n'importe quel type d'extension. L'identificateur d'une extension est défini selon les normes ITU-T Rec. X.660 ou ISO/IEC 9834-1.

2.2.3.2 Certificats d'Attributs

Les Certificats d'Attributs sont conçus pour lier une identité donnée à des Droits/Privilèges. Ils ont été proposés pour compléter les certificats à clé publique et définir ainsi les droits que possède le détenteur d'un certificat d'identité.

Les CAtt et les CI, sont tous les deux définis par le standard X509. Dans ce standard, la structure d'un CAtt est la même que celle d'un CI. Une des différences est qu'il ne contient pas la clé publique d'une personne. Comme il ne contient pas d'information explicite sur l'identité de la personne, un CAtt ne peut pas être utilisé pour authentifier le possesseur du certificat. C'est pourquoi, le possesseur d'un CAtt doit avoir aussi un CI, auquel il est fait référence dans le CAtt.

Un CAtt peut se voir modifier de façon dynamique ses droits par une autorité habilitée. Ceci permet une attribution dynamique des droits au détenteur.

Les Certificats d'Attributs comportent, à la base :

- Une partie renseignant sur le certificat lui-même
- Une autre partie sur l'identité du détenteur
- Une partie sur ses droits
- Une partie sur l'autorité ayant délivré le certificat et sa signature.

Il existe une partie appelée Options qui peut contenir des informations optionnelles. Ces champs peuvent servir pour des applications futures.

Un certificat d'attributs peut avoir des formes différentes : le certificat d'appartenance et celui de commandement sont deux formes que l'on peut donner à un CAtt.

2.2.4. Autres mécanismes pour des systèmes particuliers/spécifiques

Depuis quelques années, nous assistons à une multiplication des technologies présentes dans la communication de groupes sécurisées. L'émergence de travaux liés aux réseaux sans fil et aux réseaux Ad-Hoc traitent de façon très spécifique certaines facettes de la communication de groupes sécurisés. Certains auteurs [WAN 05] [ASO 00] [WU 05] [CHI 06] spécialisent les techniques de sécurisation. Ils les adaptent aux contraintes de ces systèmes, contraintes telles que la faible énergie des réseaux et de certains hôtes terminaux. Ces adaptations se font au travers de scénarios plus contraignants ou proches des problématiques de ces systèmes. Cette sous-section développe une étude qui montre une évolution des nouvelles solutions.

2.2.4.1 Réseaux sans Fil

Wang et Bhargava [WAN 05] présentent un protocole focalisé sur la minimisation des calculs effectués lors de la génération de clés des groupes dans des réseaux sans fil. En utilisant des *polynômes* et des *tableaux plats,* ils diminuent la charge de la génération et donc du renouvellement de clés. Ils utilisent les algorithmes de chiffrement symétriques et asymétriques.

Ils basent leur proposition sur l'idée que l'utilisation de clés privés pour la protection des dialogues entre différents groupes dans une session (communication inter-groupe), en combinaison avec la traditionnelle technique symétrique pour la protection des messages échangés à l'intérieur de chaque groupe (communication intra-groupe), peut diminuer la présence des attaques classiques comme la manipulation et l'interception des émissions multicast que

l'on trouve dans la communication purement symétrique.

La génération et la distribution des clés symétriques pour la communication intra-groupe se fait hors ligne et de façon manuelle pendant l'étape d'initialisation. L'utilisation de deux groupes de clés est nécessaire : un ensemble de clés pour protéger les messages appartenant à un groupe, un deuxième ensemble pour protéger les premières clés lors de leur actualisation. Le temps nécessaire à cette initialisation n'est ainsi pas ajouté au temps de traitement et de sécurité pour la partie symétrique, lors du déroulement d'une session.

Basés dans des techniques de génération des clés avec des polynômes et tableaux plats, Wang et Bhargava [WAN 05] proposent l'utilisation des clés privées pour la communication inter-groupes. La génération et l'actualisation des clés privées est faite en dehors de l'algorithme de renouvellement de clés symétriques, ce dernier se faisant en ligne. Pour qu'un membre d'un certain groupe puisse communiquer avec des membres d'autres groupes, il génère, à l'aide du polynôme de son groupe, la clé privée appartenant à son groupe et au groupe destinataire. Deux membres du même groupe *i,* qui veulent communiquer avec un groupe extérieur *j,* ne détiendront pas la même clé privée. Un changement d'un membre dans un groupe *i* n'oblige donc ni à changer le polynôme du groupe, ni à changer les clés privés des autres membres dans son groupe de communication. Par contre, un renouvellement de la clé symétrique du groupe modifié, y compris la distribution de cette clé chez tous les membres, sera obligé. Wang et Bhargava contribuent ainsi à la diminution du trafic causé par la gestion de clés pour la communication inter-groupes.

Pour mettre en œuvre leur mécanisme, ils utilisent une méthode de stockage additionnelle [WAN 05]. Ils minimisent cette charge additionnelle en proposant la distribution de ce stockage dans les nœuds composant le réseau.

2.2.4.2 Réseaux Ad Hoc

Dans [ASO 00], Asokan et Ginzboorg, traitent des problèmes de sécurité dans des réseaux ad-hoc. Dans ce type de réseaux, nous cherchons à établir une communication sécurisée entre des entités qui communiquent directement entre elles sans avoir accès à des infrastructures physiques publiques. Ils lancent des nouveaux protocoles qui se basent sur l'utilisation de mots de passe.

Asokan et Ginzboorg classent les différentes structures des réseaux en fonction du type d'application qui les utilisent. Trois types d'infrastructures se dégagent de leurs études :

1. Routing Infrastructure. Avec une organisation fixe des routeurs et des liens stables.
2. Server Infrastructure. Avec des serveurs dédiés à fournir différents services pour la communication : nommage, annuaire, certification, etc.
3. Infrastructure de support organisationnel et administratif qui assure des services pour : l'enregistrement, la gestion des certificats et des domaines.

Ainsi, Asokan et Ginzboorg établissent dans la communauté des communications de groupes une problématique qu'il est nécessaire de faire évoluer.

En effet, dans les réseaux ad-hoc, les besoins pour la communication de groupes sont très spécifiques. Ils s'établissent en fonction du manque d'une infrastructure dédiée à la communication des nœuds. Le besoin d'établir une

communication bien sécurisée qui réponde aux changements en ligne des membres dans le groupe doit aussi être résolu. Un taux important de perte des nœuds faisant partie du réseau est aussi caractéristique de ce type de réseau. Les protocoles utilisés doivent aussi être économes en énergie.

Dans [WU 05], les auteurs abordent plus précisément les réseaux Ad Hoc sans fil. Du fait de ses caractéristiques intrinsèques, ce médium est non fiable et ouvert. Il peut être facilement perturbé ou bien dégradé dans son fonctionnement.

Ils s'intéressent [WU 05], ainsi que Asokan et Ginzboorg [ASO 00], aux problématiques de la mobilité et du manque d'infrastructure dédiée des réseaux Ad Hoc qui compliquent fortement la sécurisation des échanges. La cryptographie est normalement utilisée pour protéger l'information échangée. Autant les techniques symétriques et asymétriques peuvent être utilisés, cependant, aucune technique ne sera effective si l'administration de la clé utilisée est mal réalisée.

En fait, dans la communication des réseaux mobiles ad hoc, la clé de session représente un élément très important et central de la sécurité. La dynamique et la composition topologique d'un réseau mobile Ad Hoc ainsi que les ressources disponibles dans chaque nœud dans ce réseau, déterminent les exigences auxquelles le protocole de gestion de la clé de session est confronté lors de sa définition.

SEKM (secure and efficient key management) est le système de gestion de clé proposé dans [WU 05]. Il s'agit d'un système qui combine à la fois une structure à clé publique (asymétrique) avec l'utilisation des secrets partagés (symétrique) par des sous-groupes définis dans le groupe en communication.

Dans les réseaux mobiles ad hoc, le développement d'un système part de l'idée qu'un nœud peut communiquer juste aves les nœuds situés dans son rayon de diffusion. Si ce nœud veut dialoguer avec un autre nœud situé hors de sa portée, il doit faire procéder à des retransmissions successives par d'autres nœuds jusqu'au moment où le message arrive au nœud destinataire. La transmission de l'information dépend de la fiabilité de transmission des nœuds participants à la retransmission tout au long du chemin.

Vu la facilité et la rapidité de structuration d'un réseau mobile ad-hoc, protéger un système avec de telles caractéristiques contre des attaques dans la communication tant actives comme passives, est une tâche plus difficile à effectuer que dans d'autres environnements comme les réseaux filaires.

Ainsi, les auteurs dans [WU 05] affirment que les techniques de chiffrement doivent être bien choisies pour garantir les propriétés de sécurité de communication nécessaires dans des réseaux mobiles ad hoc. Des propriétés comme la confidentialité, l'intégrité l'authentification et la non répudiation doivent être assurées indépendamment des conditions spécifiques caractérisant des réseaux à faibles ressources.

L'efficacité de leur technique repose sur la présence d'un groupe de serveurs qui participent à la création et à la distribution de la clé de session. Elle utilise, en plus, un système pour la gestion des certificats octroyés aux membres dans le réseau mais aussi aux autorités de certification. Un concept de « *ticket* » a été introduit pour rendre plus effective la génération et l'actualisation des certificats. On peut dire que SEKM est un système hybride. Il est distribué dans le sens qu'il ne laisse pas la tâche de gestion et de distribution de la clé à un seul nœud central. Il n'est pas complètement distribué car, pour effectuer cette gestion, un

groupe limité de nœuds est déclaré comme le groupe de serveurs en charge des opérations.

Le risque que la distribution de la gestion de la clé de session implique dans des systèmes distribués est compensé par la création bien maîtrisée d'un groupe de serveurs qui sera en charge de cette gestion. Contrôler les serveurs qui peuvent participer et les faire sortir du groupe au moment où ils ne remplissent pas le minimum de contraintes de sécurité requises permet d'éviter l'entrée ou la participation d'un attaquant dans le groupe.

Distribuer la tâche de gestion de cette clé à un groupe de nœuds permet de déléguer cette tâche à une autre autorité en cas d'absence d'une première autorité choisie.

La distribution de cette gestion la rend aussi plus sûre et plus fiable car on ne fait pas reposer cette responsabilité sur un seul nœud. L'apport de chaque nœud participant permet de créer une clé plus solide.

En ce qui concerne l'authentification des membres, une autorité parmi les membres du réseau est désignée pour attribuer des certificats. Ce système permet de protéger le groupe contre l'inscription de nœuds frauduleux. Il considère la possibilité de révoquer un certificat ainsi que de le réactiver.

Un modèle qui varie de 40 à 100 nœuds à été implanté en MatLab. Le système de chiffrement RSA de 1024 bits a été aussi implanté. Différentes analyses ont été faites, comme par exemple le temps de vérification des certificats durant l'étape d'initialisation ainsi que le temps de convergence pour la génération et l'actualisation des certificats et de la clé de session.

Les auteurs ont fait varier (i) le pourcentage de nœuds qui font partie du groupe de serveurs, (ii) le pourcentage de nœuds minimaux qui doivent participer à la gestion de la clé et des certificats, et (iii) la taille de la clé utilisée pour le système RSA, sachant que, plus la clé est longue, plus elle est sûre. Ils ont obtenu les résultats suivants : l'utilisation d'un groupe de serveurs donne des réponses plus rapides que si les services sont dispersés de façon désorganisée dans le réseau. Ils montrent aussi que plus le nombre de serveurs est grand, plus le temps de calcul de la clé de session et des certificats est long. Finalement, ils montrent que le temps de calcul de la clé de session ne change pas beaucoup si la taille de cette clé se trouve entre 256 et 1024 bits. Cependant, le temps augmente considérablement si la taille passe à 2048 bits.

2.2.4.3 Cryptographie basée ID

Avec l'utilisation de *« Identity-based cryptography »* nous pouvons contourner le processus d'authentification qui utilise des signatures digitales et des certificats. Chik et al. proposent un protocole de génération de la clé de session basé ID ainsi que quatre protocoles correspondant aux services les plus courants à la gestion de groupes : *join, leave, merge et partition*. Ils basent leur solution sur une variante du schéma de signature de Guillou-Quisquater (GQ) [GUI 88] et sur le protocole BD [BUR 94]. Ils structurent les nœuds participants dans un anneau logique. En deux tours, ils calculent l'identificateur IDi de chaque membre, selon les mêmes principes de calcul que l'algorithme DH : l'utilisation d'une fonction générateur *g,* des grands nombres premiers *p* et *q*, et des fonctions à sens unique.

Ils éliminent donc la présence de l'autorité de certification en ligne. Ils en gardent une, mais juste pour la gestion initiale des clés privées des membres.

Leurs principaux apports sont :

1. Des services sécurisés de gestion de clés de session qui répondent à la dynamique de la composition du groupe.
2. La mise à disposition d'un système de communication sécurisé point à point *(bilinear pairings)* sans devoir échanger une clé secrète de communication
3. La proposition d'un algorithme de génération de la clé de session appelé *« tripartite key agreement »* qui n'a besoin que de deux broadcasts. Il supporte efficacement la communication de groupe ainsi qu'un dialogue entre deux personnes si un administrateur désigné supervise cette communication.
4. En basant leur algorithme *« tripartite key agreement »* sur l'utilisation des arbres [RHE 05], ils montrent son efficacité. Pour l'authentification des membres, leur algorithme a été prouvé moins complexe que les algorithmes : *signature scheme ID-based SOK 194 bits* [SAK 00]*, BD-ECDSA 160 bits, BD-DSA 1024 bits] et SSN 1024 bits, algorithme basé aussi ID* [SAE 98]. Pour les protocoles de gestion de groupes, ils montrent aussi que le nombre de messages intervenant dans leur protocole est plus faible que celui du protocole BD.

2.3. Architectures pour la communication des groupes

Le domaine d'exploration étant très large, nous nous sommes volontairement limités aux travaux de recherche dans des groupes de protocoles qui peuvent s'appliquer à notre problématique.

Les sous-sections suivantes montrent les différents protocoles qui ont contribué à l'évolution des protocoles généralistes. Le but final est de fournir une structure de base pour la communication de groupes tout en prenant en compte une grande gamme de besoins concernant les applications de groupes.

2.3.1. Premières architectures pour la communication de groupes

Les architectures pour la communication de groupes peuvent se diviser en deux grands groupes. Tout d'abord, nous allons trouver un ensemble de systèmes qui ont été développés dans les années 90s. Comme les caractéristiques réseaux étaient différentes, leurs travaux de recherche se sont focalisés principalement, à garantir la fiabilité des services de transmission pour des groupes, ce qui était le principal problème à résoudre dans la communication de groupes.

Des sujets comme l'ordonnancement des messages et la tolérance aux fautes ont été abordés en profondeur. Les thèmes de la sécurité et de la performance dans la communication ont ensuite pris une place plus importante. L'intérêt de cette sous-section est de faire un état de l'art de ces travaux pour établir les fondements des protocoles faisant partie de notre domaine d'étude [TEM 01].

Un deuxième groupe correspond aux architectures qui feront l'objet d'étude de la sous-section 2.3.2. Ces architectures sont plus récentes. Elles font partie de notre axe d'étude.

2.3.1.1 Isis, Rmp, Totem

Birman et Cooper sont à la tête du projet **ISIS** depuis 1983 [BIR 90]. Ils développent les bases des systèmes pour la communication de groupes

sécurisés. Leurs travaux deviendront plus tard, une référence dans ce domaine. Dans ISIS, des concepts comme « group membership », « group communication » et « multicast communication » sont déjà abordés. Leurs travaux donneront comme résultat la création du nouveau système Horus qui sera exposé lors du développement de cette sous-section 2.3.1.

RMP est un autre protocole publié dans [WHE 94] dans le domaine de la communication distribuée. L'objectif de ce protocole est de créer une couche au dessus de services multicast datagrammes non fiables comme le protocole IP Multicast. Il fournit donc des services d'ordonnancement total, de fiabilité et d'atomicité pour des communications multicast. Le niveau de qualité attendu pendant l'émission des messages peut être choisi paquet par paquet. Nous situons ce système dans le même contexte que les systèmes Isis et Totem.

Totem [MOS 95] de son côté propose de fournir dans sa couche de protocole des services de tolérance aux fautes tout en considérant la possibilité d'avoir des partitions du réseau physique. Il a été développé avec les mêmes objectifs que Transis, qui sera développé par la suite. Une des grandes différences entre Totem et Transis est l'utilisation d'un système d'anneau logique pour son implémentation.

2.3.1.2 Transis

Dans [DOL 96], Dolev et Malki ont développé le système **Transis** avec comme objectif de maintenir en fonctionnement sans interruption de service pendant au moins une journée, un système de communication de groupes temps réel mis en œuvre pour une journée d'élections politiques. L'objectif est de fournir aux applications de groupes un système de communication multicast fiable et utilisable dans des réseaux à grande échelle. Un de ses principaux buts a été de considérer des possibles partitions physiques du réseau tout en maintenant la même vue du système chez tous les nœuds, palliant ainsi d'éventuels disfonctionnements physiques. Dolev et Malki comparent leur système avec le système Isis. Plus précisément, le système Transis dirige ses efforts vers l'élimination de l'entité centrale d'Isis pour contrôler la communication distribuée. Avec ce travail, les auteurs cherchent à éliminer les inconsistances présentes lors des déconnexions d'un certain membre ou d'un groupe de membres en communication. Ils inscrivent le système Transis dans le cadre de systèmes tolérants aux fautes, problématique travaillée de manière similaire par d'autres auteurs dans : Totem [MOS 95] et Horus.

2.3.1.3 Rampart

Rampart a été proposé dans [REI 95] comme une boîte à outils « toolkit » que les développeurs peuvent utiliser pour la production d'applications de groupes. Les services fournis par cette boîte doivent être intégrés au système final, et leur exécution doit être transparente vis-à-vis des utilisateurs. L'atomicité des envois multicast asynchrones a été une des plus grandes préoccupations lors du développement de Rampart. L'intérêt est d'intégrer, dans son noyau, une série de protocoles utiles à des systèmes distribués, qui résolvent des problèmes de base dans la communication comme l'atomicité et l'ordonnancement des messages, la synchronisation de la communication ou la tolérance aux fautes byzantines. Des services de sécurisation du système, de gestion des groupes, de fiabilité et d'atomicité dans la communication sont des exemples des services fournis à travers cette boîte.

Rampart intègre une technique de réplication des services dans le réseau avec comme objectif d'offrir une disponibilité fiable des services aux membres.

L'inconvénient de la réplication est cependant d'augmenter les possibilités d'attaque du système. L'augmentation du nombre de serveurs dans le système, oblige à surveiller un nombre plus grand de machines qui ne doivent pas être compromises, ce qui n'est pas un travail simple. Le système propose des services de chiffrement des messages et d'authentification des membres pour protéger les communications. L'utilisation de ces services peut pénaliser considérablement la performance du système. Il faut donc choisir un équilibre adéquat entre la disponibilité des services attendus et la sécurité des services à garantir dans le système. Cet équilibre est choisi en fonction des besoins des applications.

2.3.1.4 Secure Ring

Secure Ring [KIH 98] est un autre système qui fournit des services orientés vers la communication multicast. Comme Rampart, Secure Ring offre des services de gestion des membres, d'atomicité, de fiabilité et d'ordonnancement des messages envoyés de façon asynchrone dans un groupe multi utilisateurs. Sa différence avec Rampart vient de la formation d'un anneau logique avec les membres participants dans un groupe et l'utilisation de *tokens* lors de l'envoi, la réception et la détection des fautes dans le fonctionnement normal du système. Les fautes qui peuvent correspondre à des attaques dans la communication sont identifiées au moyen d'un détecteur faisant partie du protocole. Secure Ring représente une extension de Totem. Il contient en plus, un détecteur de fautes Byzantines. Cependant, comme Secure Ring utilise un « token » qui contient une même signature digitale pour tous les messages, que ce soit des messages de données ou de contrôle, échangés lors de la communication, le trafic réseau augmente considérablement. La performance du système se dégrade fortement en présence d'un médium bas débit qui produit de multiples déconnexions. Secure Ring ne supporte pas non plus les grands réseaux.

2.3.1.5 Horus

Un protocole qui couvre de manière la plus complète possible les besoins de communication des groupes doit offrir des services à différents niveaux. **Horus** [REN 96] représente à l'époque, un excellent exemple des premiers systèmes complets conçus pour supporter des applications de groupes.

Horus a été introduit en 1996 comme un système révolutionnaire qui offrait aux développeurs des applications de groupes un modèle flexible et extensible. A cette époque, les applications étaient confrontées à deux nouveaux besoins : « la gestion du passage à l'échelle des groupes » et « la garantie de temps de réponses adéquats par rapport aux besoins des systèmes temps réel ». Des solutions basées sur l'utilisation de clusters ont été proposées pour répondre à ces besoins. Des services de gestion de groupes et de gestion de l'information échangée entre leurs membres, en assurant un déploiement sécurisé, sont les principales caractéristiques de ce système. De façon plus précise, les services les plus significatifs fournis par Horus sont les suivants : diffusion de messages multimédia, distribution hiérarchique des messages, fragmentation et recomposition des messages, synchronisation, contrôle de congestion, délivrance ordonnée des messages, chiffrement et déchiffrement des messages, gestion de la mobilité des membres, fusion et séparation des groupes. Cet

ensemble complet de services est disponible et ces services peuvent s'auto configurer pendant le déroulement d'une session. Ce système est présenté comme une composition de *« Microprotocols (APIs)* » flexibles pour des nouvelles applications collaboratives. Et pour répondre à tout type de problème qui peut se présenter vis-à-vis des exigences des applications, Horus peut interagir au travers d'une interface capable de fournir des services additionnels aux applications collaboratives avec des architectures déjà existantes. L'intégration d'Horus à des systèmes déjà existants est faite de façon transparente. L'intérêt est de fournir à ces systèmes une interface qui permet d'avoir accès aux services de communication de groupes innovants d'Horus, tout en gardant les fonctionnalités du système déjà existant. Ce système n'aura pas besoin de changement substantiel dans son implémentation par rapport à sa configuration de base.

2.3.1.6 Ensemble

Dans [BIR 97], Birman et all. identifient trois périodes dans leurs travaux, chacune étant marquée par trois systèmes différents. La première est celle du système/protocole/projet ISIS, de 1985 à 1990. La deuxième est celle du système/protocole/projet Horus, de 1991 à 1996.

Finalement, dans une troisième période, deux nouvelles exigences s'imposent : (1) avoir des protocoles plus performants et (2) sécuriser la communication. De 1996 à 2001, nous trouvons le développement du nouveau système/protocole/projet **Ensemble** [BIR 97]. Le principal objectif de cette version est d'explorer des nouveaux terrains vis-à-vis de l'apparition des nouveaux environnements réseaux.

[BIR 00] part de la version du système Horus pour l'étendre dans le système Ensemble. La tolérance aux fautes, la sécurité, la consistance des réponses et la garantie des performances temps-réel du système sont les axes de ces travaux. Ils montrent comment l'intégration de la tolérance aux fautes et de la sécurité peut se réaliser tout en gardant une bonne performance du protocole de communication dans des systèmes temps réel. Leur système a été la base de communication pour plusieurs systèmes manipulant des groupes.

Pour augmenter la flexibilité, les applications peuvent être développées dans les langages C, C++, Java ou autres. Elles peuvent s'interfacer à Ensemble, pour utiliser un kit complet d'« outils de communication » sous la forme de librairies. De nouvelles plateformes comme NT et Linux sont considérées pour offrir une solution de communication de groupes à des nouveaux environnements. L'équilibre qualité-performance dans la communication est déterminé par le développeur. Il peut choir la configuration qui lui convient le mieux. Bien entendu, cet équilibre est défini vis-à-vis des exigences établies lors de la définition du système requis.

Le passage à l'échelle du système Ensemble pour des groupes de grande taille est une tâche qui reste encore un objet d'étude et de développement.

Dans [ROD 01] les auteurs parlent de la couche sécurité pour la communication qui a été intégrée à Ensemble.

La communication sécurisée d'Ensemble a servi de support à d'autres infrastructures de collaboration synchrones implantées en JAVA RMI ou Microsoft DCOM [SHA 01].

2.3.2. Architectures pour la communication des groupes sécurisées

A la fin des années 90s, les systèmes de communication de groupes ont évolué. De nouvelles architectures ont été proposées avec les buts suivants :

1. Apporter des solutions plus complètes
2. Réutiliser le travail déjà développé dans certains aspects de la communication des groupes
3. Pouvoir diviser les tâches dans différents groupes de travail.

Chacun des groupes pourra maintenant se focaliser sur ses propres centres d'intérêts et mettre en œuvre les services qui correspondent le mieux à ses besoins. Nous avons par exemple des groupes dédiés à la gestion des secrets nécessaires à la communication, des groupes dédiés a la protection des données et d'autres dédiés a la gestion de la dynamique des groupes vis-à-vis de la sécurité que l'on doit assurer.

2.3.2.1 Cactus

Ayant à la tête du projet les chercheurs Richard D. Schlichting et Matti A. Hiltunen du département des sciences de l'informatique de l'Université d'Arizona, Cactus a vu le jour en 1996. Leurs dernières publications datent de l'année 2001 [CACTUS].

La description de Cactus a été publiée la première fois dans [HIL 97], avec comme axe d'intérêt la sécurité dans la communication des applications distribuées. Basé sur des mécanismes redondants de chiffrement, la faiblesse du protocole Cactus était le grand trafic qu'il impliquait. Dans [HIL 00], Hiltunen et al. publient une nouvelle version de Cactus où ils réduisent considérablement le trafic réseau causé par les mécanismes qui assuraient la qualité des services pendant la communication entre des groupes.

Cactus se caractérise par l'introduction d'une couche middleware contenant des services nécessaires à la mise en place d'une communication des groupes.

Cactus présente des entités de services construites dans des modules logiciels indépendants appelés *Micro-protocoles.* Ces entités peuvent être utilisées pour la construction des systèmes ou des applications. Une instance personnalisée d'un service est construite en sélectionnant un ensemble de *micro-protocoles* basés dans les propriétés à garantir. Un *protocole composé* est donc proposé, et il fournit le service personnalisé dont l'application a besoin.

Cactus a été implanté dans différentes plateformes comme par exemple pour le langage C sous le système d'exploitation Linux et pour le langage Java sous différents systèmes d'exploitation.

En se basant sur les classes de services dont les applications de groupes peuvent avoir besoin, une série de services prototypes ont été créés sous la forme de *micro-protocoles.* Ces services peuvent être classés en trois grands groupes :

- Des services concernant la distribution de messages (groupe RPC) : il s'agit des services qui garantissent l'ordonnancement et l'atomicité dans la distribution des messages. La fiabilité dans la communication est aussi intégrée à Cactus. Des points comme les fautes byzantines, la perte de connexion et la non synchronisation sont traités.
- Des services concernant la sécurisation de la communication (groupe SecComm) : il s'agit des services pour la sécurisation du transport de l'information. L'intérêt étant de diminuer la charge des traitements et le trafic dû à

la configuration et à la réalisation de cette sécurisation, la mise à disposition de cet ensemble de protocoles sous la forme d'entités middleware représente une bonne solution en ce qui concerne la performance du système vis-à-vis de ses besoins sécuritaires. Seuls les services point-à-point sont considérés. Les services de sécurisation dans un environnement multipoint font objet d'études futures.
- Des services concernant la gestion des canaux pour la distribution de messages temps réel (groupe RTD) : en utilisant des abstractions de canaux, ce groupe de *micro-protocoles* vise à offrir un service de distribution d'information temps réel.

2.3.2.2 Enclaves

Enclaves représente une bonne solution pour la sécurisation des communications de groupes dans un environnement Internet. Publié dans [GON 96], il s'agit d'ignorer les équipements physiques qui protègent un réseau contre des attaques (par exemple des *firewalls*), et d'implémenter une sorte de Rempart de sécurisation de façon logique, d'où le nom d'Enclaves, en faisant participer les membres de la communication. Cette substitution est due à la difficulté d'adaptation des *firewalls* aux besoins particuliers des applications, mais aussi à la recherche de la diminution du coût de sécurisation des systèmes.

Configurer un ensemble de membres dans un réseau sécurisé virtuel permet, à un groupe d'utilisateurs, de dialoguer sans devoir utiliser des équipements plus chers pour protéger leurs échanges contres des attaquants éventuels. Chaque équipe du réseau devient un élément de sécurisation de la communication.

Nous avons considéré Enclaves car les auteurs résument dans leur proposition et avec un format très simple les fonctionnalités génériques des protocoles de groupes sécurisés. Dans [GON 96], Li Gong montre une architecture en forme de couches. Dans une première couche, située au dessus du système d'exploitation, Enclaves dispose d'une série des primitives pour l'authentification des membres ainsi que pour le chiffrement d'information échangée dans le groupe. Dans une deuxième couche, Gong propose de faire la gestion des membres des groupes. Ensuite, l'architecture présente une troisième couche dédiée à la communication sécurisée. Finalement, la dernière couche contient les applications de groupes sécurisées.

Des opérations pour la gestion des groupes de composition dynamique sont considérées. Des mécanismes de communication asymétriques, pour des émissions point-à-point, ainsi que symétriques pour des émissions multipoint, sont aussi pris en compte dans Enclaves.

2.3.2.3 Antigone

Antigone [IRR 03] a été créé avec comme premier but de réaliser une solution qui s'adapte à la variation du besoin de sécurité des applications de groupes. Cette sécurité varie en fonction de l'environnement de l'application, des données concernées et des membres participant au système.

Pour certaines applications où l'information échangée doit rester privée aux membres, la confidentialité est une garantie à assurer. Pour d'autres applications, comme celles d'annonces sur Internet, les propriétés d'intégrité et d'authentification sont une priorité. Un autre exemple peut être la grande différence existant entre les applications qui se déroulent dans des environnements haut débit et celles qui fonctionnent sur des environnements avec des ressources restreintes, où les mécanismes de protection des données

à choisir peuvent varier fortement en fonction de la performance du système attendue.

Antigone [McD 99] se présente comme un environnement avec une variété de mécanismes disponibles et à partir desquels on peut créer un ensemble de politiques de sécurité personnalisées aux besoins d'une certaine application. En effet, l'intérêt n'est pas de mettre à disposition des applications les politiques figées de sécurité dont elles peuvent disposer. Antigone offre au développeur, des mécanismes qui peuvent se combiner et se configurer entre eux de façon à considérer les particularités du système de communication et ses besoins en termes de sécurité. Une interface facile à utiliser permet l'utilisation de ces mécanismes disponibles. Le résultat de ce travail est un ensemble de politiques de sécurité dont une application spécifique a justement besoin.

Ensuite, un effort pour développer des politiques standards a été fait pour faciliter le travail de développement des applications qui peuvent facilement s'ajuster à des politiques génériques. Un kit de services sécuritaires prêt à utiliser est proposé à des applications n'ayant pas des spécifications très contraignantes et très particulières.

Antigone est proposé tout en veillant au respect des objectifs suivants : 1). Flexibilité pour la définition des politiques de sécurité. 2). Considération d'une grande variété d'applications de groupes. 3). Mise à disposition de mécanismes indépendamment de l'infrastructure disponible. 4). Independance de la couche transport. 5). Veiller à ce que la performance du système ne se dégrade pas.

Les services qui peuvent être assurés en utilisant les mécanismes d'Antigone sont : l'authentification, l'entrée et la sortie des membres d'une communication, des opérations pour le renouvellement et la distribution des clés et des opérations pour la protection des données échangées. Ils considèrent aussi les modes de transmission multicast et point-à-point.

McDaniel et Prakash [McD 00] présentent une étude de performance d'Antigone. En partant des hypothèses que la gestion de la clé de session ainsi que la protection des données consomment la grande majorité des ressources d'un protocole de groupes, ils focalisent leurs analyses sur ces sujets.

En faisant une modélisation d'un réseau LAN Ethernet à 100 Mbps, avec dix utilisateurs et avec les algorithmes DES-56 bits et RSA-512 bits pour le chiffrement symétrique et asymétrique respectivement, les auteurs montrent que la taille du groupe a un impact important dans la charge de travail pour la génération de clés pendant que pour la distribution a une répercussion plus faible. Ils proposent l'utilisation de clés hiérarchiques comme une solution palliative à la charge que le renouvellement d'une seule clé dans le groupe implique lors du déroulement d'une session. Octroyer un ensemble de clés au groupe permet de diminuer la quantité de nœuds obligés à renouveler la clé de session face à des changements de composition, ce qui diminue considérablement la charge du protocole de gestion de la clé. Un changement dans la composition du groupe oblige juste au changement de la clé correspondant au sous-groupe où le changement s'est présenté. Tous les autres sous-groupes n'auront pas besoin de changer leur clé de session.

Une API (Applications Programming Interface) accessible aux programmeurs a été développée dans Antigone. La facilité d'utilisation de cette API, et de l'environnement Antigone est mise en avant avec une application d'audio-conférence. Une relation directe entre le choix des politiques de sécurité et la performance du système est mise en avant. La facilité d'analyse des applications

qui utilisent Antigone, avec une granularité d'analyse variable, est un autre grand avantage d'utilisation de cet environnement.

2.3.2.4 Spread

En 1998 [AMI 98], les auteurs de Spread proposent un protocole qui traite à la fois la sûreté de fonctionnement, la gestion de groupes et la sécurité dans la communication des groupes. Aujourd'hui, Spread représente pour les applications, un exemple de système client-serveur distribué qui met à leur disposition plusieurs protocoles pour la gestion de la clé du groupe nécessaire à la sécurisation de leur information échangée.

Au sein du module de gestion de la sécurisation de cette communication, une application de groupes peut choisir entre cinq différents protocoles gestionnaires de clés étant issus des travaux menés au sein des autres groupes de recherche : [BUR 94], [KIM 04b], [STE 00], [KIM 04a]. Spread a intégré ces protocoles dans des librairies accessibles par des APIs. La majorité de ces protocoles sont des extensions du protocole n-party DH [DIF 76]. Cette variété de versions de protocoles permet à l'utilisateur d'établir, à partir des caractéristiques de son application, le meilleur compromis sécurité-performance pour effectuer la gestion de la clé de session.

Pour les auteurs de Spread [AMI 05], les systèmes de groupes peuvent être enrichis par des services de sécurité sans sacrifier ni leur robustesse, ni leur performance. Pour cela, Spread propose des architectures dites Intégrées. Il dédie un ensemble de serveurs à l'implantation des services de sécurité. Les membres d'une session n'ont pas besoin d'utiliser leurs ressources propres pour des opérations de sécurisation de la communication qu'ils effectuent. Spread met en avant l'importance de la diminution de cette charge de gestion chez les membres participants à la communication.

L'architecture de Spread cherche à supporter la nouvelle problématique que les applications actuelles de groupes rencontrent : le passage à l'échelle, problème qui affecte directement le processus de gestion des clés de groupes [AMI 05]. Pour Amir et al., la maîtrise du passage à l'échelle dans un réseau ainsi que le fait d'offrir la possibilité de choisir un équilibre sécurité-performance pour une application font partie de leurs priorités. Paradoxalement, la présence de serveurs devient aussi la principale faiblesse de Spread car ils ne doivent pas être défaillants.

Amir et al. s'intéressent, comme Antigone, à fournir aux applications une interface simple à utiliser et qui donne accès aux services de communication pour des applications distribuées. Des protocoles avec des services de tolérance aux fautes, de synchronisation et de sécurité sont mis à la disposition des applications au moyen d'APIs. Présenter les services disponibles d'un protocole de groupes dans des librairies accessibles au travers d'une API permet une utilisation facile et guidée du protocole. L'existence d'un noyau-moteur du protocole appelé démon sera nécessaire dans chaque nœud participant à la communication avec Spread pour répondre en temps réel aux besoins de communication.

Nous listons ici les services et les avantages les plus importants de Spread :

1. Communication de groupes fiable, avec gestion du passage à l'échelle
2. Construction d'Architectures à l'aide d'une API simple à utiliser mais aussi puissante

3. Passage facile entre des systèmes locaux (LAN) et des grands systèmes (WAN)
4. Support de milliards des groupes de différente composition
5. Garantie d'une communication fiable face aux problèmes de déconnexions, de défaillance des nœuds et des changements dans la composition des groupes
6. Garantie de la bonne performance et de la consistance de fonctionnement du système
7. Fonctionnement distribué du système

Spread est composé des éléments suivants :

1. Un module de gestion des membres et de distribution des messages, qui peuvent fonctionner en mode VS (Virtual Synchrony. VS) ou EVS (Extended Vistual Synchrony. EVS). La différence entre les deux modes fait référence aux vues du groupe au moment de l'émission des messages, ainsi qu'au moment de la réception des messages. Dans VS, les deux vues sont pareilles. Dans EVS elles peuvent varier sans que la délivrance des messages ne soit inconsistante. L'utilisation d'EVS donne des résultats plus performants que VS.
2. Un module de services de sécurisation de la communication avec principalement les services suivants : authentification des membres, contrôle d'accès, garantie de l'intégrité des données, authentification de la source des données et confidentialité dans la communication.
3. La gestion des groupes (permettant un grand nombre de groupes et de membres) en communication sans que la performance du protocole ne se dégrade.

En ce qui concerne la sécurité, les services de base de Spread veillent à interdire la participation des personnes non autorisées dans le groupe et à protéger la communication contre des attaques de capture de messages et d'injection des faux messages.

Cependant, il ne faut pas oublier qu'un nouveau travail s'ajoute aux tâches habituelles de gestion de clés, celui de la gestion de la clé pour la communication entre l'ensemble de serveurs. Comme la composition de ce réseau de serveurs ne change pas souvent, la gestion de leur clé de communication est plus simple que celle de la clé d'un groupe avec des membres.

Pour des réseaux de taille moyenne, une architecture sans serveur est facile à installer. Dans ce cas, les auteurs proposent de faire la gestion de la clé de session localement à chaque groupe. En conséquence, le fait de casser une clé d'un certain groupe, ne compromet pas les clés des autres groupes.

D'un autre côté, une architecture avec des serveurs a l'avantage de permettre facilement le passage à l'échelle du protocole des groupes puisque les charges de gestion des clés et du contrôle d'accès des membres, charges additionnelles à la communication entre les membres, font maintenant partie des tâches des serveurs, et non plus des tâches des utilisateurs. Son principal point faible est que le protocole repose sur la confidentialité des clés des groupes liées à la confidentialité de la clé des serveurs et dans l'ensemble même des serveurs qui

peuvent être attaqués.

En réalisant un ensemble de simulations de l'architecture Spread, les auteurs ont montré, par l'analyse du temps de réponse du protocole, que la performance a été largement améliorée par l'utilisation de serveurs. L'architecture de Spread répond bien au passage à échelle. Les courbes montrent un temps de réponse identique pour un petit groupe et pour un grand groupe, c.à.d. que la consommation des ressources liées au déroulement des fonctionnalités n'augmente pas proportionnellement à l'augmentation du nombre de membres dans une session.

2.3.3. Synthèse

Le tableau 2.3 synthétise les axes traités par chacun des travaux présentés précédemment. Chaque ligne du tableau décrit un axe de recherche. Chaque colonne détaille les caractéristiques d'un système de gestion de groupes sécurisé détaillés dans la sous-section 2.3., associé à sa période approximative d'existence. Une intersection noire signale un axe de recherche traité en priorité par un système. Une intersection grise indique un axe de recherche pris en compte par un système. Une intersection blanche signale un axe non traité par un système.

Architecture	ISIS	RMP	Transis / Total	Totem	Ram-part	Secure Ring	Horus	Ensem-ble	Cactus	Encla-ves	Antigo-ne	Secure Spread
Année / Axe	1983 - 1990	1994 - 1996	1992 - 1997	1995 - 1998	1994 - 1996	1998 - 2001	1991 - 1996	1996 - 2001	1997 - 2001	1996 - 1996	1999 - 2003	1998 - 2005
Communication Distribuée	■	■	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒
Ordonnancement	■	■	▒	▒	▒	▒	■	■	▒	▒	▒	▒
Tolérance aux Fautes	■	■	■	■	■	■	■	■	▒	▒	▒	▒
Fonctionnement Distribué			■		■	▒	▒	▒	▒	▒	▒	▒
Fautes Byzantines					■	■			▒	▒	▒	▒
Authentification					■	■	▒	▒	▒	▒	▒	▒
Gestion Dynamique des Membres						■	▒	▒	▒	▒	▒	▒
Système Intégré							■	▒	▒	▒	▒	▒
Systèmes Temps Réel							■	▒	▒	▒	▒	▒
Réutilisation des Systèmes Anciens							■	▒	▒	▒	▒	▒
Chiffrement des Messages							■	■	■	▒	▒	▒
Flexibilité / Adaptabilité aux Nouvelles Technologies								■	■	▒	▒	▒
Performance								■	■	▒	■	▒
Micropotocoles / Middleware / API									■	▒	▒	▒
Construction des Services personnalisés									■		■	▒
Structuration en Couches										■	▒	▒
Dynamiques / Politiques de sécurité flexibles											■	▒
Protocoles de gestion multiples												■
Renouvellement périodique de clé												■

■	Axe traité en priorité
▒	Axe traité
	Axe non traité

Tableau 2.3. Architectures pour la communication des groupes

Avec l'apparition des systèmes ISIS, RMP et Totem, au début des années 90s, nous pouvons observer que les systèmes distribués cherchaient, dans un premier temps, à fiabiliser la communication. Réussir à faire des envois multiples sans affecter le sens du « dialogue » (ordonnancement) était leur priorité. Le sujet de la tolérance aux fautes devenait important puisque les nouvelles applications distribuées se voient confrontées au nouveau problème de la

connectivité du réseau.

Transis aborde un nouveau sujet, celui de la distribution du contrôle de la communication (fonctionnement distribué). L'idée est de répartir la charge du travail dans différents nœuds du réseau, mais aussi de diminuer le risque qu'un contrôle centralisé représente. Ce système marque une étape importante d'évolution dans la communication distribuée.

Rampart et Secure Ring traitent plus formellement les premiers sujets liés à la sécurité dans la communication. Pour eux, le fait que les membres fassent partie de la communication et qu'ils aient leurs autorisations au sein du groupe ne veut pas dire qu'ils ne risquent pas de commettre des opérations frauduleuses lors de la communication. Ils traitent donc les thèmes de détection des fautes byzantines et du renforcement de l'authentification des membres participants dans le groupe. Secure Ring introduit aussi la garantie de la confidentialité et la protection des dialogues tout en permettant le changement temps réel des membres actifs dans le groupe.

Horus et plus tard sa nouvelle version, Ensemble, proposent aux applications l'utilisation d'un système plus complet. L'utilisation des techniques de chiffrement pour la protection des données devient aussi un des principaux sujets pris en compte par ces deux architectures.

L'idée de reprendre des solutions de communication pour des applications de groupes déjà existantes tout en fournissant un système de communication utilisable dans différentes plateformes est un sujet d'abord traité par Horus. Par la suite, Horus propose d'être utilisé pour des systèmes temps réel. Ensemble permet aussi la réutilisation des systèmes anciens. Cependant, il se focalise sur le traitement de la performance dans la communication et sur la possibilité d'offrir aux applications une bonne adaptabilité dans leurs contextes.

Plus tard, en réponse aux nouvelles techniques de développement des systèmes logiciels, Cactus, Enclaves, Antigone et Spread proposent des architectures middleware qui mettent à la disposition des applications des services de communication de groupes implantés sous la forme de micro-protocoles avec des interfaces API faciles d'utilisation.

Cactus et Ensemble poursuivent le même objectif, celui de proposer aux développeurs des applications qui ont la possibilité de fabriquer leurs propres services vis-à-vis de leurs besoins. Enclaves, un système fait pour des réseaux IP, apporte un modèle d'architecture structuré en couches. Il considère de manière globale les grandes problématiques dans la communication de groupe sécurisée.

Antigone de son côté, traite fortement le problème de la sécurité dans la communication. Il se focalise sur la relation performance-sécurité présente dans ce type de systèmes. Afin d'offrir des systèmes adaptables aux besoins particuliers des applications, Antigone offre un nouveau format pour gérer ce lien au travers de politiques performance-sécurité que les applications peuvent établir et appliquer à leurs modèles de communication.

Finalement, l'architecture de Spread intègre l'utilisation de serveurs pour ce qui est la gestion de la sécurité de la communication, ainsi que la possibilité de choisir entre différents algorithmes pour la gestion des clés. Le tableau 2.3 montre que, avec toutes ses caractéristiques qui couvrent tous les axes de recherche identifiés, Spread fournit un protocole qui a apporté les avancées les plus importantes dans le domaine de la sécurisation de la communication de groupes.

2.4. Conclusion

L'étude que nous avons présentée dans ce chapitre montre que le domaine de la communication de groupes implique une grande gamme des services à fournir. L'identification et l'organisation des différentes problématiques dans cette communication structurent le contenu de ce chapitre.

Dans la sous-section 2.2 nous avons présenté des techniques de chiffrement symétriques et asymétriques comme des solutions qui, combinées entre elles, permettent de protéger efficacement l'information échangée au sein des groupes en communication.

Dans la suite de cette sous-section, nous avons remarqué l'importance donnée à la gestion des éléments secrets pour leur utilisation lors de la mise en œuvre des techniques de chiffrement des messages. Le tout est considéré au moment de mesurer l'efficacité globale de la gestion de la sécurité dans la communication des groupes. La bonne combinaison des protocoles choisis et utilisés lors de cette gestion a pour but de garder le système performant.

Nous avons finalement mis en avant l'utilisation des certificats lorsqu'il s'agit de gérer la présence de hiérarchies entre les membres d'un groupe.

Dans la sous-section 2.3, nous pouvons observer l'influence de l'apparition des nouvelles technologies dans l'évolution d'architectures de communication. L'utilisation de réseaux sans fil est un bon exemple des technologies qui ont beaucoup influencé cette évolution. Finalement, les applications actuelles exigent des solutions de plus en plus sécurisées et fiables, sans oublier que la performance est devenue une exigence également importante. Le tableau de la sous-section 2.3.3 reflète de façon très simple et à travers la présentation des architectures analysées, les problématiques générales qui se présentent dans la communication des groupes.

En conclusion, en ce qui concerne l'architecture et la modélisation, un système de communication de groupes sécurisé est un système hautement complexe. En effet, concevoir et vérifier un système qui met en œuvre, en même temps, les différents besoins pour la communication de groupe, la garantie de la sécurité des communications, avec comme objectif de rendre le système performant, introduit trois niveaux de complexité, ce qui rend la tâche difficile à mener avec objectivité.

L'ensemble de ces problématiques et de ces objectifs, traitée de façon cohérente, fera partie de l'architecture présentée dans les chapitres suivants.

3 Spécification des fonctionnalités pour des communications de groupes sécurisés

3.1. Introduction

Le chapitre 3 est dédié à l'analyse des fonctionnalités de base pour la communication de groupes sécurisés. Comprendre le comportement de tels systèmes va nous aider à classifier et à énumérer les fonctionnalités nécessaires pour répondre aux besoins généraux des applications. L'objectif que nous poursuivons est de définir un guide générique que l'on pourra appliquer à une large gamme d'applications.

Tout d'abord, nous procédons à une description, puis à une synthèse la plus générale possible des besoins des applications de groupes sécurisés. Pour obtenir cette description, nous définissons d'abord les ensembles d'acteurs nécessaires. Certains d'entre eux contrôlent les fonctionnalités proposées, d'autres sont sous le contrôle des fonctionnalités.

Par la suite, nous définissons la liste des fonctions que nous avons isolées pour des systèmes de communication de groupes sécurisés. Certaines fonctionnalités relèvent plus de la communication, d'autres de la sécurisation, et les dernières de la gestion des groupes. Nous aborderons ensuite les interactions entre ces fonctionnalités.

3.2. Outils d'analyse et de modélisation

L'identification et la composition des différents modules qui participent à un système de communication de groupes sont difficiles à structurer. Le but étant d'identifier les fonctionnalités que le système doit offrir, nous avons choisi de suivre une méthodologie qui nous aide pour leur identification mais aussi pour leur implantation. Ainsi, en se basant sur certains diagrammes de la notation Unified Modeling Language (UML) 2.0 qui sera introduite par la suite, nous analyserons et documenterons les fonctionnalités de groupes sécurisés que nous proposons. Les résultats de cette analyse pourront servir à l'établissement de standards pour l'analyse et la définition de tels systèmes dans la communauté de communication des groupes sécurisés (*secure group communication SGC).*

Pour de raisons d'organisation et de compréhension, nous donnons dans ce chapitre une explication des diagrammes sur lesquels nous nous appuyons durant le cycle d'analyse du système.

3.2.1. Unified Modeling Language 2.0

L'approche d'analyse que nous proposons de suivre s'appuie sur une méthodologie qui utilise Unified Modeling Language (UML 2.0) [UML] comme langage de modélisation. Le langage UML [OMG 03] est utilisé dans de multiples domaines où se pose le problème de la modélisation de systèmes à prépondérance logicielle. UML est une notation graphique proposée par l'OMG (Object Management Group), conçue pour représenter, spécifier, construire et

documenter les systèmes logiciels. Ses deux principaux objectifs sont (i) la modélisation orientée objet et (ii) l'utilisation d'un langage abstrait compréhensible par l'homme et interprétable par les machines. UML permet de construire plusieurs modèles d'un système, chacun mettant en valeur des aspects différents : fonctionnels, statiques, dynamiques et organisationnels.

Certains diagrammes UML 2.0 seront utilisés pour décrire les fonctionnalités du système lors de l'actuel chapitre 3. Nous nous sommes appuyés sur des diagrammes de cas d'utilisation et sur des diagrammes de séquences pour l'analyse du système. Les premiers, présentés dans ce mémoire, nous ont servi à identifier les fonctionnalités, les deuxièmes à les décrire. Un diagramme global d'interaction du système est aussi présenté pour donner une vision générale du déroulement d'une session.

A l'aide d'un diagramme de contexte, un cas particulier de diagramme de flux des données qui ne fait pas partie des diagrammes UML 2.0 standards, nous montrons une vision générale des composants du système en termes de modules et de rôles nécessaires.

3.2.1.1 Diagramme de Cas d'Utilisation (Use Case Diagram)

Pour définir et organiser les fonctionnalités que l'on doit fournir à la couche application, nous utiliserons des diagrammes de Cas d'utilisation. Ces diagrammes font partie de l'ensemble des diagrammes comportementaux ou diagrammes dynamiques UML 2.0. Ils sont utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Dans un tel diagramme, les utilisateurs sont appelés acteurs (*actors*) ; ils interagissent avec les cas d'utilisation (*use cases*) qui composent le diagramme. On peut ainsi identifier et documenter les fonctionnalités dont les participants du système ont besoin, les relations entre ces fonctionnalités et leur relation avec les acteurs du système.

Le tableau suivant montre les symboles UML utilisés pour les diagrammes de cas d'utilisation de ce chapitre.

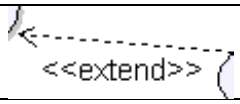
Symbole	Description
	Acteur qui correspond à une personne
	Cas d'utilisation définissant une fonctionnalité
	Relation d'extension entre deux cas d'utilisation
	Relation d'inclusion entre deux cas d'utilisation
	Lien entre un acteur et un cas d'utilisation

Tableau 3.1. Symboles UML utilisés dans les diagrammes de cas d'utilisation

Le premier élément du tableau 3.1 correspond à *l'Acteur*, le symbole utilisé pour représenter les personnes externes au système et qui sont en relation avec le système, soit pour recevoir un résultat de ce dernier, soit pour lui transmettre de l'information.

Le deuxième élément, le *Cas d'Utilisation,* représente une fonctionnalité du système. Pour établir une relation entre les cas d'utilisation, deux flèches sont utilisées : la première, *l'Extension*, montre que la fonctionnalité à l'extrémité de la flèche étend la fonctionnalité située à la source de la flèche, c.à.d. qu'elle hérite du comportement de la fonctionnalité de base mais avec certaines spécialisations. La deuxième flèche correspond à *l'Inclusion*, pour indiquer que la fonctionnalité à la source de la flèche sera appelée par l'autre fonctionnalité à l'extrémité. Ces deux fonctionnalités ont des comportements différents mais complémentaires.

Le dernier élément est utilisé pour montrer le lien entre les *acteurs* intervenants dans le système et les *cas d'utilisation*.

3.2.1.2 Diagramme Global d'Interaction (Interaction Overview Diagram. IOD)

Le diagramme global d'interaction UML 2.0 permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous la forme de diagrammes de séquences. Ils sont une variante des diagrammes d'activités dont les nœuds peuvent être des diagrammes de séquence d'une fonctionnalité. Ils servent à décrire les interactions entre les différentes fonctionnalités que le système offre.

Les notations d'un diagramme IOD sont les mêmes que celles des diagrammes d'activités. On trouve en plus, des opérateurs de choix et d'ordonnancement dans un diagramme IOD. Le tableau 3.2 montre les symboles supplémentaires des diagrammes IOD utilisés dans ce chapitre.

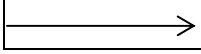
Symbole	Description
	Début de session
	Fin de session
	Fonction
	Choix
	Opérateur parallèle
	Flux

Tableau 3.2. Symboles UML utilisés dans les diagrammes globaux d'interaction

Une session correspond au déroulement d'un ensemble de fonctions. Les deux premiers symboles du tableau 3.2 correspondent au *début* et à la *fin* de session respectivement. Le troisième symbole désigne une fonction au sein d'une session. Une *fonction* peut être un ensemble de plusieurs fonctionnalités du même type, par exemple, la fonction CommunicKeyChange regroupe toutes les fonctionnalités de modification de la clé de session.

Le *choix* est utilisé pour représenter plusieurs alternatives parmi les fonctions à exécuter à un moment donné. Un seul choix est juste exécuté. L'*opérateur parallèle* représente des fonctions qui peuvent s'exécuter en même temps dans le système. Le symbole de *flux* sert à montrer l'ordre et le sens d'exécution des fonctions du système.

3.2.2. Diagrammes de Contexte (Context Diagram)

Un diagramme additionnel aux diagrammes UML 2.0 sera utilisé. Le diagramme de contexte (DC) est un cas particulier de diagramme de flux de données (DFD) qui définit le plus haut niveau des interfaces d'un système. Il décrit les composants de l'environnement d'un système tels qu'ils sont perçus du point de vue des utilisateurs de ce système. A travers ce diagramme, nous allons identifier et documenter les rôles *type* des participants au système, les relations entre ces rôles et la hiérarchie qu'ils occupent au sein du groupe (en fonction des activités associées à ce rôle). Le DC est utilisé pour aider à comprendre le contexte d'exécution du système. Le tableau suivant explique les symboles que nous allons utiliser dans ce type de diagramme.

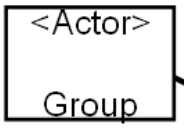
Symbole	**Description**
Group Membership Management	Module du système analysé
 :Administrator	Acteur humain (externe au système modélisé)
	Acteur représentant un système externe au système modélisé ou un ensemble des acteurs
	Relation d'héritage entre acteurs

Tableau 3.3. Symboles utilisés dans les diagrammes de contexte

Le premier symbole du tableau 3.3, le carré, représente un module faisant partie du système étudié. Le deuxième symbole, nommé *Acteur,* est utilisé pour représenter les personnes qui utilisent le système. Le carré avec le mot <*Actor*> est utilisé pour faire des agrégations d'acteurs. Il permet de représenter un ensemble d'utilisateurs du même type. La flèche décrit une relation d'héritage entre acteurs. Quand un acteur hérite d'un autre, il détient les caractéristiques de l'acteur père mais il détient aussi des caractéristiques supplémentaires, qui lui sont propres et spécifiques.

3.3. Structure des groupes

3.3.1. Dimensions dans un groupe

Pour donner une explication la plus simple possible, nous avons décomposé les groupes auxquels nous nous sommes intéressés en utilisant le concept de ***dimension.*** Ceci nous servira à expliquer progressivement les axes du groupe selon lesquels nous allons nous focaliser.

La **première dimension** traite des réseaux avec plusieurs groupes en communication. De façon plus précise, elle donne le nombre de groupes qui sont présents. La confidentialité de l'information doit être garantie à l'intérieur de chaque groupe.

La **deuxième dimension** fait référence aux différentes hiérarchies d'utilisateurs. En effet, un groupe peut avoir à l'intérieur des sous-groupes hiérarchiques que nous avons appelés *classes*. La dimension *hiérarchie des membres* est importante car la confidentialité des messages à l'intérieur de chaque classe hiérarchique doit aussi être garantie. L'intérêt d'introduire des hiérarchies consiste à donner plus de droits à la classe la plus haute et moins de droits en descendant au fur et à mesure dans les classes les plus basses. En termes de confidentialité, un membre qui appartient à la classe la plus haute pourra comprendre les messages de sa classe et aussi ceux des classes inférieures. Les membres les moins gradés, qui font partie de la classe la plus basse, ne pourront comprendre que les messages échangés dans leur classe.

La **troisième dimension** traite des groupes fusionnés. Elle est importante pour nous puisque les systèmes que nous étudions possèdent des fonctionnalités nécessaires aux groupes fusionnés. En premier lieu, des fonctionnalités de fusion et de séparation doivent être fournies à l'application ; en second lieu, un groupe faisant partie d'un groupe fusionné existe vis-à-vis de deux types de communications distincts, la communication au sein de son groupe d'origine mais aussi celle au sein de son groupe fusionné. Il est important de remarquer que le processus de fusion est itératif car il est possible de faire des fusions consécutives.

La figure 3.1 décrit les trois dimensions qui peuvent être prises en compte lorsque l'on parle de communication de groupes. Ce niveau de complexité n'est pas évident à gérer dans l'analyse, la modélisation et la vérification des systèmes de communication de groupes sécurisés.

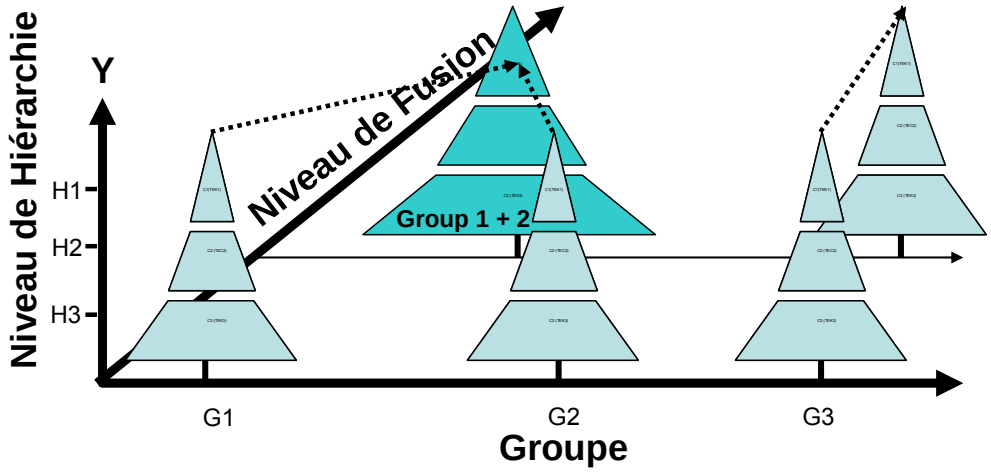


Fig. 3.1. Les trois dimensions d'une structure des groupes

La fig. 3.1 montre un exemple de session avec, dans un premier temps, trois groupes actifs : G1, G2 et G3 dans l'axe **Groupe**. La hiérarchie des groupes se compose de trois classes (H1, H2 et H3) qui apparaissent dans l'axe **Niveau de Hiérarchie**. Dans un deuxième temps, nous pouvons observer un groupe issu de la fusion des groupes G1 et G2 en parallèle avec le groupe G3 dans son état de base. Ceci est montré par l'axe **Niveau de fusion**. Chaque groupe qui participe à la fusion a la capacité soit de communiquer à l'intérieur du nouveau groupe fusionné, c.à.d. de l'union des groupes G1 et G2, soit de communiquer seulement en son sein. Cette communication peut s'effectuer car les groupes fusionnés auront toujours deux configurations : la configuration d'avant la fusion et la nouvelle configuration après la fusion.

3.3.2. Notion de rôle des utilisateurs dans un groupe

La composition des groupes pour des communications sécurisées s'appuie sur une organisation en rôles. En effet, cette organisation nous semble assez proche du niveau applicatif et suffisamment liée aux caractéristiques des systèmes que nous souhaitons décrire (des groupes supportant des hiérarchies d'utilisateurs humains) ; de plus, ce découpage est assez général pour représenter des groupes structurés sous forme hiérarchique (membres avec des rôles de différente importance) ou structurés sous forme plane (membres avec des rôles identiques). Un rôle représente pour un membre l'accès à un ensemble de ressources et confère à un membre un ensemble de droits et de fonctions. Chaque rôle est lié à la mise en œuvre d'une ou de plusieurs fonctionnalités. Les rôles que nous avons identifiés apparaîtront par la suite dans les diagrammes de cas d'utilisation des sous-sections suivantes.

La fig. 3.2 montre, au travers d'un diagramme de contexte, les principales composantes du système ainsi que ses principaux acteurs. Deux ensembles de fonctionnalités qui font partie de notre étude sont montrés : celui qui gère la sécurité pour la communication des groupes (*Group Communication Key Management)* et celui qui gère la composition des groupes (*Group Membership Management)*. Une identification des principaux rôles participant à une session est aussi montrée par des classes d'acteurs UML.

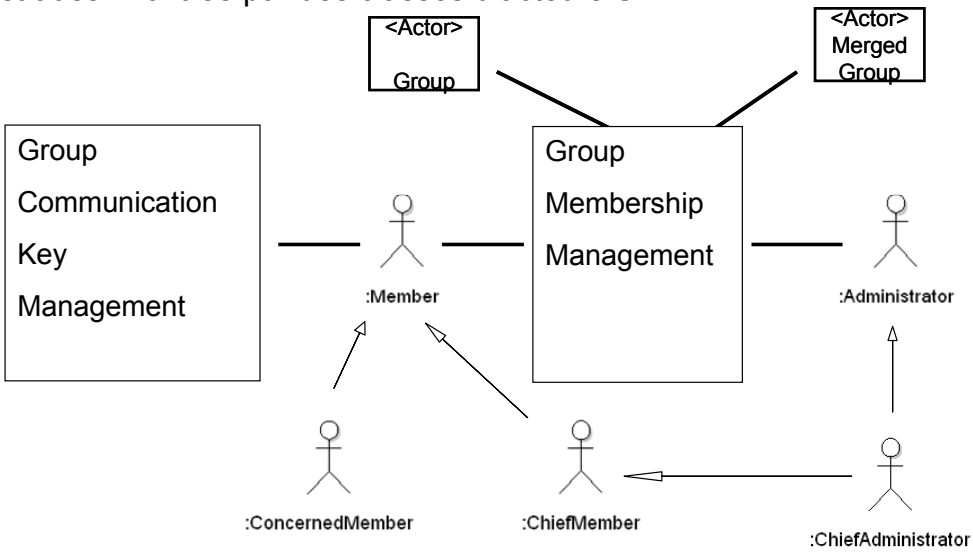


Fig. 3.2. Diagramme de contexte des fonctionnalités de gestion de la communication des groupes

Pour certaines fonctionnalités du système, un regroupement des membres est nécessaire. Par exemple, l'< acteur > *Group,* qui apparaît dans la fig. 3.2 correspond à l'abstraction d'un ensemble de membres. Ce groupe sera affecté dans son ensemble par une certaine fonctionnalité. Une telle fonctionnalité peut être le renouvellement automatique des clés de session qui s'applique à un *Groupe* actif complet. Un autre acteur de regroupement est *« MergedGroup »* qui correspond à un ensemble de groupes qui se sont associés pour effectuer une communication globale. Une fonctionnalité qui peut s'appliquer à un *MergedGroup* pour le séparer est le *Split*.

Les acteurs UML, représentés sous la forme de personnages dans la fig. 3.2, correspondent à une seule personne humaine avec au moins un rôle spécifique dans l'application. Ces personnages sont essentiels lors du déroulement d'une session. Certains d'entre eux peuvent posséder plus d'un rôle ; ils héritent pour cela des capacités des autres membres au moyen d'une relation d'héritage. Ceci introduit un degré de flexibilité supplémentaire dans l'attribution des rôles. Par exemple, nous pouvons avoir le cas où un membre de rôle *Administrator* et un autre membre de rôle *ChiefMember* sont présents à la fois sur terrain. Mais, un autre cas peut admettre une seule personne qui possède les deux rôles en même temps, au travers du rôle *ChiefAdministrator*.

Les rôles identifiés et montrés sur la fig. 3.2 sont les suivants :

Member : Une personne détient ce rôle lorsqu'elle appartient à un groupe (soit de base ou fusionné) et lorsqu'elle participe à une session.

ConcernedMember : Ce rôle est pris par la personne qui est principalement concernée par l'action d'une fonctionnalité. Par exemple, lors d'une *entrée dans un groupe*, c'est celui qui entre.

Administrator : Ce rôle confie la responsabilité d'une session. Il permet de définir la structure des groupes, de démarrer et de finir une session. Il confère aussi la responsabilité d'agir en présence des intrus ou des membres qui ont un mauvais comportement. L'administrateur peut en plus ordonner la fusion des groupes actifs. Dans certains cas, ce sont les chefs des groupes qui veulent fusionner qui autorisent, eux-mêmes, une fusion.

ChiefMember : Le chef du groupe fait partie des membres. Il représente une autorité d'attributs vis-à-vis des membres dans son groupe, car il contrôle les utilisateurs pour tout mouvement à l'intérieur du groupe comme des entrées, des sorties, des montées en grade ou des descentes. Il est en charge de maintenir la cohérence de la composition de son groupe. Il autorise et gère aussi des changements entre des groupes, notamment la fusion demandée et la séparation des groupes. Le *ChiefMember* est aussi en charge de la gestion de la clé de session. Il hérite du rôle *member* car il fait partie de la session mais, en tant que chef, il possède plus de droits qu'un simple membre. Il a aussi la responsabilité d'agir en présence des intrus ou des membres qui ont un mauvais comportement.

A la fin, l'*Autorité de Certification* est une entité fixe qui ne participe pas à la dynamique du protocole. Elle ne fait pas partie de nos modélisations car elle intervient hors ligne. Elle se charge d'accorder les Certificats d'Identité aux membres. L'assignation est faite avant de commencer une session. Si nécessaire, durant l'exécution d'une session, des actualisations éventuelles des Certificats d'Identité peuvent s'effectuer mais ils se font de façon manuelle. Elle représente de manière globale une autorité pour la gestion des certificats dans le sens où elle peut aussi contrôler la gestion des certificats d'attributs, même si

elle a la possibilité de déléguer cette responsabilité aux chefs de groupes [L2.3]. Dans notre cas, nous proposons que le *chiefMember* prenne cette responsabilité si cela devient nécessaire.

ChiefAdministrator : Dans le cas où un administrateur de la session n'existe pas, un membre du groupe prend deux rôles, celui du chef et celui de l'administrateur.

Tous les services de la fonctionnalité de gestion de groupes utilisent des certificats pour leur exécution. Les Certificats d'Identité (CI) serviront à identifier les membres d'un groupe. Les Certificats d'Attributs (CAt) associent les rôles aux membres de la session. Ces membres se répartissent dans leurs classes respectives et ils détiennent les droits correspondant aux rôles (voir chapitre 4, sous-sections 4.4.2.1 et 4.4.2.2 pour une spécification plus précise).

Le travail d'identification de la structure des groupes, des rôles des participants dans le groupe et des fonctionnalités des protocoles sécurisés, se déroule pendant une session active, c'est-à-dire lorsque les membres sont présents dans les groupes et qu'ils communiquent entre eux. Nous n'avons pas décrit les fonctionnalités nécessaires en dehors des sessions (par ex. la création d'une session et la formation des groupes) car elles ont moins de contraintes en terme d'interactivité, de garanties temporelles et de sécurité. Cependant, ces fonctionnalités font partie du système et nous les montrerons dans le diagramme global d'interaction de la sous-section 3.7.

La fig. 3.3 montre un exemple d'une session avec quatre groupes (brigades) actifs. Les quatre brigades n'ont pas le même nombre de membres, mais leur structure hiérarchique reste similaire, ce qui autorise, pour notre cas, des fusions entre groupes.

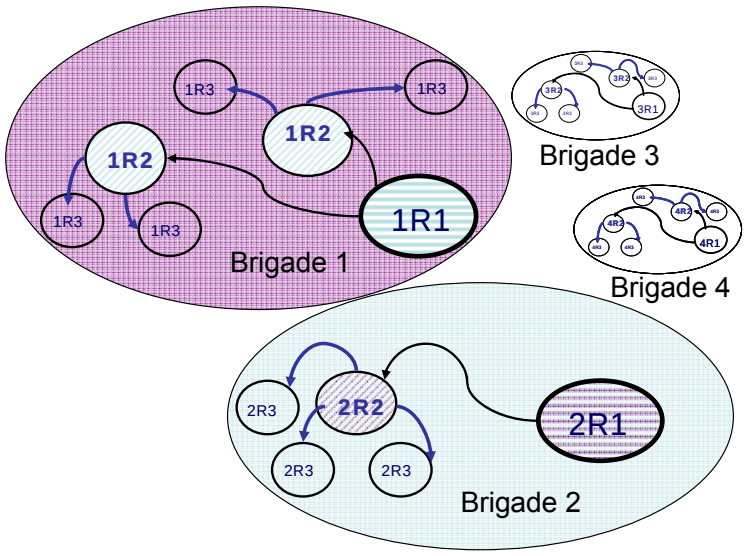


Fig. 3.3. Exemple d'une structure de Groupes Hiérarchiques

Les brigades de la fig. 3.3 possèdent une structure hiérarchique unifiée qui se compose de trois classes : iR1, iR2 et iR3. Par contre, leur composition diffère en termes de nombre de membres. Par exemple, la brigade 1 possède un chef dans la classe 1, deux sous chefs, et deux membres qui sont sous les ordres de chaque sous-chef. Par contre, la brigade 2 a un seul chef, un seul sous-chef et trois membres sous les ordres de ce sous-chef.

Les groupes dans la session de la fig. 3.3 ne présentent pas encore de fusion, ils sont dans leur état de base.

3.4. Fonctionnalités pour la gestion des groupes

Il s'agit des fonctionnalités qui traitent de tout mouvement d'un membre, soit en modifiant la composition interne d'un groupe, soit en changeant la connectivité entre des groupes.

L'identification des fonctionnalités faite dans cette sous-section correspond aux besoins détectés lors de la phase d'analyse du système.

3.4.1. Composition interne des groupes

Les fonctionnalités de gestion des groupes qui traitent de leurs changements intérieurs seront expliquées dans cette sous-section. La fig. 3.4 montre le diagramme de cas d'utilisation regroupant ces fonctionnalités.

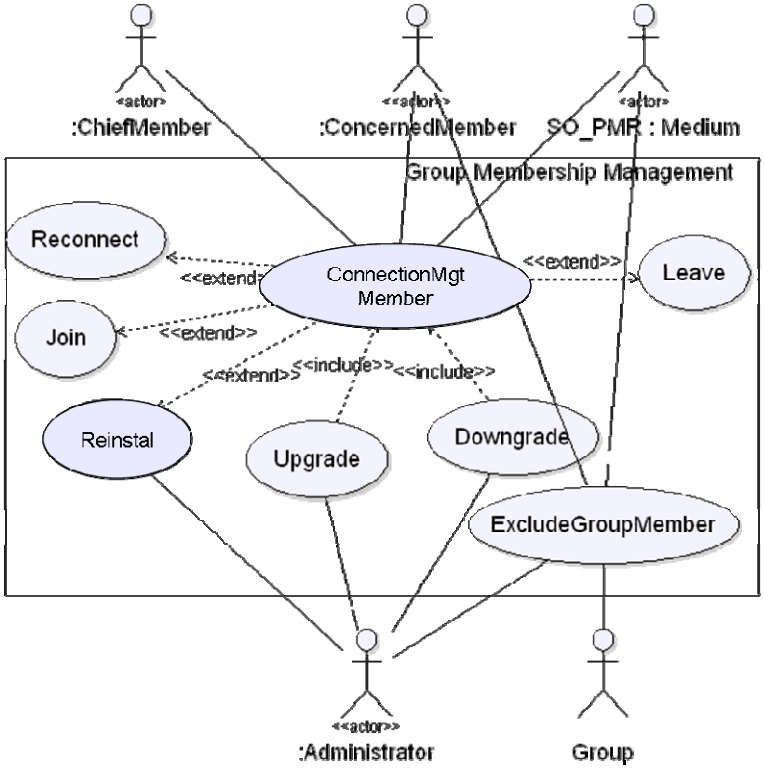


Fig. 3.4. Diagramme de cas d'utilisation des fonctionnalités de gestion intra-groupe

Les fonctionnalités élémentaires intra-groupes du système comprennent : l'entrée *« Join »* d'un membre,, la sortie *« Leave »* d'un membre, la descente *« Downgrade »* dans la hiérarchie, la montée dans la hiérarchie *« Upgrade »,* la reconnexion *« Reconnect »* applicable lorsqu'un ancien membre du groupe a perdu la connexion, et finalement la fonctionnalité *« Reinstal »* si le membre concerné avait été exclu. Ces fonctionnalités utilisent le fragment *« ConnectionMgtMember»* (administration de la connexion du membre) qui contient l'action de connexion à un groupe et qui gère cette connexion, soit pour connecter, soit pour déconnecter un membre d'une classe spécifique.

« ExcludeGroupMember », qui traite de l'exclusion d'un membre, est une fonctionnalité qui fait partie de l'ensemble considéré. Elle n'a pas besoin de la fonctionnalité *« ConnectionMgtMember».*

Nous rappelons que les fonctionnalités considérées prennent leur place dans des groupes déjà actifs. Une description détaillée de ces fonctionnalités est donnée dans la suite.

3.4.1.1 Fonctionnalité Elémentaire ConnectionMgtMember

L'identification de cette fonctionnalité comme un sous ensemble commun à une majorité d'autres fonctionnalités a été d'une grande importance pour la spécification, la modélisation et la vérification du système. Elle factorise le processus de traitement de la connectivité d'un membre pour toutes les fonctionnalités de gestion intra-groupe, sauf pour l'exclusion d'un membre, comme montré dans le diagramme de la fig. 3.4.

Avec cette fonctionnalité, nous pouvons diviser les fonctionnalités qui se servent d'elle en deux groupes : celles qui représentent juste une extension de cette fonction de base et celles qui l'utilisent mais avec leur propre procédure.

Les fonctionnalités qui étendent « ConnectMgtMember » suivent la même procédure de connexion/déconnexion sauf que les certificats à vérifier et les résultats varient. Il s'agit du : *join, leave, reconnect, reinstat,* qui donnent à un utilisateur le droit d'entrée ou de sortie dans la classe d'un groupe qui lui correspond. Pour chaque fonctionnalité, le chef vérifie les certificats nécessaires et peut autoriser ou non l'entrée/sortie du membre. Ces fonctionnalités apparaissent dans la fig. 3.4 et elles sont liées à la fonctionnalité *« ConnectMember »* à travers la relation « *extend »* pour montrer qu'elles héritent du *« ConnectMgtMember »* mais qu'elles ajoutent certaines particularités.

Le deuxième groupe correspond aux fonctionnalités qui ont leur propre procédure mais qui, à un moment donné, ont besoin de gérer la connexion du membre concerné. En effet, pendant le *« Upgrade »* et le *« Downgrade »*, un travail d'autorisation initial doit avoir lieu pour monter ou descendre respectivement, et par la suite, pour effectuer la connexion à la nouvelle classe. Ces fonctionnalités apparaissent dans la fig. 3.4 avec un lien *« include »* vers la fonctionnalité *« ConnectMgtMember »,* la procédure générale de gestion de la connexion.

3.4.1.2 Join

Une fois la session ouverte, les groupes actifs sont déjà composés. La fonctionnalité de *join* va donc permettre à un nouveau membre de se connecter la première fois dans une session déjà existante. Il appartiendra à une certaine classe dans un groupe.

Pour réussir une connexion dans un groupe, le membre doit s'authentifier avec son certificat d'identité auprès du chef du groupe et ce chef doit vérifier que le membre entre bien dans la bonne classe. Une fois autorisé, le chef envoie au membre la clé de communication ainsi qu'un certificat d'appartenance avec les droits qui lui correspondent.

Le certificat d'identité interdit les entrées clandestines dans le *join* comme, par exemple, des membres antérieurement exclus.

Il est important de considérer le cas d'une actualisation de la clé de session puisque, si un membre entre au milieu d'une session et qu'on lui donne la clé actuelle (sans l'actualiser), ce membre peut, en utilisant cette nouvelle clé, comprendre les messages échangés avant son entrée et chiffrés avec cette même clé de session.

3.4.1.3 Leave

Cette fonctionnalité permet à un membre de quitter la session dont il fait partie. Nous considérons qu'il est important que le chef soit informé des sorties qui affectent la composition du groupe, pour qu'il garde une vision cohérente du groupe.

Informer de la sortie d'un membre est important pour une deuxième raison, car il pourrait arriver qu'une sortie cause une inconsistance au niveau de la composition d'une session. Par exemple, la sortie d'un chef du groupe doit s'effectuer en laissant un autre chef substitut à sa place. En autorisant la sortie d'un membre, il faut que le chef réalise les actions nécessaires pour s'assurer que le groupe ait suffisamment de membres et de ressources restants et que la session reste cohérente.

Nous proposons donc qu'une réponse de la part du chef soit envoyée au membre qui quitte le groupe pour l'informer qu'il peut en effet sortir. Le chef, de sa part, doit aussi enregistrer cette sortie.

3.4.1.4 Upgrade

Une montée de classe se passe quand un chef a besoin qu'une personne monte dans la hiérarchie. Par exemple, cette fonctionnalité est utilisée dans le cas où le chef lui-même doit sortir du groupe et qu'il a besoin d'un chef remplaçant (une personne prise parmi les membres de la session).

Une montée est demandée directement au membre concerné par le chef du groupe. Le membre qui monte détient déjà un certificat d'identité et un certificat d'appartenance. Il va alors les utiliser auprès du chef du groupe comme « ticket d'entrée » pour la nouvelle classe. De façon plus spécifique, en coordination avec le chef du groupe, son certificat d'identité, qui servira à l'authentifier, et son certificat d'appartenance, lui donneront le droit de la montée et de l'accès à la nouvelle classe.

3.4.1.5 Downgrade

Cette fonctionnalité est utilisée pour faire descendre dans la hiérarchie un membre d'un groupe.

L'intérêt étant de continuer à maintenir la confidentialité des messages à l'intérieur des classes existantes, la descente dans la hiérarchie d'un membre en communication implique un changement au niveau des clés de communication.

Un *downgrade* de classe peut se passer dans deux cas :

1. Si l'on n'a plus besoin d'un membre dans sa hiérarchie actuelle.
2. Si son groupe a perdu la confiance qu'il avait en lui.

La solution retenue devra supprimer l'ancienne clé de session du membre qui descend pour qu'il ne puisse plus comprendre les messages de son ancienne classe et garantir ainsi la confidentialité entre classes. Bien entendu, ce membre devra récupérer la clé de communication de sa nouvelle classe. Cependant, la suppression de l'ancienne clé n'est pas une problématique évidente en termes de sécurité. En effet, dans un protocole de groupes sécurisés, l'actualisation des clés s'appuie sur la possession des clés antérieures. Dans ce cas là, le membre qui descend détient l'information ancienne et il pourrait bien obtenir toute l'actualisation des clés de son ancienne classe, ceci créant un problème de garantie pour la confidentialité des messages.

3.4.1.6 ExcludeGroupMember

La fonctionnalité « ExcludeGroupMember» traite de l'exclusion d'un membre du groupe, ce qui représente un cas très particulier.

Une exclusion d'un membre veut dire que l'on a perdu la confiance en ce membre. La confidentialité des groupes étant basée sur une crypto période de changement de la clé de session, le groupe ne peut pas attendre le prochain renouvellement de clé, car le membre exclu continuerait à comprendre l'information qui provient de son ancienne classe. Donc, un changement immédiat de la clé de session doit se faire, même au détriment de la bande passante disponible pour l'application.

Nous proposons que le chef soit la seule autorité ayant le droit d'exclure un membre. Dans le cas où un membre normal détecterait un membre malicieux, ce membre normal doit informer le chef de groupe pour qu'il prenne la décision d'exclusion. Ceci donne plus de sécurité au système, car le contrôle de l'exclusion est seulement confié au chef du groupe. En effet, ce dernier doit maintenir une liste des membres exclus dans le groupe. De plus, il est en charge de maintenir une vue actualisée de la structure de son groupe.

Après l'exclusion d'un membre, le groupe doit être protégé contre des tentatives d'entrée frauduleuses de ce membre, par la mise en œuvre d'autres fonctionnalités comme la montée, la descente ou bien la reconnexion dans un groupe.

Dans la fig. 3.4 l'exclusion apparaît comme une fonctionnalité isolée car elle implique un changement de la clé de session pour que le membre exclu n'ait plus accès à la session active.

3.4.1.7 Reinstal

La seule possibilité de faire revenir un membre exclu dans une session active, se fait au travers de la réinstallation (« *reinstal* ») du membre. Seul le chef peut autoriser cette reconnexion. Il a besoin d'actualiser sa liste des membres exclus en enlevant le membre concerné. Il remet à jour aussi la vue de la structure de son groupe.

Pour garantir l'authenticité d'une réinstallation et, étant donné que peu de réinstallations des membres auparavant exclus auront lieu, nous proposons que le membre soit réintégré en ayant un nouveau certificat d'identité. L'autorité qui donne au membre exclu de nouveau l'accès à la session, doit donner au membre un nouveau certificat qui l'autorise à se réinstaller. L'octroi d'un certificat d'identité ne s'effectue pas par le réseau de communication existant mais par des moyens hors ligne, notamment en déplaçant l'autorité indiquée vers l'utilisateur pour lui donner ce certificat. Comme il n'existe pas beaucoup d'exclusion, cette solution ne pénalise pas la performance du système, et elle représente un renforcement dans sa sécurité.

3.4.1.8 Reconnection

C'est une fonctionnalité qui permet d'améliorer la communication face aux déconnexions que les utilisateurs actifs peuvent subir pendant le déroulement d'une session. L'intérêt est de fournir aux membres déconnectés une façon de rejoindre leur ancien groupe de communication en demandant de nouveau la clé de communication.

Pour avoir droit à une reconnexion, le membre doit présenter les deux certificats qu'il avait avant la dernière déconnexion : son certificat d'identité qui sert à l'authentifier et son certificat d'appartenance avec lequel il était un membre actif d'une classe dans un groupe.

Une reconnexion doit être validée par le chef du groupe. Lui seul peut vérifier l'authenticité de la demande du membre. Une fois la reconnexion autorisée, le chef donne la clé de communication du groupe au membre demandeur.

3.4.2. Gestion de la connectivité entre des groupes

Ces fonctionnalités traitent des relations entre plusieurs groupes en communication. Elles font intervenir deux ou plusieurs groupes lors de leur exécution. La fig.3.5 montre le diagramme de cas d'utilisation regroupant ces fonctionnalités.

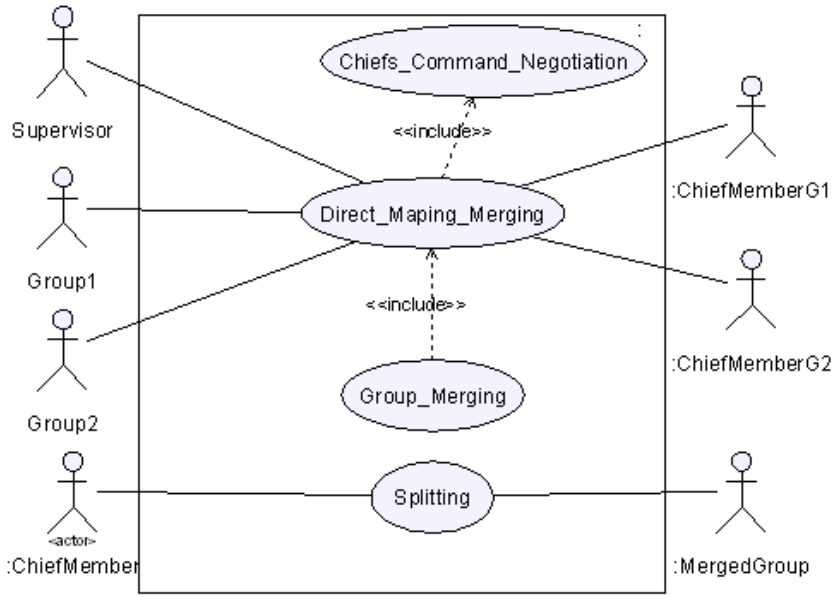


Fig. 3.5. Diagramme de cas d'utilisation des fonctionnalités de gestion inter-groupes

Ce diagramme de cas d'utilisation comprend les opérations fondamentales de fusion et de séparation des groupes.

3.4.2.1 Merging

Les groupes qui fusionnent utilisent la fonctionnalité « *Group_Merging* » pour créer la nouvelle structure. La fonction *« Direct_Maping_Merging* » est utilisée par les groupes qui possèdent la même structure hiérarchique et qui souhaitent fusionner entre eux. Les membres des groupes à fusionner feront partie d'un seul nouveau groupe et leur niveau hiérarchique sera le même que celui de leur groupe de base. Le cas de groupes de structures hiérarchiques différentes n'a pas été envisagé. Il nécessiterait de passer par une étape supplémentaire qui créerait une structure hiérarchique commune.

A partir des chefs des groupes qui fusionnent, la fonction « *Chief_Command_Negotiation* » permet de ne garder qu'un seul chef pour le groupe fusionné. Après cette négociation entre les chefs, un seul chef reste comme chef leader dans le groupe fusionné. Les autres chefs prennent l'état de membres normaux.

Dans le processus de fusion, un renouvellement des clés est fait et tous les membres détiendront la nouvelle clé de session. Cette clé permettra de déchiffrer les messages destinés au nouveau groupe fusionné. Ce sera après une séparation que les groupes fusionnés recommenceront leur communication ancienne.

Chaque groupe de base garde sa configuration d'origine puisqu'une séparation postérieure à la fusion doit ramener les groupes fusionnés dans leur configuration de base. Pour cela, les groupes qui fusionnent gardent un historique de chaque groupe fusionné intermédiaire dont ils ont fait partie. Ils peuvent donc participer à des séparations successives avec la fonctionnalité « *Split* » et retrouver, à chaque split, l'état et l'information de leurs anciens groupes avant la dernière fusion.

L'opération de fusion peut se dérouler de façon récursive : Plusieurs fusions successives peuvent s'accumuler, plusieurs séparations peuvent remettre les groupes participants dans une fusion dans leur état précédent.

La communication au sein des groupes de base ne sera plus nécessaire mais elle reste possible en utilisant l'ancienne clé de session.

La fig. 3.6 montre un exemple d'une fusion de groupes. L'étape 1 montre les groupes d'origine avec leur structure de base. Dans l'étape 2, le chef du groupe « Pompier 2 » devient le chef du nouveau groupe fusionné, alors que l'autre chef devient un membre normal. Aucune nouvelle classe n'est ajoutée à la hiérarchie résultante. Nous visualisons dans cette étape le nouveau groupe créé.

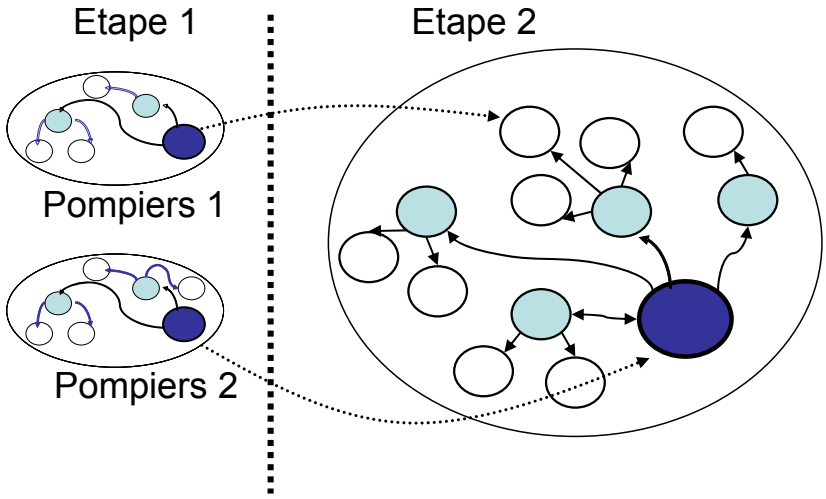


Fig. 3.6. Fusion des groupes appartenant à la même organisation

3.4.2.2 Spliting

L'avantage de garder avant toute fusion la configuration des groupes qui fusionnent, permet de réaliser la séparation des groupes avec un simple avertissement de ce *split* aux membres du groupe fusionné. Puis, les groupes reviendront à leur état avant la fusion.

Ceci est possible car tout groupe fusionné garde sa configuration d'avant et tous ses éléments secrets de participation dans sa configuration antérieure à la fusion. Ainsi, les membres doivent juste reprendre leurs éléments initiaux et continuer à communiquer dans leur groupe de base.

3.5. Fonctionnalités pour la gestion des communications de groupes

La protection des données échangées dans le réseau est faite de manière à ce que personne hors du groupe ne puisse comprendre l'information échangée. Cette protection d'information est un des principaux problèmes d'étude dans la communication des groupes.

Comme présenté dans l'état de l'art, une évolution des techniques de protection des échanges nous fait passer de communications point-à-point à des communications multipoint, broadcast, voire à des communications entre plusieurs utilisateurs dans différents groupes en communication au sein d'une même session. La confidentialité doit être maintenue à l'intérieur de chaque groupe. Ces techniques de protection doivent aussi supporter les changements dans la composition des groupes et des hiérarchies d'utilisateurs.

3.5.1. Confidentialité de communication à l'intérieur de chaque groupe, par utilisation de clés de session par groupe

Les fonctionnalités présentées dans cette sous-section fournissent les mécanismes de contrôle d'accès et de confidentialité des échanges dans les groupes pendant la durée d'une session, y compris lors de son évolution dynamique. Une gestion sécurisée et efficace de ces éléments est nécessaire pour garantir le bon fonctionnement de ces mécanismes.

Adapter les techniques des émissions sécurisées point-à-point pour des communications de groupes a été nécessaire pour minimiser la charge de gestion des éléments de sécurité utilisés. Le chiffrement symétrique, avec une seule clé de session, a été adopté dans les communications des groupes afin de protéger les messages échangés entre les éléments d'un groupe. En effet, l'emploi de nombreuses combinaisons de couples de clés asymétriques nous semble difficile à gérer pour des diffusions de groupes. Ce choix évite une gestion inadéquate et excessive de nombreuses clés assignées à chaque utilisateur.

Dans le chiffrement symétrique, l'intérêt est d'assigner à chaque groupe en communication, une clé de session qui sert à protéger les données échangées en son sein. Cependant, une gestion bien sécurisée et efficace de cette clé de session doit être garantie, tâche qui n'est pas facile à effectuer.

Un juste équilibre doit être trouvé entre la sécurité et la performance du système que l'on cherche à offrir à l'application. Mettre en œuvre beaucoup de techniques pour garantir un haut niveau de sécurité peut dégrader la performance du système. De façon inverse, essayer de produire un système très performant peut affaiblir le système en termes de sécurité.

La dynamique des groupes de communication complique davantage la tâche, puisque le mécanisme de sécurité choisi doit continuer à garantir la confidentialité des échanges face à des changements des membres actifs du groupe. Par exemple, si un membre se fait exclure d'un groupe, une modification de la clé de session appartenant au groupe concerné doit s'effectuer afin que le membre exclu ne puisse plus comprendre l'information appartenant au groupe.

Plus la dynamique augmente, plus la gestion des éléments secrets nécessaires à la protection des échanges devient lourde.

Le système doit donc assurer un équilibre, potentiellement négociable, entre la dynamique des groupes, la sécurité et la performance qu'il s'engage à maintenir.

3.5.2. Confidentialité des communications à l'intérieur de chaque classe

En considérant la hiérarchie des utilisateurs, il nous a semblé nécessaire de permettre aux membres des niveaux les plus hauts de comprendre l'information des niveaux inférieurs. Pour cela, une création des clés par niveau hiérarchique représente une bonne solution.

Une génération des clés très spécifique à l'intérieur de chaque groupe a donc été retenue. En utilisant la clé de session comme première clé du niveau hiérarchique le plus haut, nous pouvons générer les clés suivantes. La clé « *clé_N »*, appartenant à une certaine hiérarchie *N* va se calculer en appliquant une fonction à sens unique *f(x)* à la clé « *clé_N-1 »* de la hiérarchie N-1, hiérarchie immédiatement supérieure à celle considérée [LIV 2.5].

Une fonction à sens unique *f(x)* est une fonction qui peut être aisément calculée, mais difficile à inverser, c.à.d. qu'il est difficile de calculer la donnée d'entrée à partir de la seule donnée de sortie. Les fonctions à sens unique sont utilisées en cryptographie asymétrique et dans les fonctions de hachage cryptographiques [WIK 07].

Ce type de fonction a été proposé pour que tout membre d'une certaine classe, en ayant sa propre clé et en connaissant la fonction *f(x),* puisse calculer les clés de communication des classes inférieures. Il pourra ainsi comprendre les messages échangés dans sa classe mais aussi les messages des classes inférieures. Grâce à la difficulté en termes de temps et de puissance de calcul nécessaire à l'inversion des fonctions à sens unique, un membre d'une classe inférieure ne peut pas générer la clé d'une classe qui lui est supérieure.

La formule pour effectuer le calcul des clés de communication dans une certaine classe est la suivante :

Soit :
- w : le numéro des groupes dans la session.
- x : le numéro de classes dans un groupe.
- G_i : le groupe i en communication. Pour i : 1..w.
- C_j : la classe j. Pour j : 1..x.
- $k_{j\,i}$: La clé de la classe j, du groupe i.
- f(x) : Fonction à sens unique

La clé d'une classe est calculée comme suit :

$k_{j\,i} = f^{j-1}(k_{1\,i})$ ou bien
$k_{j\,i} = f(k_{(j-1)\,i})$

Ainsi, pour qu'un membre dans le groupe puisse comprendre les messages d'une classe inférieure à la sienne, il lui suffit d'appliquer la fonction *f(x)* à sa propre clé autant de fois que l'on a d'écart de niveau entre la classe du membre et la classe inférieure.

Une fois les clés des classes inférieures à sa classe calculées, le membre pourra comprendre les messages appartenant à sa classe mais aussi aux classes moins hautes. La fig. 3.7 montre un exemple de distribution de deux messages dans un groupe hiérarchisé.

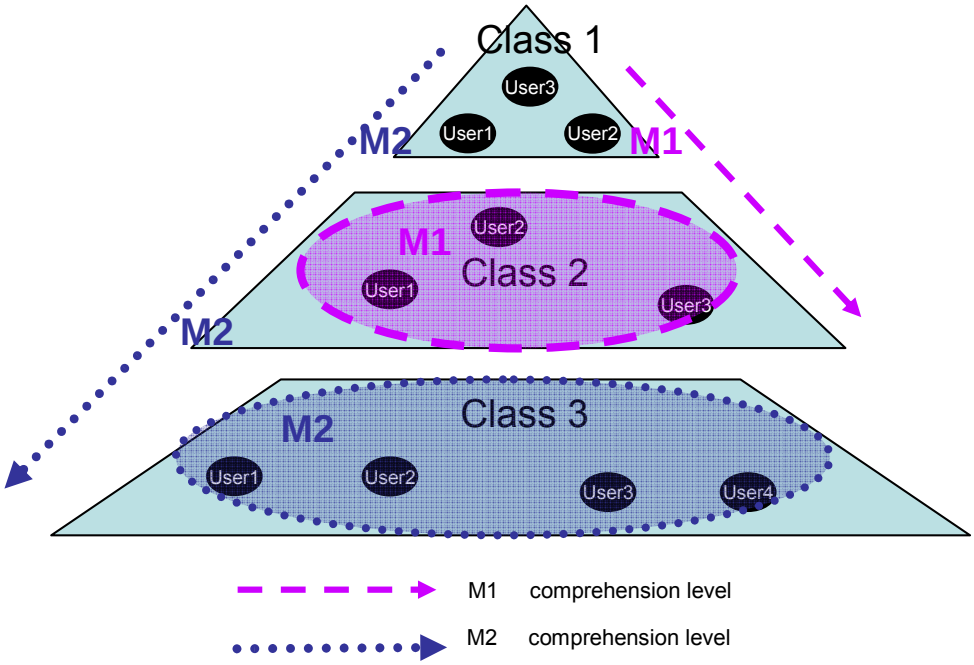


Fig. 3.7. Distribution des messages dans un groupe hiérarchisé

Dans la fig. 3.7 le message M2 est envoyé dans le réseau par l'utilisateur *« user 1 »* de la classe *« class 1 »*. Chiffré avec la clé symétrique de la classe 3, ce message sera directement compris par tous les membres de la classe 3. Ensuite, avec la structure choisie de clés hiérarchiques, les membres de la classe 2 et de la classe 1 pourront calculer la clé de la classe 3 et ils pourront aussi comprendre le message M2. Cette compréhension est représentée par la flèche pointillée.

Dans la même figure, le message M1 est envoyé dans le réseau et il sera directement compris par tous les membres de la classe 2, car il est chiffré avec la clé symétrique de cette classe. Les membres de la classe 1 pourront calculer la clé de la classe 2 et ils comprendront aussi le message M1, s'ils le souhaitent. Cependant, les membres de la classe 3 ne pourront pas calculer la clé de la classe 2 puis qu'ils sont dans une classe moins haute, ils ne pourront donc pas comprendre le message M1. La compréhension de M1 est représentée par la flèche hachurée.

3.5.3. Gestion de la clé de session

Les fonctionnalités présentées dans cette sous-section doivent assurer le niveau de sécurité choisi, en combinaison avec les mécanismes de confidentialité utilisés durant les communications de groupes. En effet, des mécanismes plus sophistiqués de protection des messages ne serviront à rien si les secrets dont ils ont besoin pour fonctionner peuvent être facilement compromis, ou interceptés. Ces mécanismes ont besoin d'une gestion sécurisée des secrets nécessaires.

La liste suivante nomme, de manière générale, les secrets dont les mécanismes de protection des messages ont besoin :

1). Une clé de session Trafic Encryption Key (TEK) par groupe, qui sert à protéger les messages échangés. La clé de session sera gérée par le chef du groupe ;

2). Des clés appelées Key Encryption Key (KEK), qui servent à protéger les clés de session pendant leur manipulation et leur actualisation. On trouve soit une seule clé, soit un ensemble de clés sous forme de vecteur ;

3). Des fonctions de hachage ;

4). Des fonctions pour le calcul des clés des classes du groupe ;

5). Des certificats ;

6). Des éléments pour la gestion des indices pour les messages

Les secrets correspondant aux points 2 à 5 sont des éléments donnés aux membres avant de commencer une session. Il s'agit des éléments fixes dans l'application qui ne sont pas gérés pendant le déroulement de la session. Les éléments du point 6 font partie du système : ils n'ont pas non plus besoin d'une gestion spécifique. Par contre, l'élément du point 1 (la clé de session TEK), indispensable pour la sécurité du système, est dynamique lors du déroulement d'une session. Il s'agit d'une clé partagée entre tous les membres du groupe, ce qui représente l'élément cœur des mécanismes de protection des messages. La gestion de cette clé de session devient compliquée et l'assurance de son intégrité et de sa confidentialité doit être garantie vis-à-vis de la présence des éléments malicieux. Nous rappelons que la dynamique de cette clé doit s'adapter à la dynamique des groupes.

En jouant avec tous ces facteurs, les fonctionnalités de cette sous-section ont pour but la gestion effective de cette clé de session. Cette clé doit être donnée pour la première fois aux utilisateurs qui entrent dans une session. Ensuite, elle s'actualise en fonction de certains changements dans la composition des groupes, mais aussi elle se renouvelle périodiquement.

Cette clé de session doit rester robuste tout en prenant le moins possible de bande passante lors de son actualisation. Autrement dit, la clé de session est changée uniquement et chaque fois qu'une modification du groupe peut compromettre le niveau de sécurité.

En premier lieu, nous nous appuierons sur le concept de confidentialité Pré-Adhésion et Post-Résiliation, où le résultat sera la diminution d'utilisation de la bande passante en éliminant au maximum des messages de contrôle non nécessaires pendant des actualisations de la clé de session. Les confidentialités Pré-Adhésion et Post-Résiliation sont un concept utilisé dans les communications de groupes avec l'idée que certains changements dans la composition des groupes viennent du simple besoin d'intégrer ou de faire sortir des membres auxquels on fait confiance. De ce fait, un ensemble de mouvements dans un groupe peut s'effectuer sans problème avant de changer sa clé de session. Une explication plus complète est donnée dans le chapitre suivant.

En deuxième lieu, la confidentialité dans le groupe doit exclure des membres qui ne sont plus désirés. La clé de session et celles des classes seront actualisés immédiatement après l'événement origine de cette actualisation.

La fig. 3.8 montre le diagramme de cas d'utilisation regroupant les fonctionnalités pour la gestion de la (des) clé (s) de session.

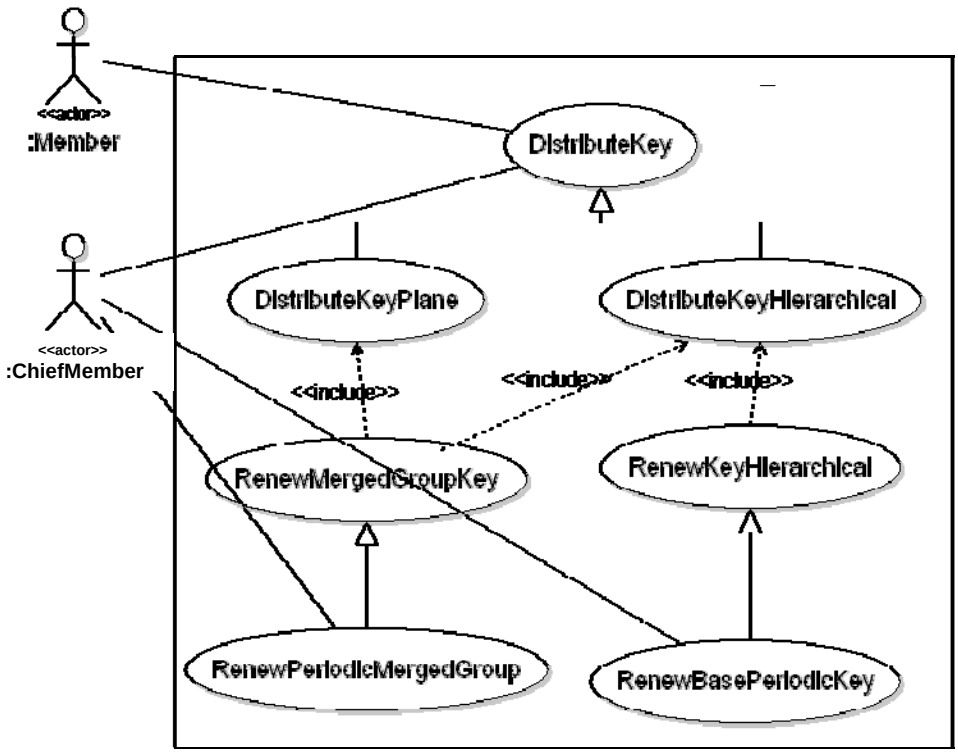


Fig. 3.8. Diagramme de cas d'utilisation des fonctionnalités de gestion de la communication

La génération des clés de communication dans des groupes de structure hiérarchique de base « RenewKeyHierarchical » et dans des groupes fusionnés « RenewMergedGroupKey» sont les deux fonctions disponibles pour renouveler les clés de communication avec leurs éléments secrets. La première fonction sert à renouveler une clé dans un groupe de base qui n'a jamais été fusionné. La seconde fonction s'utilise dans le cas où le groupe qui renouvelle cette clé est le résultat d'une ou de plusieurs fusions. Ces fonctionnalités peuvent être appelées, soit par les fonctionnalités de renouvellement périodique, soit par d'autres fonctionnalités de changement dans les groupes.

La fonction de renouvellement dans un groupe de base utilise le fragment « DistributeKeyHierarchical» pour effectuer la distribution de la (les) clé (s) à tous les membres dans un groupe hiérarchique. La fonction pour effectuer cette distribution dans un groupe fusionné utilise d'abord, « DistributeKeyPlane» pour distribuer la (les) clé (s) aux chefs des groupes qui fusionnent et ensuite, « DistributeKeyHierarchical» pour faire cette distribution dans tous les groupes impliqués, en prenant en compte l'existence des classes dans les groupes (fig. 3.8).

La fonction « RenewBasePeriodicKey» sert à renouveler périodiquement les clés de session. Ce renouvellement est effectué toutes les « N » heures. Le chiffre « N » annonce la plage temporelle que le protocole respecte avant de renouveler la clé de session. La fonction « RenewPeriodicMergedKey» est en charge aussi du renouvellement périodique de la clé de session mais dans un groupe fusionné donc, dans tous les groupes qui font partie du groupe fusionné.

Les deux fragments mentionnés dans le paragraphe précédent héritent du fragment générique « DistributeKey », qui fait abstraction de la structure des groupes des utilisateurs et qui effectue une distribution physique de la (les) clé (s) de session vers l'ensemble des membres indiqués.

3.6. Fonctionnalités pour la transmission sécurisée

Des fonctionnalités de transmission sécurisée des messages seront nécessaires. Dans notre cas, nous incluons les opérations de transmission élémentaires et les opérations de sécurisation de ces messages dans la couche transmission. L'objectif est d'étudier en détail les temps d'exécution de ces deux types d'opérations : les opérations élémentaires de protection et de vérification des messages à échanger et les opérations de transmission et de réception de ces messages.

3.6.1. Transmission de messages

Les opérations effectuées pour la transmission des messages dans un protocole de groupes peuvent varier selon le réseau physique de déploiement de l'application.

Une première forme de réseau peut être Internet, où le type de transmission de messages peut être considéré comme *multicast*. Un autre type de médium peut être un réseau sans fil (wireless ou radio), où le type de transmission de messages peut être considéré comme *broadcast*. Un troisième type peut être les réseaux *peer-to-peer,* où les envois point-à-point sont le type de transmission de messages.

Pendant le déroulement d'une communication entre des groupes, les trois types de communication peuvent être requis. Pour des raisons physiques, il n'est pas toujours possible que le réseau fournisse les trois types d'émission. Par exemple, dans un réseau radio tous les envois des messages sont faits au travers des émissions *broadcast.* Cependant, pour des questions de généralité, nous conservons les trois types d'envois.

Pour prendre en compte toutes ces variétés de transmissions, nous proposons de faire une abstraction du médium qui fournisse ces trois types de transmission de messages : 1) Point-à-point (*P2P*), 2) Multipoint (*Multicast*) et 3) Diffusion (*Broadcast*).

3.6.2. Protection des échanges

« Opérations élémentaires de sécurisation des échanges » est le nom que nous associons aux tâches de protection des échanges de messages.

L'utilisation des opérations élémentaires de sécurisation peut varier en fonction des besoins de l'application. Par exemple, pendant la transmission d'un concert dans un réseau public, l'émetteur ne doit pas permettre la réception du concert par n'importe qui. Seules les personnes qui ont payé les droits de réception peuvent recevoir. Donc, une autorisation pour les personnes réceptrices doit être mise en œuvre.

Les performances du protocole sont aussi importantes vis-à-vis de la spécification des mécanismes élémentaires à utiliser dans le système. Il faudra garantir les propriétés en fonction de la sécurité et de la performance que l'on cherche à offrir à l'application.

Notre travail prend donc en compte un ensemble générique d'opérations élémentaires dans le but de fournir, pendant la modélisation et la vérification des protocoles de groupes, diverses possibilités pour la sécurisation des échanges. L'énoncé des propriétés que l'on cherche à garantir pendant la communication et pendant la mise en œuvre des mécanismes ou opérations élémentaires utilisés, est fait dans le tableau 3.4 ci dessous :

Propriété	Mécanisme. Opération Elémentaire
Confidentialité	Chiffrement Symétrique (groupe) Chiffrement Asymétrique (point-à-point)
Intégrité	Hashing
Authentification de la source	Certificat d'Identité (CI)
Non répudiation	Certificat d'Identité (CI) + Signature (Chiffrement Asymétrique)
Non Rejeu	Numéros de séquence

Tableau 3.4. Opérations élémentaires de sécurité pour des échanges de groupes

La première propriété dans le tableau 3.4, la confidentialité, cherche à maintenir l'information échangée à l'intérieur d'un groupe secrète pour des membres extérieurs, en utilisant le chiffrement symétrique. Si l'échange des messages est fait entre deux utilisateurs du groupe, le chiffrement asymétrique sera utilisé.

L'intégrité vise à ne pas permettre que l'information échangée soit malicieusement modifiée. L'authentification de la source permet aux membres récepteurs d'un message de connaître son émetteur. La Non Répudiation empêche l'émetteur de ce message de nier son origine.

Le Non Rejeu des messages est une propriété qui empêche une entité malicieuse de rejouer des messages qui sont été émis antérieurement dans le réseau. Ceci a pour intérêt de lutter contre les attaques qui utilisent des techniques de rejeu des messages dans le système.

Finalement, avec l'utilisation des certificats d'identité, le système attribue aux utilisateurs une identité, nécessaire pour assurer certaines opérations du tableau 3.4. [LIV 2.3]

3.7. Récapitulatif des fonctionnalités

Le comportement d'une session donne une idée générale des besoins de l'application et des fonctionnalités sur lesquelles elle repose. Ce comportement montre aussi l'interaction des fonctionnalités qui seront mises en œuvre lors du déroulement d'une session et donc lors de l'exécution de l'application. Nous montrerons les modules du système et les interactions que nous proposons d'étudier à l'aide du diagramme global d'interaction (IOD) de la fig. 3.9.

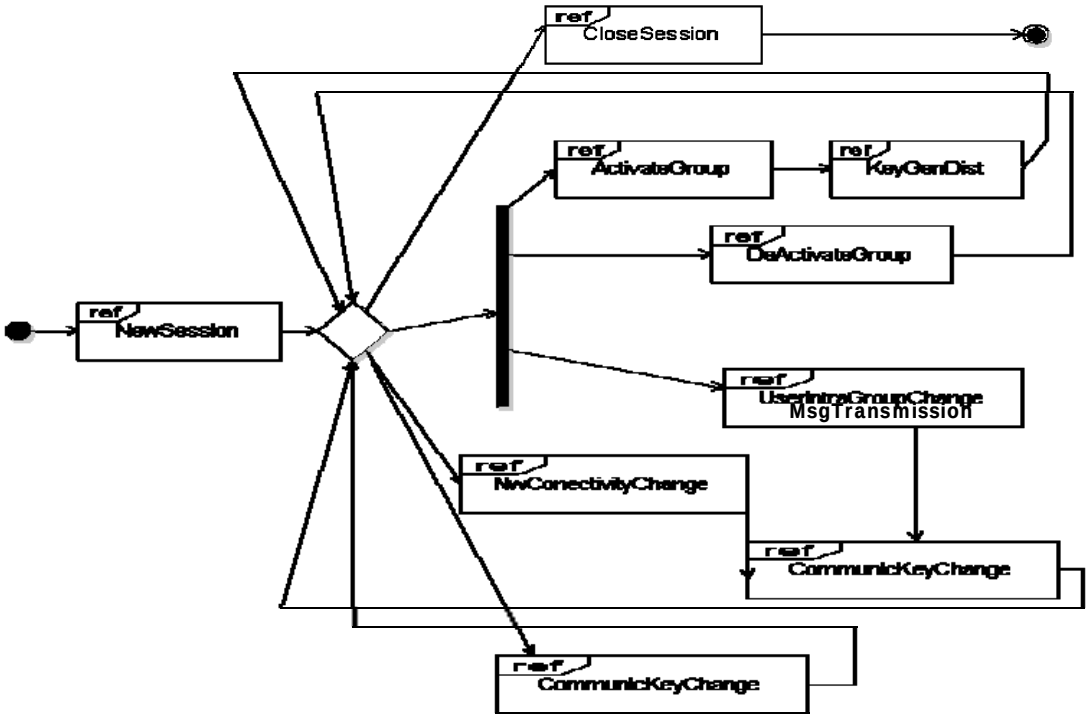


Fig. 3.9. IOD d'un système de communication de groupes sécurisés

La première fonctionnalité à instancier est : *NewSession.* Elle fournit la structure initiale des groupes sur laquelle le système va s'exécuter.

Les fonctionnalités *ActivateGroup, DeActivateGroup* et *UserIntraGroupChange* peuvent s'exécuter en même temps de façon concurrente pour des groupes différents. La première fonctionnalité sert à activer un groupe nouveau dans la session, la deuxième inactive un groupe déjà existant, s'il a fini sa communication. Les fonctionnalités inclues dans *UserIntraGroupChange* représentent les fonctionnalités de la section 3.4.1 : Gestion des Groupes.

NewConnectivityChange représente les fonctionnalités de la section 3.4.2 : Gestion de la Connectivité entre des Groupes, pour fusionner ou pour séparer des groupes en communication dans la session.

En ce qui concerne la gestion de la clé de session (3.5.3), nous pouvons observer dans la fig. 3.9 qu'elle s'actualise en étant la conséquence d'une modification dans la composition des groupes. Cette clé s'actualise aussi pendant l'activation d'un nouveau groupe, où une nouvelle clé de session est créée avec la fonctionnalité *KeyGenDist.* Cette fonctionnalité produit et distribue la nouvelle clé au sein du groupe qui s'active. La fonctionnalité de modification de la clé de session, *CommunicKeyChange,* doit s'effectuer après un changement dans la connectivité des groupes (*NwConnectivityChange)* ou bien après certains changements dans la composition interne d'un groupe *(UserIntraGroupChange*). *CommunicKeyChange* s'effectue aussi indépendamment des autres fonctionnalités, au moment d'actualiser la clé de session périodiquement.

La fonctionnalité *CloseSession* sert à fermer une session qui n'a plus d'intérêt. Toutes les fonctionnalités précédentes sont désactivées. Ceci représente un cas de fin de communication.

3.8. Conclusion

La définition des fonctionnalités à prendre en compte dans des protocoles de groupes sécurisés est une tâche assez difficile à réaliser. Nous nous sommes appuyés sur l'analyse des besoins des applications pour lesquelles le protocole doit être proposé. Le travail fait dans ce chapitre, montre et décrit les fonctionnalités que nous avons isolées.

Pour la définition des fonctionnalités à considérer, nous sommes partis de l'étude de groupes complexes et dynamiques, avec une structure qui présente des niveaux hiérarchiques. Nous avons isolé plusieurs classes de fonctionnalités qui s'appliquent sur ces groupes, qui offrent plusieurs facettes pour la modélisation, l'analyse et la vérification des propriétés du système : Il s'agit de la gestion des groupes, de la gestion de la communication sécurisée des groupes et de la transmission sécurisée des messages.

En ce qui concerne la sécurité, nous proposons aussi de permettre l'utilisation d'une gamme variée de mécanismes de sécurisation des échanges, dans un but de généralité et afin de considérer différentes options lors des étapes postérieures de modélisation, de vérification des performances, et de satisfaction des contraintes temporelles.

Pour des raisons de généralité, nous avons considéré différents types de transmission pour des médiums de communication variés. Ceci offre au concepteur des modèles toutes les possibilités de communication que l'on peut avoir entre des membres de groupes.

Traiter ces problématiques nous permettra de faire des analyses de protocoles proposés pour des cas d'utilisation différents et déployés au-dessus de réseaux physiques avec des caractéristiques différentes.

4 Architecture pour des communications de groupes sécurisés

4.1. Introduction

A notre connaissance, il n'existe pas de cadre général d'architecture communément admis pour modéliser des systèmes de sécurité [AMI 05]. Nous avons donc commencé par proposer une architecture générique qui peut être utilisée pour des protocoles de communication de groupes sécurisés. La définition de cette architecture s'appuie sur les fonctionnalités précédemment identifiées dans ce type de communications.

Dans la lignée du modèle de référence OSI (Open System Interconnection) [TAN 96] pour concevoir les services et les protocoles de communications des systèmes distribués, nous avons choisi une architecture en couches. Une telle architecture est décrite sous la forme d'un empilement de modules ou couches de protocole, chaque couche définissant un niveau d'abstraction.

Un service de niveau N, représente l'abstraction de la couche N et de l'ensemble des couches inférieures sur lesquelles la couche N s'appuie. Un service N est défini par un ensemble de primitives de service. Ces primitives servent pour le dialogue entre les utilisateurs du service N et ce service N. Un service N n'est pas un bloc monolithique, mais est composé d'un ensemble d'entités de niveau N. Ces entités communiquent et collaborent entre elles, et leur association fournit le service N. L'ensemble des messages échangés entre les entités N est appelé Unités de Données de Protocole (PDUs). Les règles de dialogue entre les entités N, associées aux PDUs, forment le protocole N qui réalise le service N. Pour communiquer entre elles, les entités N doivent utiliser un service de communication sous-jacent de niveau N-1 ou N-i accessible au moyen d'un ensemble de primitives de service de niveau N-1 ou N-i.

Ce type d'architecture est en fait « récursif ». En effet, les utilisateurs du service N peuvent former un ensemble d'entités de niveau N+1, et ainsi fournir un service de niveau N+1. De même, le service N-1 peut être formé par des entités de niveau N-1, qui communiquent en utilisant un service de niveau N-2.

La fig. 4.1 donne le schéma général d'organisation des primitives et des PDUs utilisés dans nos descriptions.

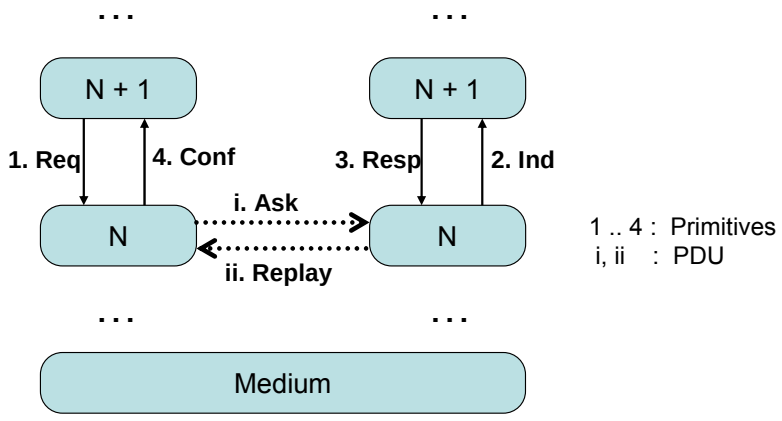


Fig. 4.1. Primitives et PDUs du modèle OSI

Afin de mieux positionner l'architecture présentée par la suite, la partie gauche de la fig. 4.2, appelée SGCva (Secure Group Communication verification architecture), montre les niveaux qui composent notre architecture. La partie droite montre la correspondance avec le modèle OSI [ZIM 80].

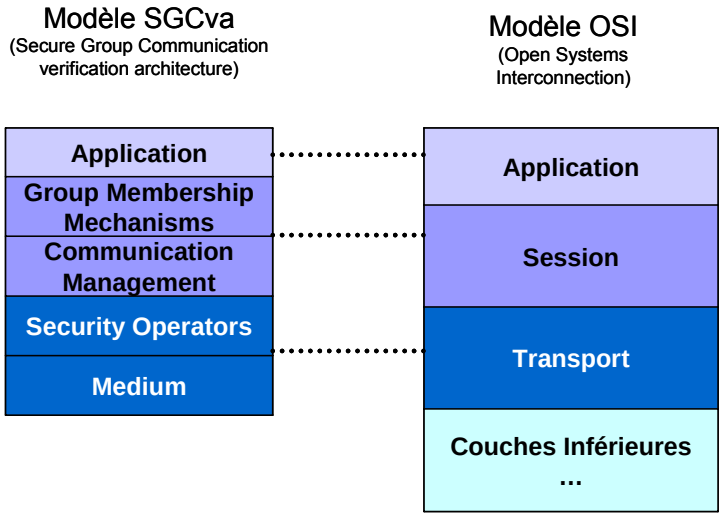


Fig. 4.2. Positionnement de l'architecture proposée par rapport au modèle OSI

Les niveaux que nous prenons en compte pour concevoir un système manipulant des groupes sécurisés s'inscrivent dans les couches hautes du modèle OSI. De façon plus précise, le travail proposé dans ce mémoire se positionne dans les couches Application, Session et Transport du modèle OSI. Le niveau le plus bas de notre architecture, situé au niveau transport, correspond à des services de communication ; le niveau intermédiaire, situé au niveau session, correspond à des services de gestion, utilisés par l'application finale au niveau le plus haut.

4.2. Architecture proposée

L'architecture proposée supporte des groupes de n utilisateurs. Pour chaque utilisateur, cinq modules doivent être installés. Ces modules correspondent aux services de base pour des communications de groupes sécurisés. Les flèches dans l'architecture montrent les liens entre les modules. Chacune des couches de notre architecture sera détaillée dans la suite de cette section.

La fig. 4.3 montre l'*Architecture pour la Vérification des Protocoles de Communication des Groupes* Sécurisés, appelée SGCva (Secure Group Communication Vérification Architecture), que nous proposons.

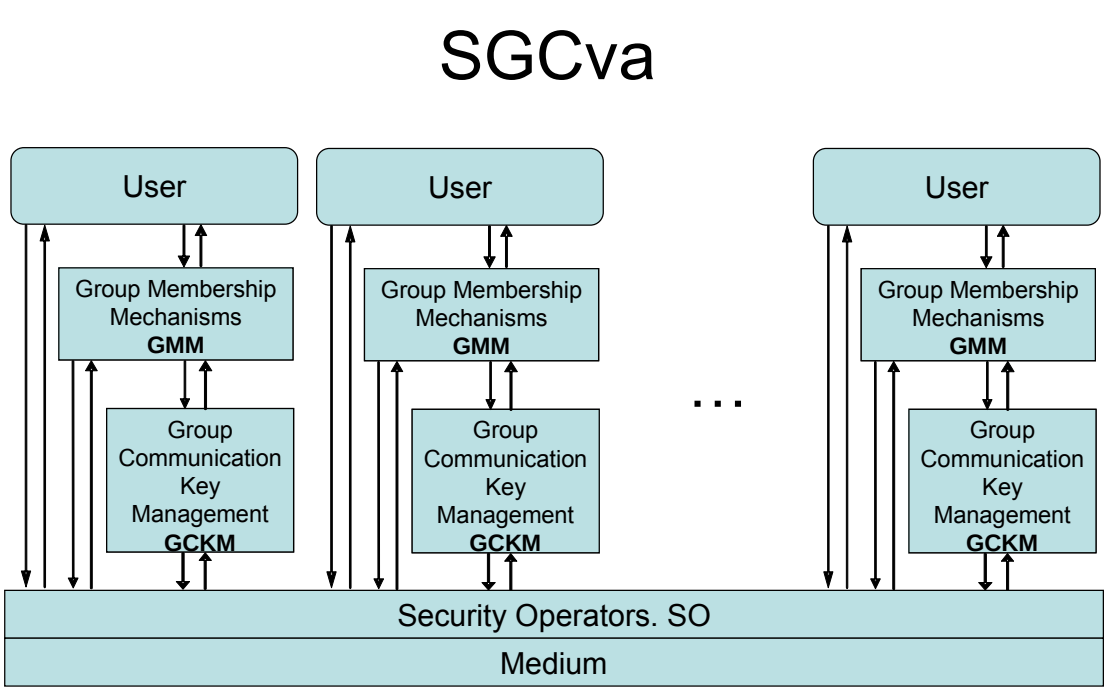


Fig. 4.3. Architecture pour la vérification des protocoles de communication de groupes sécurisés

Cinq niveaux développés sous la forme de quatre couches indépendantes composent l'architecture de modélisation que nous proposons (fig. 4.3).

Le premier niveau, le plus bas, *Medium,* offre les services élémentaires de diffusion fiable dans des médiums multipoint. Il sera expliqué dans la sous-section 4.3.2.

Le deuxième niveau, *Security Operators* (SO), offre les actions élémentaires de sécurité lors des échanges. On trouve des opérations pour garantir l'intégrité des données, des opérations qui servent à garantir la confidentialité dans la communication et des opérations de gestion d'index pour empêcher le rejeu.

Les niveaux SO et medium, tous deux de bas niveau, ont été rassemblées dans une couche générique de transport sécurisé que nous avons appelée couche « Medium/SO ». Nous avons introduit des liens directs depuis les couches supérieures de notre architecture vers cette couche *Medium/SO.* En effet, en accord avec le schéma de la fig. 4.1, ces liens directs permettent aux entités des couches supérieures d'utiliser directement les services de la couche la plus basse, sans passer par des couches intermédiaires qui ne serviraient que de relais, sans aucun autre traitement.

La couche *Group Communication Key Management (GCKM)* gère les opérations pour la sécurisation de la communication des groupes, opérations liées aux clés de session. Le service de renouvellement des clés inclut deux sous-services : la génération et la distribution des clés. Les clés de session générées sont destinées à chaque niveau de hiérarchie d'un groupe de communication. La distribution des clés se fait selon deux cas : entre des chefs d'un groupe et ses groupes correspondants ou bien à l'intérieur d'un groupe. Le renouvellement des clés doit se faire en maintenant l'intégrité du système vis-à-vis des changements des rôles et des entrée/sorties des utilisateurs.

La couche suivante, *Group Membership Mechanisms (GMM),* se charge des éléments de gestion de groupes en contrôlant leur structure, leur évolution et

leur dynamique. En premier lieu, cette couche intègre les fonctionnalités intra groupes identifiées dans le chapitre précédent, et intégrées sous la forme d'un service par fonctionnalité. Le fonctionnement de ces services se fait en relation avec les mécanismes de gestion des clés de session sous-jacents et l'utilisation des rôles qui donnent des droits aux membres. En second lieu, cette couche intègre aussi les fonctionnalités inter groupe, identifiées dans le chapitre précédent, et intégrées sous la forme d'un service par fonctionnalité. Avec ces services nous effectuons des opérations entre K groupes, c'est-à-dire des fusions et des divisions entre groupes.

La dernière couche contient une abstraction des comportements et des possibilités des utilisateurs. Cette abstraction que nous proposons nous permettra de représenter les différents aspects applicatifs importants lors de l'étude des protocoles de groupes, afin de faciliter la vérification et l'interprétation des résultats. Elle sera détaillée dans le chapitre cinq.

4.3. Couche Transport sécurisée

La couche *Transport Sécurisé* se charge à la fois des opérations de transmission et des opérations de protection des messages. Des garanties temporelles dans les services de communication ainsi que des garanties en termes de sécurité des échanges font partie de notre étude.

Au niveau le plus bas, nous avons effectué une abstraction des caractéristiques du réseau physique au-dessus duquel les protocoles de communication des groupes seront déployés, sous la forme d'une sous–couche de communication appelée Médium. Sa modélisation intègre tous les éléments composant un médium générique. Ce médium peut être particularisé afin de prendre en compte certaines spécificités lors de la vérification.

Au dessus, la réalisation des opérations de protection des échanges se fait dans la sous-couche que nous avons appelée *Opérations Elémentaires de Sécurisation des Echanges (Security Operators SO)*.

Dans notre architecture, le grand avantage que nous avons trouvé dans cette décomposition en deux sous-couches est la modularité introduite. Ainsi, nous pouvons particulariser et modifier le médium à adopter et les paramètres de sécurité des échanges de façon indépendante entre eux lors de la modélisation et de la vérification du protocole, suivant les besoins de l'application qui utilisera cette architecture.

La description des primitives de service que nous proposons respecte les conventions de l'OSI montrées dans la fig. 4.1. Les requêtes (Req) et les réponses (Resp) vont de la couche supérieure vers la couche inférieure. Les indications (Ind) et les confirmations (Conf) vont de la couche inférieure vers la couche supérieure.

4.3.1. Primitives de service de la couche transport sécurisée

L'architecture a été construite de manière à simplifier l'étude des mécanismes qu'elle utilise. Dans la couche médium, seulement deux primitives sont nécessaires pour fournir tous les services de transmission des messages sécurisés. *« send »* et *« receive »* permettent, respectivement, d'envoyer et de recevoir des messages.

4.3.1.1 Primitives de la couche transport sécurisée

- *send* (*_receiver Address, _sender Address, _control Information + Message, _certificates, _security Operations for Building, _security Operations for Reading*)
- *receive* (*_receiver Address, _sender Address, _control Information + Message, _certificates, _security Operations for Building, _security Operations for Reading*)

Avec la primitive *send* nous pouvons fournir les trois types de services de transmission suivants : 1. Point-à-point (de 1 vers 1), 2. Multicast (de 1 vers n), 3. Broadcast (de 1 vers tous). Pour permettre cela, nous nous appuyons sur l'organisation des membres en groupes structurés. Nous traduisons cette structure en adresses. Les destinataires d'un message sont désignés par le paramètre *_sender Address* qui se compose des trois parties suivantes : *_group*, *_class*, *_member*. Les valeurs que ces paramètres peuvent prendre sont : soit le numéro correspondant au groupe, à la classe ou au membre, soit un *« 0 »* pour indiquer *« vers tous »*.

Avec une seule primitive *receive*, un membre en communication peut recevoir un message qui a été envoyé, soit seulement à lui, soit à plusieurs utilisateurs dont il fait partie, soit à tous les membres du réseau, quitte à vérifier ensuite que le message reçu est légitime vis-à-vis des destinataires et de son émetteur. Le récepteur peut connaître l'origine du message ainsi que ses destinataires.

Chacune des couches supérieures GCKM, GMM et Application, au travers des connexions directes de l'architecture, se sert des primitives *send* et *receive.*

4.3.1.2 Paramètres des primitives de la couche transport sécurisée

Receiver Address : Un message peut-être dirigé vers un, vers plusieurs utilisateurs (multicast) ou vers tous les utilisateurs en communication (broadcast).

Sender Address : L'identification de la source est nécessaire pour avoir la capacité de répondre à un message reçu.

Control Information : Les informations comme le type de message, le type de PDU et le numéro de séquence, en tant que mécanismes de protocole de niveau supérieur, sont considérés comme information de contrôle.

Message : normalement un PDU de niveau supérieur est envoyé dans cette partie.

Certificates : Ces informations correspondent aux certificats. Elles comprennent le type du certificat ainsi que sa taille.

Security Operations for Building : Les opérations de sécurité qui devront s'appliquer au message avant de l'envoyer sont incluses dans cette partie. Ces opérations comprennent des mécanismes pour la confidentialité, l'intégrité, l'authentification et la non répudiation.

Security Operations for Reading : Les opérations de sécurité qui devront s'appliquer au message après sa réception sont incluses dans cette partie. Ces opérations permettent notamment la vérification de l'intégrité des messages reçus, à partir des opérations faites lors de la construction du message.

4.3.2. Sous-couche Médium

Il est difficile de parler dans cette sous-section d'un médium spécifique car une application de communication de groupes peut se déployer dans des réseaux qui ont des caractéristiques physiques très différentes. Le réseau Internet est un exemple de médium sur lequel nous pouvons rencontrer plus d'un groupe avec des membres en communication pour une même application, par exemple un jeu. Les caractéristiques physiques du réseau peuvent être très différentes d'un groupe à un autre ou même d'un utilisateur à un autre dans un même groupe.

C'est pour cela que, afin de rester assez général et afin de considérer un médium universel, nous faisons une abstraction du réseau physique, et nous nous focalisons sur les services que nous attendons de cette sous-couche.

Nous définissons les hypothèses suivantes : dans une communication de groupes, nous pouvons trouver plus d'un groupe ; chaque groupe peut avoir plus d'une classe et chaque classe peut être composée de plus d'un membre. Nous donnons les exemples de la fig. 4.4. qui présentent différents schémas de transmission pour des communications de groupes.

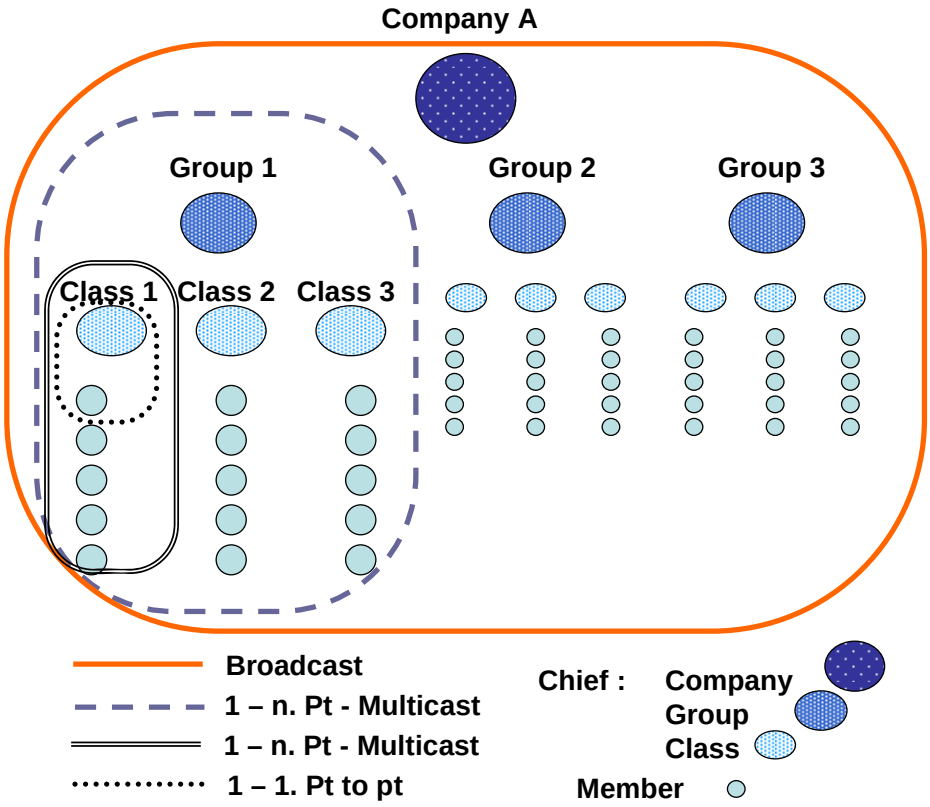


Fig. 4.4. Services de transmission de messages de l'architecture SGCva

Si (_group == 0), alors le message sera envoyé vers tous les groupes de la compagnie. Le spectre de réception est matérialisé par la ligne continue de la fig. 4.4.

Si (_group == 1) et (_class == 0), alors le message sera envoyé vers toutes les classes contenues dans le groupe 1. Le spectre de réception est matérialisé par la ligne hachurée de la fig. 4.4.

Si (_group == 1) et (_class == 1) et (_member == 0), alors le message sera envoyé vers toutes les membres de la classe 1 du groupe 1. Les membres qui recoivent font partie de l'ensemble matérialisé par la ligne double de la fig. 4.4.

Si (_group == 1) et (_class == 1) et (_member == 1), alors le message sera

envoyé vers le membre 1 de la classe 1 du groupe 1. Ce membre est entouré par une ligne pointillée dans la fig. 4.4.

La ligne pointillée de la fig. 4.4 montre une communication point-à-point. Les lignes hachurées et doubles montrent des diffusions de *1 vers n (multicast)* et la ligne continue montre une communication de type *broadcast*. Le concept de distribution des messages reste, avec ces modes de transmission, beaucoup plus flexible, dynamique et générique.

Les services rendus par cette couche Médium s'abstraient de la qualité du réseau physique, car les études que nous effectuons ne concernent pas les problèmes de fiabilité ou de routage. En première approche, nous supposons que le médium au-dessus duquel nous bâtissons notre protocole est sans perte. Cependant, l'architecture proposée nous permet de représenter les caractéristiques physiques qui nous intéressent afin de vérifier les contraintes temporelles sur l'ensemble du système. Ainsi, nous considérons la distance et le débit pour les communications. La taille des messages intervient aussi pour décrire les messages. Les autres paramètres sont ignorés. Pour la partie de sécurité des échanges (SO) nous prenons en compte les temps d'exécution des services de sécurité de base.

4.3.3. Sous-couche Operations élémentaires de sécurisation des échanges (Security Opérations SO)

La sous-couche *Opérations Elémentaires de sécurisation des échanges* contient les algorithmes et les techniques nécessaires pour protéger les messages contre des actions malicieuses extérieures. Nous garantissons la confidentialité et l'intégrité des messages en utilisant des techniques de chiffrement et de hashing. Les propriétés de non-répudiation sont garanties au moyen de l'utilisation de certificats et signatures. La protection contre le non rejeu des messages fait aussi partie des opérations élémentaires de cette sous-couche. Nous utilisons des numéros de séquence.

Chaque application possède ses propres besoins et ne requiert pas forcément une garantie de toutes les propriétés mises en œuvre dans cette sous-couche. Nous pouvons ainsi utiliser de façon modulaire et indépendante les fonctionnalités que cette sous-couche implante de façon complète.

Les sous-sections suivantes décrivent les mécanismes mentionnés pour garantir les propriétés de sécurité.

4.3.3.1 Confidentialité des messages

Nous considérons dans cette couche, deux types de chiffrement des messages : le chiffrement asymétrique (systèmes à clé publique) et le chiffrement symétrique (systèmes à clé secrète). Nous appliquons dans notre architecture les deux mécanismes de protection des messages car leur utilisation est complémentaire dans les protocoles de groupes sécurisés complexes.

Dans un système à clé publique, chaque utilisateur possède une paire de clés, une clé publique *k* (qui peut être vue par tous les membres en communication) et une clé secrète/privée k^{-1}(qui est connue seulement par son propriétaire). Si un utilisateur, disons *« Alice »*, veut envoyer le message *messageAlice* à un autre utilisateur, disons *« Bob »,* il chiffre *messageAlice* à l'aide d'une fonction de

chiffrement *C* et de la clé publique de Bob *kbob,* pour ensuite envoyer le message. Le récepteur *« Bob »* déchiffre le message à l'aide de la fonction de déchiffrement *D* et de sa clé privée $kbob^{-1}$ pour obtenir le message original.

La technique asymétrique s'applique très bien pour des échanges de messages entre deux utilisateurs fixes. Par contre, lorsque l'on utilise cet algorithme pour des communications de groupes, il se pose le problème de devoir gérer une grande quantité de paires de clés pour tous les utilisateurs participants. Ceci nécessiterait un nombre trop important d'opérations de protection des messages, pour maintenir la confidentialité d'échanges vers un ensemble de membres destinataires. Cette solution est non viable pour des communications des groupes, car, de plus, elle ne supporte pas le passage à l'échelle.

Pour pallier cela, nous proposons d'utiliser des techniques de chiffrement symétrique. Une seule clé est générée et partagée par tous les membres dans un groupe. Cette clé secrète sera utilisée pour chiffrer et pour déchiffrer les messages échangés au sein du groupe. Dans un système à clé secrète (symétrique), un groupe d'utilisateurs détient une seule clé de communication *kDuGroupe*. Si un utilisateur, disons encore *« Alice »,* veut envoyer un message *messageAlice*, maintenant dirigé vers tous les membres du groupe, il chiffre *messageAlice* à l'aide d'une fonction de chiffrement *C* et de la clé symétrique du groupe *kDuGroupe,* pour ensuite envoyer le message. Tous les récepteurs déchiffrent le message à l'aide de la fonction *D* et de la même clé secrète *kDuGroupe* pour obtenir le message original.

En plus des deux solutions élémentaires de chiffrement précédemment retenues, nous avons implanté un mécanisme de séquences de clés qui sont assignées à chaque classe hiérarchique pour garantir la confidentialité entre ces classes à l'intérieur d'un groupe. Les clés seront produites comme expliqué dans la section 3.5.2.

En conclusion, pour garantir la confidentialité des échanges dans des groupes hiérarchiques, les trois mécanismes seront retenus. Pour la transmission point-à-point, des algorithmes de chiffrement asymétriques seront utilisés. Pour la diffusion multicast et *broadcast*, le chiffrement symétrique sera adopté et, finalement, pour garantir des propriétés de confidentialité entre les classes d'un groupe, des clés symétriques hiérarchiques seront considérées.

Un autre niveau de confidentialité sera aussi considéré, la confidentialité dans un groupe modifié après une ou plusieurs opérations de fusions/séparations.

Les propriétés que nous prenons en compte lors de la sécurisation des messages échangés sont énumérées et expliquées par la suite.

4.3.3.2 Intégrité des messages.

Pour garantir l'intégrité des messages, notre architecture permet d'appliquer des fonctions de hachage au message original. Une fois appliquée sur le message à envoyer, cette fonction produit un résultat unique qui est accolé au message transmis. Dès que le message arrive à son destinataire, un nouveau calcul de hachage est fait sur ce message. Si le message est malicieusement changé pendant sa transmission, alors nous trouvons une différence entre les deux valeurs transmises et calculées.

Nous ajoutons dans notre modèle les temps de réalisation et d'application des fonctions de hachage retenues.

4.3.3.3 Authentification des entités

Un service essentiel pour des communications de groupe est la capacité d'identifier et de vérifier l'origine d'un message arrivant à un utilisateur dans le réseau. L'utilisation de certificats est une manière sûre de procéder à l'authentification de la source, et de connaître l'identité de l'émetteur d'un message.

Des certificats d'identité seront ajoutés aux messages pour que le(s) récepteur(s) connaissent l'identité de la personne qui leur envoie un message et pour prendre les mesures correspondantes. Ainsi, tout membre non autorisé qui envoie un message dans le réseau sera détecté par n'importe quel membre du groupe.

Nous ajoutons dans notre modèle le temps de lecture et d'interprétation des certificats d'identité.

4.3.3.4 Non répudiation de la source

Lors de communications de groupes, il est important de vérifier que tout émetteur d'un message ne puisse nier cette émission. Ainsi, pour lutter contre la répudiation des messages, nous signons tout message émis avec la clé privée de l'émetteur. En effet, avec la signature et le certificat d'identité dans le message émis, le récepteur identifiera de façon formelle l'émetteur du message.

Nous ajoutons dans notre modèle les temps qui sont nécessaires pour faire signer un message par la source et pour le faire vérifier par le récepteur.

4.3.3.5 Non Rejeu des messages

Finalement, le non rejeu est le dernier mécanisme introduit dans cette couche. Tout message qui circule dans le réseau aura un champ qui s'appelle *Num_de_Sequence*. Ce champ est un numéro de séquence, incrémenté par tout émetteur à chaque nouvelle émission. Ce mécanisme évite que tout message ne puisse être réémis. Le système vérifie les numéros de séquence, et identifie tout numéro identique comme une réémission.

Nous ajoutons dans notre modèle les temps qui sont nécessaires pour générer et pour vérifier les numéros de séquence des messages.

4.4. Couche Session

Les services qui interviennent au niveau de la gestion soit des membres soit de la sécurisation des groupes ont été positionnés dans cette couche car, à notre avis, ils font partie de fonctionnalités intermédiaires. Cette couche se compose des deux sous-couches GCKM et GMM. La sous-couche GCKM se charge de l'activité la plus importante des protocoles de groupes sécurisés : la gestion de la clé de session. La sous-couche GMM se charge de la composition et de la dynamique des groupes.

Les deux sous-couches sont connectées à la couche de transport sécurisé pour communiquer avec les entités homologues installées sur les autres machines. Une connexion directe entre la couche de gestion de groupes et les services de gestion de clés est nécessaire pour accéder aux services concernant la clé de session à partir des services de gestion de groupes.

4.4.1. Sous-couche de Gestion de la Communication des Groupes (Group Communication Key Management GCKM).

Cette sous-couche gère les opérations liées à la sécurisation de la communication entre des groupes, opérations liées aux clés de session.

Le service de renouvellement de clés est le service cœur de cette sous couche. Il inclut le service de génération de la clé ainsi que le service de distribution des clés. Les clés de session générées sont destinées à chaque groupe de communication. Ce groupe peut être un groupe de base ou bien un groupe résultant des fusions et/ou séparations. La distribution des clés se fait donc selon deux cas : à l'intérieur d'un groupe de base ou dans un groupe fusionné.

Le renouvellement des clés se fait en gardant ainsi l'intégrité du système vis-à-vis de la dynamique des utilisateurs dans les groupes et de la connectivité des groupes dans la session.

Les services fournis pour la gestion de la clé de session sont assurés par les primitives suivantes :

4.4.1.1 Primitives de la couche GCKM

Les primitives de la couche GCKM sont en charge de fournir l'accès aux services de gestion de clés de session. Les primitives répondent aux échanges nécessaires entre la couche de gestion de groupes et le module de gestion de la clé de session pour actualiser en temps réel la clé de session d'un ou de plusieurs utilisateurs dont le statut a changé. Les paramètres des primitives de cette sous-couche contiennent seulement de l'information concernant les membres qui participent. L'acheminement des messages est fait par la couche Transport Sécurisé.

Afin de simplifier leur compréhension, nous classons les primitives de la couche session en fonction des types de groupes et des entités qui les utilisent.

A). Primitives pour des groupes de base

Les primitives qui fournissent l'accès aux services correspondants à la clé de session en réponse à des mouvements des membres à l'intérieur d'un groupe sont décrites dans ce qui suit. Ces primitives sont de type requête *Req*, dont l'appel provient de l'extérieur.

Chief Administrator GMM → Chief Administrator GCKM

- *keyReq* (*_control Information*)
- *renewKeyExclusionReq* (*_control Information*)
- *newSessionKeyRenewReq* (*_control Information*)

Le *Chief Administrator* dans sa couche GMM est le seul qui peut demander à sa couche GCKM de distribuer la clé de session de son groupe vers *un,* vers *n* ou vers *tous* les membres de son groupe. La primitive *keyReq* sert à demander d'envoyer la clé de session à l'utilisateur identifié par le champ *_control information. Suite à l'exclusion d'un membre,* la primitive *renewKeyExclusionReq* sert au *Chief Administrator* pour qu'il renouvelle la clé de session pour tous les membres d'un groupe.

Lors de la création d'une nouvelle session, la primitive *newSessionKeyRenewReq* sert à demander au *Chief Administrator* de la couche GCKM de générer et de distribuer la nouvelle clé de session vers tous les groupes participant à cette session.

Concerned Member GCKM → Concerned Member GMM
- *keyReceivedConf* (*_control Information*)
- *renKeyExclusionInd* (*_control Information*)

Le *Concerned Member,* dans sa couche GMM, doit être informé de tout changement de clés dans la couche GCKM. Les primitives *keyReceivedConf* et *renKeyExclusionInd* informent que la clé de session vient d'être changée. La première primitive correspond à une clé attendue et la deuxième à une clé changée à cause de l'exclusion d'un membre.

Member GCKM → Member GMM
- *newSessionKeyRenewInd* (*_control Information*)

newSessionKeyRenewInd permet à la couche GCKM d'informer la couche GMM d'un changement de clés qui vient de se passer à cause d'une création de session. La couche GMM du membre sait qu'il fait partie d'une nouvelle session.

B). Primitives pour des groupes fusionnés

Tout changement dans la connectivité des groupes doit être supporté par des services d'actualisation de la clé de communication fournis par les primitives de cette sous-section.

Chief Administrator GMM → Chief Administrator GCKM
- *renKeyMergeReq* (*_control Information*)
- *chngKeySplitReq* (*_control Information*)

Seuls les chefs des groupes sont autorisés à faire des changements inter groupes et donc à demander la génération de la clé de session correspondante. La primitive *renKeyMergeReq* sert à demander la génération de cette clé de session en réponse à une fusion (*merge)* effectuée. La primitive *chngKeySplitReq* sert pour que le *chiefAdministrator* avertisse la couche de gestion de clés qu'une fusion vient de s'effectuer et que le chiffrement des messages sera maintenant effectué avec l'ancienne clé du groupe.

Member GCKM → Member GMM
- *renKeyMergeInd* (*_control Information*)

La couche GCKM informe, sous la forme d'une indication (*Ind*), la couche GMM qu'un changement de clés vient de se passer suite à un *renKeyMergeReq.* Donc, elle est utilisée pour que la couche GMM sache que le membre qui a reçu une nouvelle clé fait partie d'un groupe fusionné, soit parce qu'il vient d'être créé, soit parce que la clé a été renouvelée après une certaine période.

Member GMM → Member GCKM
- *chngKeySplitReq (_control Information*)

La couche GMM informe son entité de la couche GCKM qu'un *split* vient de se passer. La clé de communication du groupe ancien doit être utilisée.

4.4.1.2 Description des services de la couche GCKM

Nous décrivons dans cette sous-section les services génériques qui couvrent les différents besoins en termes de sécurité de communication des groupes. Le protocole considéré dans ce mémoire propose deux types de renouvellement de clés. Le premier peut se passer au sein d'un groupe de base et le deuxième peut se passer au sein d'un group fusionné. Nous allons considérer dans cette sous-section les deux cas.

4.4.1.2.1. Renouvellement d'une clé de session dans un groupe de base.

Dans un groupe de base, le processus de renouvellement peut s'exécuter lorsqu'un membre intègre un nouveau groupe dans une session de communication, lorsqu'un membre est exclu, ou bien, pour renouveler périodiquement une clé de session.

Le processus de renouvellement d'une clé doit s'utiliser avec précaution car il est considéré comme un processus critique. Tout d'abord, il est critique pour la sécurité dans le sens où il distribue une clé de session vers tous les membres d'un groupe et que cette clé ne doit pas être interceptée lors de son actualisation. Ensuite, il est critique pour la performance car, lorsque le système a besoin de bande passante pour des communications, il ne faut pas que le protocole de renouvellement surcharge la bande passante avec de nombreux messages de contrôle, lors de trop fréquentes actualisations.

C'est la raison pour laquelle, une fois qu'une session est déjà ouverte, nous limitons le renouvellement. Nous ne lançons ce processus que lorsqu'il est absolument nécessaire, par exemple quand un membre est exclu du groupe. Dans le cas contraire, le renouvellement est fait périodiquement après une période *T* établie en fonction des besoins en termes de sécurité.

La fig. 4.5 montre le contenu d'un message de renouvellement de clés :

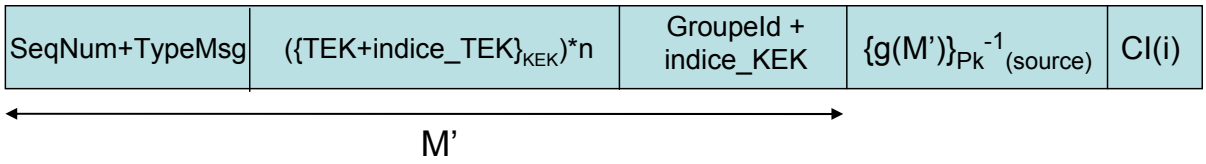


Fig. 4.5. Message de renouvellement de la clé de session TEK dans un groupe de base

Ce type de message correspond au format contenant la (les) clé(s) de session.

En effet, dans le renouvellement des clés, la ou les nouvelles clés de session sont calculées et envoyées. Dans ce message, *n* représente le nombre de niveaux hiérarchiques dans le groupe. Toutes les clés qui correspondent aux niveaux hiérarchiques sont envoyées avec les indices des clés TEKs. Les clés TEK et les indices sont chiffrés avec une KEK.

Comme les membres de chaque classe hiérarchique dans le groupe contiennent leur propre vecteur de clés KEK, à l'arrivée du message, chaque membre prend la clé KEK qui lui correspond et qu'il trouve à l'aide de la valeur indice_KEK envoyée dans le message. Chaque membre peut ainsi déchiffrer la partie correspondante à la clé TEK de sa classe dans le message car les clés de toutes les classes seront y incluses.

L'algorithme de renouvellement de la clé de session dans un groupe de base est le suivant :

- Système -> Chef : Demande *RenewKeyHierarchical*. Cette demande active le processus en charge de faire un renouvellement dans un groupe de base.
    - Le Chef renouvelle la clé de session et génère les clés des classes.
    - Le Chef fait *DistributeKeyHierarchical* pour effectuer la distribution des clés aux différentes classes dans son groupe.

4.4.1.2.2. Renouvellement d'une clé de session dans un groupe fusionné

Comme précédemment, nous limitons le plus possible le processus de renouvellement dans un groupe fusionné pour ne pas augmenter la possibilité d'interception des clés ni consommer trop de bande passante par des changements trop fréquents.

Un groupe fusionné est composé de plusieurs groupes de base. Parmi les groupes fusionnés, certains peuvent à leur tour provenir de fusions précédentes. Dans tous les cas, tous les groupes participant dans une fusion garderont leur clé d'origine, pour avoir la capacité de revenir à leur configuration précédente après une séparation. Donc, les clés d'un groupe fusionné s'ajoutent aux clés des groupes de base.

Le processus de renouvellement des clés dans un groupe fusionné est exécuté lors de l'opération de fusion des groupes ou bien lors du renouvellement périodique de la clé de session du groupe déjà fusionné.

Le renouvellement des clés dans un groupe fusionné s'appuie sur la structure des groupes d'origine. Il s'effectue dans deux étapes : la première étape réalise la génération des clés du groupe fusionné et la distribution de ces clés aux chefs des groupes participants à la fusion. La deuxième étape utilise la structure d'origine des groupes pour la distribution de la nouvelle clé (c.à.d. la clé du groupe fusionné) vers tous les membres. Dans cette deuxième étape, chaque chef d'un groupe de base, en utilisant ses clés de session d'origine, distribue la nouvelle clé aux membres de son groupe pour la future communication au sein du groupe fusionné.

Le fait de diviser la distribution des clés dans ces deux étapes marque la différence entre le renouvellement des clés dans un groupe de base et dans un groupe fusionné. Le message d'envoi de la nouvelle clé de session aux chefs des groupes participants à une fusion est montré dans la fig. 4.6. Cette étape ne se réalise pas dans un groupe de base. La deuxième étape correspond à l'envoi des clés à l'intérieur de chaque groupe à l'aide du message montré par la fig. 4.5. Cette étape correspond à une distribution de la clé dans un groupe de base mais elle est réalisée en parallèle plusieurs fois, pour chaque groupe composant la fusion.

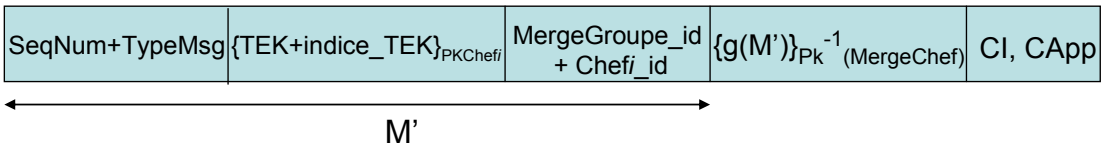


Fig. 4.6. Message de distribution de la clé de session TEK aux chefs d'un groupe fusionné

Dans la fig. 4.6, la clé de session TEK est envoyée dans un message direct au chef du groupe *i*. L'identification du nouveau groupe fusionné est aussi envoyée. Cet envoi est fait vers plusieurs chefs si plusieurs groupes participent à la fusion.

L'algorithme de renouvellement de la clé de session dans un groupe fusionné avec deux groupes de base est le suivant :

Hypothèse : Le groupe fusionné est composé des Groupes 1 et 2. Le leader du groupe fusionné est le Chef du groupe 1, Chef 1.

- Système -> Chef 1 : Appel de *RenewMergedGroupKey.* Cette demande active le processus en charge de faire un renouvellement dans un groupe fusionné.
  - Le Chef 1 renouvelle la clé de session
  - Chef 1 -> Chef 2 : La nouvelle clé de session est transmise.
- On réalise la distribution des nouvelles clés de session dans les groupes (distributions parallèles).
  - Chef 1 -> Groupe 1 : Transmission du message contenant les clés du groupe fusionné
  - Chef 2 -> Groupe 2 : Transmission du message contenant les clés du groupe fusionné

4.4.1.2.3. Renouvellement périodique de la clé de session dans des groupes de base

Le renouvellement périodique de la clé de session est fait pour garantir les propriétés de confidentialité pré-adhésion et post-résiliation. Cette confidentialité assure qu'un membre qui entre dans un groupe déjà en communication ne peut pas comprendre un message émis avant un temps T1 avant son arrivée (pré-adhésion). De la même façon, il ne peut pas comprendre un message émis après un temps T2 après sa sortie (post-résiliation). La relation entre les temps de pré-adhésion et de post-résiliation est la suivante : La somme des temps de pré-adhésion et post-résiliation est égale à la période T de renouvellement de la clé. Cette propriété garantit que les messages transmis avant la période ne seront pas compris par le membre. De la même façon, les messages échangés après la période ne seront pas compris puisque le membre sortant n'actualise pas la clé de session.

La période *T* de renouvellement de la clé peut être choisie en fonction des besoins de l'application. Un appel périodique au renouvellement de la clé de session pour un groupe de base est fait toutes les *T* unités de temps.

Toutes les clés correspondant aux classes hiérarchiques du groupe sont renouvelées avec la clé de session. Une distribution sécurisée actualise les clés des utilisateurs dans le groupe avec l'envoi du message montré dans la fig. 4.6.

4.4.1.2.4. Renouvellement périodique de la clé de session dans des groupes fusionnés

Le principe de fonctionnement de ce renouvellement est le même que pour un groupe de base, ceci afin de garantir la confidentialité pré-adhésion et post-résiliation. En utilisant la période *T,* le renouvellement est effectué périodiquement. Et comme dans la section 4.4.1.2.2, ce processus se fait en deux étapes. Tout d'abord, une distribution de la clé de session se déroule entre les chefs. Ensuite, une distribution se produit vers les membres de chaque groupe. La même procédure est effectuée en parallèle dans tous les groupes en communication.

4.4.1.3 Messages de protocole de la sous couche GCKM (PDU)

Les PDUs échangés entre deux entités de la couche GCKM du protocole sont listés dans cette sous-section. Il s'agit des messages échangés pour gérer la clé de communication dans une session.

Les PDUs présentés dans cette sous-section ont comme paramètres, l'identification de l'émetteur, du destinataire et la/les clé(s) de session à transmettre.

A. PDUs pour des groupes de base

ChiefAdministrator → Members

- *keyRenNewSession* (*_keySecrets*)

Ce PDU sert à envoyer la (les) clé (s) d'une nouvelle session aux membres (couche GCKM) dans un groupe qui commence sa communication. C'est le *ChiefAdministrator* qui envoie aux membres de son groupe les PDUs contenant la clé de session.

- *keyRenExclusion* (_addressSource, _addressDest, *_keySecrets*)

Ce PDU correspond à une exclusion d'un membre faite par le *ChiefAdministrator.* Il envoie aux membres de son groupe un PDU contenant l'identification du membre qui se fait exclure et la (les) nouvelle (s) clé (s) de session qui s'actualise chez tous les membres.

ChiefAdministrator → ConcernedMember

- *keyJoinRep* (_addressSource, _addressDest, *_keySecrets*)
- *keyLeaveRep* (_addressSource, _addressDest, *_keySecrets*)
- *keyUpgradeRep* (_addressSource, _addressDest, *_keySecrets*)
- *keyDowngradeRep* (_addressSource, _addressDest, *_keySecrets*)
- *keyReconnectionRep* (_addressSource, _addressDest, *_keySecrets*)
- *keyReinstalRep* (_addressSource, _addressDest, *_keySecrets*)

Ce sont les réponses aux demandes de connexion faites avec les PDUs du type *Ask* de la couche GMM. Ces réponses du *ChiefAdministrator* sont envoyées au *ConcernedMember* avec des PDUs de type *Reply* dans la couche d'administration des clés (GCKM). Les PDUs échangés dans cette couche transportent des clés de session et c'est seulement dans cette couche que les clés seront transportées. Les clés de session envoyées dans ces PDUs correspondent à une demande de (s) clé (s) pour participer à la communication déjà établie dans un certain groupe.

B. PDUs pour des groupes fusionnés

Il s'agit des PDUs transmis entre des entités de la couche GCKM pour échanger des clés de session dans des groupes issus d'une ou de plusieurs fusions.

Pour effectuer un *merge,* deux PDUs servent à distribuer la (les) clé (s) de session entre des chefs et vers les membres.

ChiefLeader → ChiefAdministrator

- *keyDistChiefs* (*_chiefAdministratorId,* _addressDest, *_keySecrets*)

Ce PDU sert à envoyer la clé de session du *ChiefLeader* aux autres chefs qui participent à la fusion d'un groupe.

ChiefAdministrator → Members

- *keyRenMerge* (_addressSource, _addressDest, *_keySecrets*)

Ce PDU sert à envoyer la (les) clé (s) de session du Chef de chaque groupe vers les membres de son groupe.

4.4.2. Sous-couche de gestion des groupes (Group Membership Management GMM)

Cette couche intègre l'ensemble des services intra groupes et inter groupes. Les services offerts dans cette couche (*join, leave, downgrade, upgrade, exclusion, reconnexion, reinstat, merging, spliting*) reprennent la liste des fonctionnalités intra et inter groupes identifiées dans le chapitre 3. Le fonctionnement de ces services se fait en relation avec les services de gestion des clés sous-jacents et l'utilisation des certificats qui donnent des droits aux membres au sein d'un groupe.

Certaines hypothèses, détaillées par la suite, ont été faites par rapport aux rôles des membres participants et aux certificats utilisés. Elles serviront notamment lors de l'analyse des interactions entre les fonctionnalités considérées dans le chapitre 3.

4.4.2.1 Certificats pour la communication des groupes

Deux types de certificats sont utilisés : les Certificats d'Identité (CI) et les Certificats d'Attributs (CAt). Un certificat d'Identité permet à l'utilisateur qui le détient l'accès à un groupe, soit pour lui donner le droit d'une nouvelle entrée dans un groupe spécifique, soit pour lui permettre de changer de position dans son groupe. Ces certificats seront donnés aux utilisateurs en dehors de la session par l'autorité correspondante de façon sécurisée et personnalisée.

Pour être un membre actif dans une session, l'autorité d'attributs du groupe concerné décerne au membre participant le certificat d'attribut. Ce certificat indique les droits qu'un utilisateur détient au sein de son groupe. Les Certificats d'Attribut peuvent encore se diviser en deux types : les Certificats d'Appartenance (CApp) et les Certificats de Commandement (CC). Un Certificat d'Appartenance atteste qu'un membre qui le détient appartient bien au groupe. Un Certificat de Commandement désigne l'utilisateur qui commande le groupe. Plusieurs membres peuvent détenir un certificat d'appartenance, tandis qu'un seul membre détiendra celui de commandement.

Si l'application détermine qu'il faut avoir un membre secondaire pour le commandement du groupe, celui-ci pourra aussi détenir un certificat de commandement (en plus de son certificat d'appartenance) mais il ne l'utilisera que si le chef du groupe quitte le groupe de façon urgente.

Souvent, le chef du groupe prend le rôle de l'autorité d'attribut en distribuant lui-même les certificats d'appartenance aux membres qui entrent. Dans ce cas là, le chef négocie son propre certificat d'attributs, qui s'appelle *Certificat de*

Commandement, car il commande le groupe. Il doit aussi négocier avec ses collègues de même rang d'autres types de certificats de commandement lorsque des groupes fusionnés existent [LIV 2.2], [L2.3].

Les tableaux 4.1 et 4.2 présentés dans la sous-section suivante montrent, en combinaison avec les rôles définis, l'utilisation des certificats dans le protocole.

4.4.2.2 Rôles des utilisateurs dans la communication des groupes

Les rôles communs qui se présentent dans les communications de groupes sont :1). *Concerned Member*, qui représente un membre sur lequel s'applique un service. Il n'a pas de droit de commandement, 2). *Chief,* qui représente le rôle du chef du groupe, 3). *L'Autorité de Certification,* qui est en charge de la génération et la gestion des certificats et 4). *Administrateur,* qui représente le rôle de l'administrateur d'un groupe. Ces rôles sont détaillés dans le chapitre 3 sous-section 3.3.2.

Les tableaux 4.1 et 4.2 montrent la participation des rôles et des certificats dans la mise en œuvre des fonctionnalités du protocole. Dans ces tableaux nous pouvons observer les entités qui participent aux autorisations des mouvements, ainsi que les certificats intervenants lors de l'activation de la fonction demandée.

Fonctionnalité	**Personne qui autorise**	**Certificats vérifiés du *concerned member***	**Certificats accordés**
Join	Chief / Administrateur	C Identité	C Attributs
Leave	-	C Identité	-
Upgrade	Chief / Administrateur	C Identité, C Attributs	C Attributs
Reconnection	Chief / Administrateur	C Identité, C Attributs	C Attributs

Tableau 4.1. Rôles et certificats pour les fonctionnalités qui sont demandées par les membres

Nous indiquons dans le tableau 4.1 les fonctionnalités qui peuvent être demandés par le *concerned member*.

Fonctionnalité	**Certificats vérifiés du *concerned member***	**Certificats accordés**
Reinstal	C Identité, C Attributs	C Attributs
Exclusion	-	-
Downgrade	C Identité, C Attributs	C Attributs
Merge	C Identité	C Attributs
Split	-	-

Tableau 4.2. Rôles et certificats pour les fonctionnalités qui sont imposées par un chief ou par un administrateur

Le tableau 4.2 montre les fonctionnalités qui sont imposées aux membres dans une session active. Le *reinstal* est activé sur ordre de l'administrateur de la session, les autres fonctionnalités le sont par le *chief* du groupe.

La différence entre une « Demande » et un « Ordre imposé » nous indique d'un côté les services qui peuvent être démarrés directement par l'application ou l'utilisateur et d'un autre côté ceux qui doivent être démarrés par une autorité. Dans notre cas, l'autorité *ChiefAdministrator,* qui inclut à la fois le rôle de Chef et le rôle d'Administrateur, est utilisée.

La première colonne des tableaux 4.1 et 4.2 donne le nom de la fonction concernée.

La colonne « Personne qui Autorise » du tableau 4.1 indique qui modère cette fonctionnalité. Par exemple, nous voyons que le service « *Join* », demandé par un *concerned member*, a besoin que l'autorité *Chief/Administrator* donne son accord. Par contre, le service *leave*, n'a pas besoin d'une autorisation. Le membre qui sort doit juste informer le *ChiefAdministrateur* pour que celui-ci prenne les mesures correspondantes (voir sous-section 4.4.2.3.2.3). Le fait d'informer le chef lui permet de mettre à jour la vue courante du groupe. Comme une sortie peut se présenter de manière urgente et définitive, une demande d'autorisation ne parait pas nécessaire.

La colonne « Certificats vérifiés du concerned member » des deux tableaux donne la liste des certificats qui sont détenus par le ConcernedMember et qui sont vérifiés lors de l'activation de la fonctionnalité, pour lui accorder ou non un certain droit. Nous pouvons constater que le Certificat d'Identité doit être vérifié pour tout mouvement dans le groupe. Le Certificat d'Attribut, par contre, s'utilise juste pour se repositionner dans un groupe. Le membre qui se repositionne était déjà dans la session et il doit faire vérifier son certificat d'attributs. Dans la colonne *« Certificat Accordé » n*ous pouvons notamment observer les services pour lesquels un nouveau Certificat d'Attributs sera accordé avec la réponse. Dans tous les cas, le nouveau certificat d'attribut accordé donne le droit d'appartenir à la session. Dans le cas d'un *upgrade*, si le membre qui monte prend un rôle de commandement, alors son certificat d'attribut contient en plus un attribut de commandement. De la même façon, dans le cas d'un *merge*, le chef qui prend la place du *leader* reçoit aussi un certificat d'attribut qui contient en plus un attribut de commandement.

4.4.2.3 Services Intra-groupes

Les services intra-groupes traitent de la dynamique des membres à l'intérieur d'un groupe.

Dans cette partie de l'architecture, nous avons défini la liste suivante de services : **l'entrée (join)** ou la **sortie (leave)** d'un membre, la **montée (upgrade)** ou la **descente (downgrade)** en grade d'un membre, **l'exclusion (exclude)** ou la **réinstallation (reinstal)** d'un membre et finalement la **reconnexion (reconnection)** d'un membre qui, auparavant était connecté et qui a perdu sa connexion.

4.4.2.3.1. Primitives Intra-groupes

Les primitives fournies par la couche GMM servent à la gestion de groupes individuels. Elles sont utilisées par la couche Application qui contient les utilisateurs. Certaines primitives permettent à l'Application d'exécuter des changements de membres. D'autres primitives fournissent de l'information provenant de la couche GMM à la couche Application, qui actualise ainsi sa vision des groupes.

ChiefAdministrator User ←→ ChiefAdministrator GMM

- *downgradeReq* (*_control Information*)
- *downgradeConf* (*_control Information*)
- *reinstalReq* (*_control Information*)
- *reinstalConf*(*_control Information*)
- *exclusionReq* (*_control Information)*

Ces primitives de services que le *Chief/Administrator (*couche GMM) fournit à la couche Application, permettent d'effectuer un *Downgrade* ou d'ordonner l'*Exclusion* d'un membre actif dans le groupe. Le *Reinstal* d'un membre qui avait été exclu peut aussi être démarré par le niveau applicatif*.*

Concerned Member User ←→ Concerned Member GMM
- *joinReq* (*_control Information*)
- *joinConf* (*_control Information*)
- *leaveReq* (*_control Information)*
- *leaveConf* (*_control Information)*
- *upgradeReq* (*_control Information*)
- *upgradeConf* (*_control Information*)
- *reconnectionReq* (*_control Information*)
- *reconnectionConf* (*_control Information*)

Les primitives de cette liste permettent à un utilisateur qui possède le rôle *ConcernedMember,* de changer de niveau à l'intérieur d'un groupe.

Concerned Member GMM → Concerned Member User
- *downgradeInd* (*_control Information*)
- *reinstalInd*(*_control Information*)
- *renKeyExclusionInd* (*_control Information)*

Ces primitives indiquent à un membre qu'il a subi un downgrade, une réinstallation ou une exclusion. En conséquence, elles l'informent que la structure de clés dans la couche GCKM de l'utilisateur vient d'être changée.

Concerned Member GMM ← → Connection GMM
- *connectionJoinReq* (*_control Information*)
- *connectionJoinConf* (*_control Information*)
- *connectionLeaveReq* (*_control Information*)
- *connectionLeaveConf* (*_control Information*)
- *connectionUpgradeReq* (*_control Information*)
- *connectionUpgradeConf* (*_control Information*)
- *connectionReconnectionReq* (*_control Information*)
- *connectionReconnectionConf* (*_control Information*)
- *connectionDowngradeReq* (*_control Information*)
- *connectionDowngradeConf* (*_control Information*)
- *connectionReinstalReq* (*_control Information*)
- *connectionReinstalConf* (*_control Information*)

Ces primitives de service servent à exécuter le changement générique de connexion d'un membre dans le groupe. Un changement de connexion peut être causé par des mouvements simples comme le *join, leave, upgrade ou reconnection.* Mais il peut aussi être causé par des mouvements plus complexes comme le *downgrade et le reinstal,* qui réalisent d'autres tâches, puis se servent du changement de connexion pour intégrer le membre concerné dans sa nouvelle classe hiérarchique.

4.4.2.3.2. Description des services intra-groupes

Nous décrivons dans cette section les services associés aux primitives précédemment décrites.

4.4.2.3.2.1. Sous-service Connection_Actualisation

La factorisation du service générique *d'actualisation de la connexion* a été une des propositions de ce mémoire pour offrir une architecture la plus modulaire possible.

Les services intra-groupes ont un lien avec le sous-service *Connection_Actualisation* qui correspond à la gestion de la connexion des utilisateurs. Ce sous-service se compose d'un appel à une connexion ou déconnexion, puis d'une réponse à cet appel.

Cette actualisation se décompose en quatre sous-services :

- Ask Connection Actualisation **avec** lecture des Certificats CI et CAtt
- Ask Connection Actualisation **avec** Lecture du Certificat CI
- Réponse **avec** Clé de session
- Réponse **sans** Clé de session

La première primitive de service pour la *connection_Actualisation* est utilisée par les services : *reconnection, upgrade, downgrade, reinstal* car ils ont besoin de vérifier les certificats d'Identité et d'Attribut avant d'intégrer le membre concerné dans un groupe. La seconde primitive est utilisée par les services *Join* et *Leave* qui ont juste besoin de vérifier le Certificat d'Identité du membre concerné.

Les services suivants sont une réponse à une demande d'actualisation de connexion. Le troisième service est utilisé par tous les services, hormis le *leave*. Le dernier service, qui correspond à la demande du *leave,* n'envoie pas la clé de session.

Le service *Connection_Actualisation* réagit de façon différente selon l'appelant. Par exemple, si le service qui l'appelle est un *join*, alors le service d'actualisation ne regarde pas le Certificat d'Attributs. Il donne l'accord d'entrée en vérifiant seulement le Certificat d'Identité du demandeur. Par contre, si la demande vient d'un *upgrade*, alors le chef a besoin de vérifier les Certificats d'Identité et d'Attributs actuels du demandeur, pour voir si ce membre a bien la capacité de changer à la classe demandée.

Tous les membres envoient leurs Certificats d'Identité pour être identifiés et pour vérifier que leurs rôles associés leur donnent le droit d'utiliser les services concernés. En ce qui concerne l'utilisation des certificats voir section 4.4.2.2 pour plus d'information.

4.4.2.3.2.2. Service Join

L'entrée d'un membre *(join)* dans un groupe nécessite de changer la clé de session du groupe qu'il rejoint. La problématique posée dans cette section, également traitée par d'autres auteurs, pose le problème d'équilibre entre deux besoins opposés : 1). Maintien de la confidentialité de la communication par actualisation des clés de session après tout mouvement des membres. 2). Non gaspillage de l'utilisation de la bande passante, avec minimisation du nombre de messages de contrôle, y compris les messages d'actualisation des clés de session.

Une première optimisation proposée est de réunir plusieurs entrées, sorties et changement de classe hiérarchique, qui s'effectuent en une seule fois, et pour lesquelles on ne change qu'une seule fois la clé de session. Cette solution, proposée par [BAN 02] porte le nom de *« Bulk operation »* [ZOU 04]. Une fois les opérations multiples regroupées, la clé de session est changée de manière globale. Par la suite, le renouvellement se fait selon les critères établis par l'application. Une première possibilité serait de faire les renouvellements ultérieurs de la clé selon l'axe temporel, par exemple toutes les 12 heures. Une autre possibilité serait de le faire selon l'axe du nombre de requêtes traitées, par exemple après 10 requêtes traitées.

Ainsi, dans notre cas, la solution *« Bulk operation »* a été adoptée selon l'axe temporel en garantissant une confidentialité pré-adhésion et post-résiliation dans la communication des groupe. Elle s'appuie sur des renouvellements périodiques de clés en utilisant une période *T* de renouvellement de la clé de session.

L'Algorithme du *join* est le suivant :

- Membre-> Chef : Demande de *join* en envoyant son CI.
- Le Chef vérifie son certificat d'identité

SI le membre est accepté

- Chef -> Membre : Acceptation du *join* + Clé de Session + CAtt

SI le membre n'est pas accepté

- Chef -> Membre : Refus du *join*

Le service *join* est assuré par un appel au sous-service *ask_Connection_Actualisation.*. Lors de la réponse retour, le membre accepté reçoit un nouveau Certificat d'Attribut.

4.4.2.3.2.3. Service Leave

Suivant le principe *« Bulk operation »* adopté, la sortie d'un membre d'un groupe ne cause pas de modification immédiate de la clé de session. Après le renouvellement périodique futur de la clé de session, les messages ultérieurs deviendront confidentiels pour les membres qui sont sortis.

Dans un *leave*, le chef du groupe est informé pour qu'il prenne en compte la structure du groupe modifié. Si le membre qui sort laisse une place qui rend incohérente la structure du groupe, alors le chef est obligé de faire les changements nécessaires pour supprimer ces incohérences.

L'algorithme du *leave* est le suivant :

- Membre-> Chef : Demande *leave* en envoyant son CI.
  - Le Chef vérifie son certificat d'identité et actualise ses listes
  - Si nécessaire :
    - Le chef change la stucture du groupe, par exemple en faisant monter ou descendre des membres, pour conserver une structure valide.
- Chef -> Membre : Acceptation du *leave*

Le service de *leave* est assuré par un appel au sous-service *ask_Connection_Actualisation,* en lui disant que c'est bien le *leave* qui fait la demande.

Rappel : l'envoi du Certificat d'Attribut n'est pas nécessaire car le *leave* sera autorisé indépendamment des droits de l'utilisateur qui veut sortir.

4.4.2.3.2.4. Service Upgrade

La montée en grade d'un membre est conditionnée par l'autorisation du chef du groupe. Ce dernier analyse les Certificats du membre pour vérifier la possibilité de monter vers la classe demandée.

L'algorithme de l'*upgrade* est le suivant :

- Membre -> Chef : Demande *upgrade* en envoyant son CI et son CAtt
- Le Chef vérifie le Certificat d'Identité et d'Attributs du membre

SI le membre est accepté

- Chef -> Membre : Acceptation de *l'upgrade* + Clé (nouvelle) + CAtt (nouveau)

SI le membre n'est pas accepté

- Chef -> Membre : Refus de *l'upgrade*

Lors de l'upgrade, l'entrée dans la nouvelle classe se fait par un appel au sous-service *ask_Connection_Actualisation,* en lui indiquant que c'est l'*upgrade* qui a fait une demande. La présentation du Certificat d'Attribut est nécessaire pour cela.

4.4.2.3.2.5. Service Downgrade

La descente d'un membre de sa classe d'origine vers une classe inférieure est plus complexe que le montée en grade.

Si l'on fait descendre un membre de sa classe d'origine, et que l'on ne renouvelle pas la clé de session, alors le membre qui descend, du fait de la possession de la clé de sa classe d'origine, peut décrypter les messages de cette classe. Il compromet ainsi les propriétés de confidentialité au sein des groupes, propriétés qui empêchent les membres de classes inférieures de comprendre les échangés par les classes supérieures.

Trois propositions s'offrent à nous :

1. La clé de session est actualisée dans tout le groupe ; le membre n'obtient que la clé de sa nouvelle classe.
2. On exclut le membre. On change son CI et on le réintègre dans sa nouvelle classe.
3. On fait confiance au membre pour qu'il efface un ensemble de secrets qui incluent sa clé initiale. Il ne garde que les certificats nécessaires pour demander son entrée dans la classe inférieure. Son CAtt actuel sert à justifier de son droit de descendre.

Chacune de ces propositions a des avantages et des inconvénients différents. La première proposition parait la plus simple car il s'agit d'un renouvellement de clé. Cependant, le renouvellement envoie la nouvelle clé protégée avec une autre clé que chaque utilisateur détient, y compris le membre qui descend. En utilisant son ancienne clé, le membre peut continuer à accéder aux clés de communication de plus haut niveau. Il faut donc transformer le protocole de gestion de clés de session pour individualiser la transmission des clés de chaque utilisateur, ce qui semble difficile à accomplir et ne parait pas réaliste en termes de passage à l'échelle.

La deuxième proposition parait aussi assez simple. Cependant, sa faiblesse réside dans le fait qu'un CI envoyé par le réseau pourrait être intercepté par une

entité malicieuse. Une amélioration consisterait à transmettre de certificat hors bande, mais les moyens envisagés dans ce cas (courriers, déplacement physique…) sont peu interactifs.

La troisième proposition repose sur la confiance que l'on met dans un utilisateur. Ce dernier doit réellement effacer sa clé pour ne plus comprendre les messages de son ancienne classe.

La troisième proposition sera la solution que nous retiendrons. Elle nous semble la plus appropriée car l'utilisateur qui descend n'est pas dépendant de ressources extérieures. Elle permet de plus une continuité car le CI du membre ne change pas ; seul son Catt est modifié.

L'algorithme du *downgrade* est le suivant :

- Chef -> Membre : Ordre d'exclusion pour le *downgrade* en envoyant son CI
- Le Membre efface la clé de session et l'information nécessaire.
- Membre-> Chef : Demande du *downgrade* pour entrer dans la classe inférieure en envoyant son CI et son CAtt

Le membre est accepté (car il était déjà membre d'une classe supérieure).

- Chef -> Membre : L'acceptation du downgrade + Clé (nouvelle) + CAtt (Nouveau)

L'entrée dans la nouvelle classe, à l'issue du *downgrade,* est assurée par un appel au sous-service *ask_Connection_Actualisation,* en lui disant que le *downgrade* fait la demande, et en ajoutant le CAtt.

4.4.2.3.2.6. Service Exclude

La conception de ce service représente aussi un défi car, comme pour la descente en grade, le membre exclu doit perdre la clé de communication. Renouveler la clé semble être difficile car un membre qui se fait exclure détient toutes les clés anciennes TEK ainsi que toutes les clés KEK. Lors du renouvellement des clés, il pourra accéder et actualiser les nouvelles clés de session.

La difficulté de cette actualisation provient du fait que, si une classe actualise sa clé de communication, alors toutes les classes inférieures doivent aussi actualiser leurs clés. Notre solution suppose que l'administrateur possède un moyen pour obliger l'utilisateur que l'on veut exclure à effacer les secrets de communication qu'il détient.

En ce qui concerne l'actualisation des clés, une fois que toutes les clés de session ont été générées pour chaque classe, un seul message sera envoyé. Le message contiendra l'identification du membre à exclure pour effacer sa clé de communication et d'autres informations secrètes. Il contiendra aussi toutes les clés des autres classes en communication. À l'arrivée du message, le membre à exclure se désactivera et les autres membres ouvriront seulement la partie correspondant à la clé de leur classe, les autres clés restant incompréhensibles pour les membres non destinataires.

L'algorithme de l'*exclude* est le suivant :

- Chef -> Tous : Ordre *exclude* en envoyant son CI
- Le membre à exclure efface ses clés (TEKs, KEKs).
- Les autres membres reçoivent le message. Ils changent leur clé de communication.

4.4.2.3.2.7. Reinstal

Le processus de Réinstallation correspond à l'autorisation de retour pour un membre qui avait été exclu. La solution que nous proposons implique que l'autorité nécessaire se déplace vers l'utilisateur pour lui donner un nouveau CI dont il aura besoin pour continuer sa participation à la communication. Le cas d'une réinstallation se produit assez rarement, ce qui peut justifier le peu d'interactivité de la solution retenue. Il faut de plus réinstaller les certificats d'un utilisateur avec un maximum de sécurité.

L'algorithme du *reinstal* est le suivant :

- L'Autorité de Certification se déplace vers le membre à réinstaller. Elle lui donne un nouveau CI
- Le service de *join*, demandé par le membre qui était exclu, assure la nouvelle entrée.

Le *join* est assuré par un appel au sous-service *ask_Connection_Actualisation,* en lui disant que le *reinstal* fait la demande.

4.4.2.3.2.8. Reconnection

La reconnexion d'un membre peut être nécessaire lorsqu'il a perdu la communication avec son groupe durant un certain temps. La clé de session du groupe ayant changé durant ce laps de temps, les clés qu'il détient ne sont plus correctes. Dans ce cas, le membre déconnecté doit récupérer la nouvelle clé de session en la demandant au groupe en communication. Le membre récupère cette clé auprès du Chef de groupe. Le chef, représentant l'Autorité des Attributs du groupe, connaît tous les membres actifs et tous les membres exclus. En vérifiant ces deux listes, il ne donnera l'accès que seulement aux membres ayant le droit d'entrer.

L'algorithme *reconnection* est le suivant :

- Membre -> Chef : Demande de *reconnection* en envoyant son CI et son CAtt
- Le Chef vérifie le statut du membre déconnecté.

SI le membre est accepté

- Chef -> Membre : Acceptation de la reconnexion + Clé de Session + CAtt (Nouveau)

SI le membre n'est pas accepté

- Chef -> Membre : Refus de la *reconnection*

La *reconnection* est assurée par un appel au sous-service *ask_Connection_Actualisation,* en lui indiquant que le service *reconnection* fait la demande.

4.4.2.3.3. PDUs entre les entités de la couche GMM

Les unités de PDUs échangées entre les entités de protocole de la couche GMM sont listées dans cette sous-section.

Les PDUs de la couche GMM pour les services intra-groupe font partie de la réalisation des services de gestion des groupes. Ils ont comme paramètres l'identification de l'émetteur et/ou du destinataire.

Messages de base :

- *joinAsk* (*_concernedMemberId*)
- *leaveAsk* (*_concernedMemberId*)
- *leaveRep* (*_concernedMemberId*)
- *upgradeAsk* (*_chiefAdminId, _concernedMemberId*)
- *o_downgradeAsk* (*_concernedMemberId*)
- *joinDowngradeAsk* (*_concernedMemberId*)
- *reconnectionAsk* (*_concernedMemberId*)
- *o_reinstalAsk* (*_concernedMemberId*)
- *joinReinstalAsk* (*_concernedMemberId*)

Toutes les demandes de mouvements à l'intérieur d'un groupe sont faites avec l'envoi d'un PDU du type *Ask*. Les PDUs qui commencent avec une lettre *o* représentent un ordre que le *ChiefAdministrator* donne à un certain utilisateur, ce qui est le cas du *Downgrade* et du *Reinstal*. Toutes les autres PDUs du type *Ask* représentent des demandes qu'un membre peut faire à un *ChiefAdministrator.*

Le message *leaveRep* est le seul qui fournisse une réponse. Il est envoyé pour informer le membre sortant si la composition du groupe reste cohérente après sa sortie.

4.4.2.4 Services inter-groupes

Les services inter-groupes gèrent la connectivité entre des groupes afin de les réunir au sein d'une même communication. Cela peut être utile pour des envois que tous les membres dans la session doivent écouter comme, par exemple, des appels d'urgence.

Après avoir fini cette communication commune, les groupes doivent avoir la capacité de se séparer, et de revenir dans leur configuration initiale.

4.4.2.4.1. Primitives inter-groupes

Les primitives présentées dans cette sous-section fournissent les services nécessaires à la gestion inter groupes. Elles permettent de créer une nouvelle session, ainsi que de fusionner ou de séparer des groupes.

Nous supposons que cette session initiale existe déjà, avec une structure de groupes définie.

Nos définissons dans cette section, la liste de services que la couche GMM, à travers un membre autorisé, doit fournir à la couche Application pour maintenir une structure et une connectivité des groupes cohérente. Ensuite, la dynamique des groupes doit être considérée, car elle conditionne l'actualisation en ligne des éléments nécessaires pour sécuriser les groupes.

Certaines primitives informent la couche application sur la configuration des groupes.

Le paramètre *_control information* contient le nom des groupes qui fusionnent ou qui se séparent. Il permet de communiquer au chef de la nouvelle session les informations sur les groupes fusionnées ou séparés.

Application ←→ ChiefAdministrator GMM
- *newSessionReq* (*_control Information*)
- *newSessionConf* (*_control Information*)
- *endSessionReq* (*_control Information*)
- *endSessionConf* (*_control Information*)
- *a_mergeReq* (*_control Information*)
- *a_mergeConf* (*_control Information*)
- *o_mergeReq* (*_control Information*)
- *o_mergeConf* (*_control Information*)
- *splitReq* (*_control Information*)
- *splitConf* (*_control Information*)

Le *Chief/Administrator* de la couche GMM fournit la liste des primitives de service précédentes à la couche Application pour gérer la connectivité entre des groupes. Les primitives *newSession* et *endSession* sont rendues disponibles par le *Chief/Administrator* pour commencer et pour finir une session. Les primitives a_merge et o_merge sont appelées par l'application pour que le *Chief/Administrator (*couche GMM) effectue la fusion des groupes nécessaires. La lettre *a_* indique que la fusion sera demandée, la lettre *o_* indique qu'il s'agit d'un *merge* ordonné par une autorité extérieure à la communication. Un deuxième service que le *Chief/Administrator* offre à l'application est la séparation des groupes fusionnés antérieurement. Ces primitives commencent leur nom par le mot *split.*

Groupe_i_GMM – Application
- *renKeyMergeInd* (*_control Information)*
- *renKeyMergePeriodiqueInd* (*_control Information*)
- *splitInd* (*_control Information*)
- *newSessionInd* (*_control Information*)
- *endSessionInd* (*_control Information*)

Les primitives de type *ind,* utilisées entre les membres d'un groupe *i* (Groupe_i_GMM) et la couche application, servent à informer de l'état actuel du groupe. Les primitives *renKeyMerge* et *split* annoncent à l'application qu'un *merge* ou *un split* vient de se terminer. *renKeyMergePeriodique* annonce un renouvellement périodique de clé, *newSession* et *endSession* indiquent respectivement le début et la fin de la session à la couche application.

Comme précédemment, le paramètre *_control information* contient le nom des groupes qui sont concernées par l'opération.

4.4.2.4.2. Description des services inter-groupes

La solution que nous avons retenue au niveau du traitement de la connectivité entre des groupes est la suivante :

Nous proposons de geler la configuration courante de chaque groupe qui fusionne. Nous gardons la configuration gelée dans un registre des groupes. Ensuite, on ajoute un nouvel élément de communication avec de nouvelles clés de session. Le retour aux configurations précédentes se fait en appelant le service de split. L'élément de fusion est ainsi détruit.

Les fusions que nous considérons s'appliquent dans des groupes avec la

même structure. Pour *« ajouter cette nouvelle couche »* un nouveau chef parmi tous les groupes qui fusionnent doit être élu et des nouvelles clés de session pour toutes les classes doivent être générées et distribuées au sein de chaque groupe.

4.4.2.4.2.1. Merge

Le service de fusion (*merge)* nécessite trois étapes :

1. La synchronisation entre les chefs ;
2. La génération et la distribution de la nouvelle clé de session entre les chefs ;
3. La génération et distribution des clés à l'intérieur des classes, faite en parallèle dans chaque groupe. Cette étape s'appuie sur la structure des groupes de base.

Le service de *merge* est déclenché selon deux possibilités : soit un ordre, qui ne peut pas être rejeté, soit une demande entre des chefs, qui peut être acceptée ou rejetée. L'étape de synchronisation sert à préparer les chefs pour commencer la fusion. Les chefs des groupes qui fusionnent élisent parmi eux un chef leader.

Dans la seconde étape, le chef leader génère la clé de session, puis il l'envoie aux autres chefs.

Dans la troisième étape, chaque chef de groupe génère les clés des classes de son groupe. Il construit le message contenant les clés et il envoie ce message aux membres de son groupe.

L'algorithme du *merge* entre deux groupes est le suivant :

Pour un *merge* se déroulant suite à un ordre de l'administrateur :

- Pour tous les groupes qui fusionnent, envoi concurrent de l'ordre aux chefs des groupes à fusionner
  - ChefAdmin -> Chef i : Ordre de *merge* en envoyant l'identificateur du nouveau Chef, c.à.d. le ChefLeader.
- Envoi concurrent de la clé de session du groupe fusionné aux chefs des groupes à fusionner
  - ChefLeader -> Chef i : Clé de session du groupe fusionné.
- Distributions des nouvelles clés de session dans les groupes (distributions concurrentes dans les groupes).
  - Chef i -> Groupe i : Message contenant les clés du groupe fusionné

Pour un *merge* se déroulant suite à une demande

- Pour tous les groupes qui fusionnent, envoi concurrent de la demande de fusion aux chefs des groupes à fusionner
  - Chef demandant -> Chef i : Demande de *merge*, accompagné de la transmission du Certificat de Commandement

La demande de fusion est toujours accompagnée du certificat de commandement du demandeur. Ce certificat permet au Chef i de vérifier la validité du demandeur, et de sa demande, pour accepter ou refuser la fusion.

- Envoi concurrent de la réponse de fusion des chefs des groupes à fusionner
  - Chef i -> Chef demandant : Réponse de *merge* en envoyant le Certificat de Commandement de Chef i

La validité de la réponse est garantie par le certificat de commandement du Chef i. Le chef leader est élu parmi ceux des groupes qui fusionnent.

- SI la fusion est acceptée par tous les chefs participants
  - Envoi concurrent de la clé de session du groupe fusionné aux chefs des groupes à fusionner
    - ChefLeader -> Chef i : Clé de session du groupe fusionné
  - Distributions des nouvelles clés de session dans les groupes (distributions concurrentes).
    - Chef i -> Groupe i : Message contenant les clés du groupe fusionné

4.4.2.4.2.2. Split

Ce service permet de revenir à la configuration de base des groupes, c.à.d. la composition des groupes avant de les faire fusionner.

L'algorithme du *split* est le suivant :

- ChefLeader ->Group Fusionné : Ordre de split

Les groupes reviennent à leur configuration d'origine avant la fusion.

4.4.2.4.3. PDU entre les entités de la couche GMM pour les services inter-groupes

Les PDUs de cette sous-section contiennent comme paramètres l'identification du *chiefAdministrateur* qui demande/ordonne le *merge/split* et le champ *_control information*, pour connaître les noms des groupes qui fusionnent ou qui se séparent.

Les deux premières unités de messages sont échangées entre les *ChiefAdministrators* pour gérer les fusions entre des groupes.

- *mergeAsk* (*_chiefAdministrateurId*, *_control Information*)
- *mergeRep (_control Information*)

Ces deux PDUs sont transmis entre les chefs des groupes qui doivent fusionner.

Le *split* n'a pas besoin d'échanger des messages entre des Chefs, un PDU est juste envoyé par le *ChiefAdministrator* à tous les membres dans le groupe fusionné pour avertir de la séparation.

- *splitAsk* (*_chiefAdministrateurId, _control Information*)

4.5. Conclusion

Le travail effectué dans ce chapitre a permis d'établir une structure générique de fonctionnement des protocoles de communication pour des groupes sécurisés. Pour mieux situer notre contribution, nous avons positionné la structure proposée par rapport au modèle OSI. Nous avons cherché à proposer un canevas standard qui guide l'analyse et la conception de protocoles de groupes. Ce canevas s'est traduit sous la forme de l'architecture SGCva. Cette architecture inclut toutes les activités que l'on peut trouver pendant le déroulement d'une session active pour des communications de groupes. La description de l'architecture SGCva nous a permis d'identifier, d'une part, les primitives échangées entre les entités des différentes couches d'un utilisateur du protocole et d'autre part, les unités de messages (PDUs) échangées entre deux entités qui appartiennent à la même couche mais qui relient N utilisateurs différents qui font partie d'une même session de communication.
En première approche, nous trouvons deux grandes parties dans cette architecture : la première partie est en charge de transporter les messages de façon sécurisée entre les membres de la session et la deuxième partie sert à gérer la communication de groupe tout en prenant en compte la dynamique des groupes.
Pour le transport sécurisé des messages, nous avons proposé un ensemble de primitives qui peuvent servir de façon standard à représenter tous les services de transport des messages possibles dans des environnements multipoint. La structure des messages que nous avons considérée permet d'intégrer dans les messages le niveau de sécurisation dont l'application a besoin. Des services d'intégrité de confidentialité de protection contre le rejeu, d'authentification de la source de messages et de non répudiation sont pris en compte.
Pour la gestion de la communication des groupes nous trouvons que la nature dynamique de la composition des groupes affecte directement les propriétés de sécurisation des échanges entre les membres de ces groupes. En gardant la relation directe et existante entre les mouvements des membres dans une session active et la garantie d'une communication sécurisée dans cette session, nous détaillons aussi dans ce chapitre les deux modules de base qui composent un protocole de groupes, celui de la gestion des groupes et celui de la gestion de leurs communications.
Nous avons analysé les besoins des membres pour une application de groupes. Nous avons aussi proposé un ensemble de rôles qu'ils doivent prendre. Ceci permet de prendre en compte un ensemble très complet de différentes situations qui peuvent se présenter lors de l'utilisation des protocoles de groupes. Ainsi, nous pouvons dire que l'architecture SGCva est suffisamment complète pour s'appliquer dans une grande variété de cas de communications de groupes. Nous la recommandons comme une base d'étude de protocoles des groupes qui peut s'adapter et se personnaliser selon les besoins particuliers des applications. Son adaptation pour des applications déployées au-dessus de mediums physiques liés à des réseaux variés est aussi possible. Cette adaptation sera notamment illustrée par le chapitre suivant qui s'appuiera sur des médiums avec des caractéristiques différentes.

5 Spécification, modélisation et vérification du protocole

5.1. Introduction

Ce chapitre montre l'utilisation de l'architecture précédente pour un système de communication de groupes spécifique. De façon plus précise, nous proposons dans ce chapitre cinq une méthodologie de développement et de vérification des protocoles de communication que nous appliquons à l'architecture du chapitre quatre [MOT 05]. Cette application s'est déroulée dans le cadre du projet SAFECAST [SAFECAST].

Nous commençons le chapitre en décrivant le projet SAFECAST. Ce projet est le cadre d'étude d'un protocole de communication de groupes assez complexe pour instancier notre travail de thèse. Notre objectif est, au travers de notre architecture, de vérifier les exigences sécuritaires et temporelles qui ont été établies. Nous donnons une description complète de cette application ainsi que du réseau et de la technologie sous laquelle s'instanciera cette étude.

Ensuite, nous décrivons la méthodologie incrémentale de développement du modèle que nous utilisons. L'objectif de cette méthodologie est de produire un modèle proche de la réalité, pour en tirer des résultats concrets. La modélisation du protocole s'est faite à l'aide du profil UML temps réel TURTLE [TURTLE], car nous avons trouvé que cet outil est bien adapté aux besoins de vérification temporels auxquels nous nous sommes intéressés.

Nous avons effectué une analyse temporelle par couches, et extrait pour chaque couche des résultats en termes de satisfaction temporelle. Nous avons cherché à établir, pendant l'analyse et la présentation des résultats, des standards généraux qui servent de base pour l'étude générique des protocoles de communication pour des groupes sécurisés.

Les observations comportementales ont ainsi mis en évidence les relations existant entre les contraintes de sécurité et les exigences temporelles, ainsi que les compromis à effectuer. Des améliorations sont proposées comme résultat de cette analyse temporelle.

5.2. Projet SAFECAST

5.2.1. Description du projet SAFECAST

Les missions de sécurité civiles et militaires nécessitent la coopération de divers corps d'intervention tels que les secouristes, les pompiers, les policiers, les forces de défense. Chaque corps possède ses spécificités en termes de règles hiérarchiques et de commandement. Leurs interventions sur des théâtres d'opération communs impliquent une organisation dynamique des effectifs placés sur le terrain pour former des groupes hétérogènes selon leur corps d'origine, mais structurés et commandés de façon cohérente en fonction des rôles de chacun. Telle est la problématique étudiée par le projet SAFECAST, projet du Réseau National de la Recherche en Télécommunications dédié à l'étude, à la conception et à la validation de fonctionnalités et de mécanismes

garants de communications sécurisées à l'intérieur de groupes dynamiques [EAD 04], [SAFECAST], [LIV 1.1]. La société European Aeronautic Defense and Space Company (EADS), leader européen de l'industrie de l'aéronautique de l'espace et de la défense, au sein de sa division *« Defense & Security »* est le leader du projet SAFECAST. Le Laboratoire d'Analyse d'Architecture des Systèmes du CNRS (LAAS-CNRS), le Laboratoire lorrain de Recherche en Informatique et ses Applications (LORIA), l'Université Technologique de Compiègnes (UTC) et l'Ecole Nationale Supérieure des Télécommunications (ENST) sont les quatre autres participants académiques. Ce projet contractuel a commencé en décembre 2003 et s'est terminé en avril 2007.

Ce projet, qui représente le cadre de notre étude, porte sur des communications dans des réseaux de diffusion multicast de type Professional Mobile Radiocommunication (PMR) utilisés dans des zones d'intervention. Les utilisateurs ont à leur disposition des terminaux numériques, mobiles et sécurisés qui supportent des flux données textuels et audio. Ils communiquent de façon sécurisée à l'intérieur de leur groupe. La section suivante donne une idée générale de cette technologie.

Du fait de la dynamique des groupes, les éléments de sécurité devront s'actualiser en ligne pour garantir la sécurité établie lors des spécifications du système. Le déploiement à large échelle de la diffusion d'information en mode « multicast » au sein d'un groupe est freiné par les problèmes de sécurité et de performance. Des problèmes de respect de la hiérarchie des membres dans le groupe s'y ajoutent.

Le but de ce projet est de développer une architecture globale de sécurité pour des communications multipoint dans un environnement ouvert. Tout membre du groupe peut être simultanément émetteur et récepteur, contrairement aux applications multicast classiques connues sous le nom de « Single Source Multicast ».

Le développement des applications de groupes ayant généré des besoins nouveaux et différents en termes de protocoles, la communication a ainsi évolué d'un mode point-à-point vers une diffusion multipoint. Les mécanismes anciens de protection de données (chiffrement asymétrique) ne sont pas adaptés à la diffusion. De nouvelles techniques basées sur le chiffrement symétrique doivent être considérées dans la solution Safecast [SAFECAST].

Les points à traiter dans ce projet sont en liaison avec la difficulté de passer de communications point-à-point vers des communications multipoint. Le chiffrement des données, la dynamique des membres, la confidentialité en fonction de cette dynamique et la sécurité forment une première liste des exigences des protocoles de groupes étudiés dans ce projet.

La première étape de ce projet est le développement d'une architecture compatible avec les besoins définis pendant l'étape d'analyse du système. La deuxième étape s'attache à la modélisation d'un protocole basé sur cette architecture définie, puis à la vérification du système résultat en se focalisant sur la satisfaction des exigences de sécurité, de temps, et de performance des différents modules de communication de groupes.

5.2.2. Technologie sous-jacente : TETRA et TETRAPOL

Un des objectifs les plus importants des réseaux PMR est de fournir aux organisations des communications dans des situations d'urgence. Il s'agit d'un système sécurisé, robuste et fiable de radiocommunication. Il coordonne efficacement les procédures de communication au sein d'un groupe.

Ce type de réseau est utilisé dans des missions militaires de sécurité civiles, mais aussi dans d'autres organisations comme des aéroports, des réseaux de transport urbains et des sites stratégiques de grandes entreprises industrielles. La sécurité publique, le transport et l'industrie sont les trois principaux domaines d'utilisation de réseaux PMR. Cette technologie représente une solution avancée de radiocommunications qui contribue fortement à l'efficacité opérationnelle des organisations en général.

Les utilisateurs des réseaux PMR exigent un système de communications sécurisé mobile qui leur assure une grande fiabilité et une grande robustesse sur le terrain.

- Un réseau fiable offrant toutes les fonctionnalités essentielles de sécurité ;
- Des terminaux radio faciles à utiliser ;
- Des communications de groupe ;
- Des appels d'urgence ;
- L'échange de données de type statut et géo localisation ;
- L'accès aux bases de données.

Les utilisateurs des trois domaines précédemment cités délaissent actuellement leurs systèmes analogiques pour les remplacer par des solutions numériques PMR. EADS propose deux technologies PMR pour répondre aux besoins de communications sécurisées : TETRA et TETRAPOL [TETRAPOL].

TETRA (Terrestrial Trunked Radio) est un standard de radiocommunications mobile numérique développé par l'*European Telecommunications Standards Institute* (ETSI). Il a été établi comme le standard officiel européen pour la Radio Numérique Mobile Professionelle (PMR) [TETRA].

Le système TETRA constitue une solution complète et sécurisée pour les professionnels. Les terminaux radio sont efficaces et robustes. D'une utilisation simple, les radios PMR sont spécialement étudiées pour correspondre le plus possible aux exigences des équipes engagées sur le terrain.

Tous les fabricants TETRA ont pour but d'assurer la compatibilité de leurs produits, tant au niveau terminaux radio qu'infrastructure. Actuellement, plus de quinze fabricants offrent des réseaux TETRA, des terminaux radio TETRA, ou les deux.

L'évolution des systèmes de Télécommunications ainsi que les changements des besoins des utilisateurs ont abouti à la nouvelle Version 2 du standard TETRA, à la fin 2005 [TETRA]. TETRAPOL est ainsi le résultat de cette évolution.

La technologie TETRAPOL est une technologie de radio numérique. Elle appartient à la même génération que les normes GSM et GPRS, qui sont des standards de téléphonie cellulaire. Actuellement, le standard TETRAPOL est déployé à l'échelle mondiale [EADS], sous la forme de 91 réseaux dans 35 pays, dont 19 européens. Ce déploiement représente une population d'environ 2 millions d'utilisateurs.

5.2.3. Description de l'application SAFECAST

Pour des raisons de cohérence, les sous-sections suivantes s'appuient sur les travaux des chapitres précédents. Nous utiliserons la spécification des protocoles de groupes sécurisés faite dans le chapitre 3, ainsi que certaines définitions liées à l'architecture du chapitre 4.

5.2.3.1 Structure des groupes

Différents corps d'intervention mobilisés dans des situations d'urgence devront être en communication efficacement. Cette communication sert à la gestion des tâches à réaliser et sert aussi à coordonner les utilisateurs dans la réalisation de ces tâches. Comme ces tâches sont souvent critiques et qu'elles nécessitent une grande concentration de la part des utilisateurs, le système de communication doit rester facile d'accès et d'utilisation mais il doit cependant maintenir un certain niveau de sécurité et de confidentialité dans les échanges d'information réalisés. De façon plus précise, la sécurité des émissions et des réceptions est un des principaux problèmes à aborder lors du déroulement de cette communication, problème d'autant plus complexe si nous prenons en compte l'hétérogénéité des membres appartenant aux groupes ainsi que la dynamique du groupe.

En se servant de radios numériques PMR, différents groupes seront en communication de façon indépendante entre eux à l'intérieur d'une session. Les membres sur terrain peuvent ensuite réaliser différents changements à l'intérieur de leur groupe en termes de responsabilité assignée. De façon plus précise, les membres entrent, sortent, changent de rôle ou se déconnectent. Un groupe et ses membres font partie d'un univers, l'univers de la session. La vision d'un groupe sera communément la vision de ses utilisateurs. Ces derniers réalisent leurs activités au sein de leur groupe.

Si un ensemble de groupes est affecté à un même théâtre d'intervention, une communication inter-groupes peut devenir nécessaire. On rassemble des groupes actifs pour faire partager aux membres rassemblés certaines informations, au sein d'une même session de communication. La possibilité de séparer des groupes fusionnés est aussi considérée. Les groupes doivent donc posséder des capacités de fusion et de séparation.

Le niveau de sécurité des communications entre les membres contient des caractéristiques assez complexes au niveau des garanties assurées. La performance du protocole ne doit pas être sacrifiée vis-à-vis des exigences de sécurité établies et des besoins de communication de l'application.

5.2.3.2 Organisation des utilisateurs : groupes et classes

La composition des groupes qui correspond à la description de l'application SAFECAST est conforme au modèle hiérarchique décrit dans le chapitre 3.

La notion de rôles des membres dans des groupes, décrite dans 3.3.2, est la base pour les rôles que nous utiliserons dans ce projet. Dans le cas de Safecast, le membre *ChiefAdministrateur* sera représenté dans le modèle. Il sera l'agrégation des rôles *Chief* et *Adminstrateur*. Il sera chargé de gérer les certificats d'attributs et de commandement.

5.2.3.3 Le protocole retenu au sein du projet SAFECAST

Suite à la description des services de la sous-section 5.2.3, le protocole retenu sera constitué des trois types d'opérations définis dans l'architecture du chapitre 4 : (1) la transmission et la sécurisation des messages échangés ; (2) la gestion des éléments secrets utilisés pour la sécurisation des échanges et (3) la gestion de la dynamique des membres et des groupes dans une session. L'objectif principal du protocole étant la sécurisation des messages échangés, les opérations qui lui sont liées, c'est-à-dire la gestion des éléments secrets et la gestion des groupes, seront effectuées en fonction de cette sécurisation.
L'architecture que nous instancierons de notre modèle est celle du chapitre 4.

5.2.3.4 Hypothèses pour le développement du protocole

5.2.3.4.1. Composition du groupe

Les groupes n'ont pas plus de 10 utilisateurs.

Nous partons de l'existence sécurisée des certificats d'identité et d'attributs nécessaires au moment du démarrage du protocole sous étude. Au début de la session les certificats d'identité sont déjà en possession des membres qui participent à la communication. De la même façon, les certificats d'attributs sont en possession des chefs des groupes. Ces derniers certificats seront manipulés par les chefs si le besoin d'actualisation lors du déroulement d'une session se présente. Ces hypothèses ne restreignent pas notre étude qui se focalise sur la dynamique de la session active. Or, ces certificats sont distribués hors session.

5.2.3.4.2. Sécurité dans la transmission des messages

La confidentialité et l'intégrité de l'information doivent être garanties de bout en bout.

En plus du renouvellement des clés suite aux changements dans les groupes, le renouvellement périodique doit renforcer la protection des éléments secrets. La période est un paramètre variable qui peut aller d'une heure jusqu'à 24 heures.

Certains éléments de sécurité sont fixes. Ils sont donc pré-chargés dans les terminaux des usagers et, si besoin, ils sont renouvelés manuellement. Un exemple d'un élément qui ne change pas dynamiquement est le Certificat d'Identité, lequel peut être révoqué ou bien renouvelé selon les besoins de l'application. Cette gestion des éléments fixes est supposée faite par des acteurs externes au système.

Les temps établis pour l'exécution des opérations de sécurité des éléments variables, lors du déroulement d'une session, ont été fournis par le partenaire EADS. Nous les détaillons dans les tableaux suivants.

Pour le chiffrement des données, nous disposons de deux types d'algorithmes. Si le message est juste échangé entre deux points de communication, un algorithme asymétrique est le plus adapté. Si l'échange est du type multicast ou broadcast, il est préférable d'utiliser des algorithmes symétriques.

Dans le cadre du projet SAFECAST, l'algorithme AES a été recommandé pour le chiffrement symétrique, et RSA a été recommandé pour le chiffrement asymétrique [LIV 3.1]. Le tableau 5.1 donne les temps requis pour effectuer les deux types de chiffrement / déchiffrement.

Temps des actions de sécurité	Type d'algorithme	Temps
Chiffrement	Asymétrique	20 ms + 0.25 µs/octet
	Symétrique	0.25 µs/octet
Déchiffrement	Asymétrique	20 ms + 0.25 µs/octet
	Symétrique	0.25 µs/octet

Tableau 5.1. Temps requis pour le chiffrement des messages

Le temps considéré pour le traitement de la signature est donné par le tableau 5.2:

Temps des actions de sécurité	Type d'algorithme	Temps
Signature	Asymétrique	20 ms + 0.25 µs/octet
Vérification de signature	Asymétrique	20 ms + 0.25 µs/octet

Tableau 5.2. Temps requis pour une signature digitale

Le tableau 5.3 donne les temps pour l'application de l'opération de hachage dans un message

Temps des actions de sécurité	Type d'algorithme	Temps
Temps de construction de Hashing	Hachage	0.25 µs/octet
Temps de vérification de Hashing	Hachage	0.25 µs/octet

Tableau 5.3. Temps requis pour un processus de hachage

Le temps de vérification des certificats inclus dans un message est donné par le tableau 5.4.

Temps des actions de sécurité	Type d'algorithme	Temps
Vérification des certificats	Asymétrique	20 ms + 0.25 µs/octet

Tableau 5.4. Temps pour la vérification des certificats

5.2.3.4.3. Transmission des messages

Nous ne traitons pas la problématique de fiabilité du médium. En première approche, nous considérons que le médium est parfait. La délivrance des messages se fait sans perte. Cette première hypothèse nous donne des temps de transmission minimums, temps suffisants pour une première vérification des exigences temporelles des couches supérieures. Dans le cas où les exigences temporelles ne seront pas respectées avec un médium idéal, il est inutile d'envisager un médium imparfait, qui augmentera de fait les temps de transmission. Nous supposons que le médium parfait contient des mécanismes de récupération, de synchronisation et d'ordonnancement des messages, si besoin est.

En ce qui concerne le calcul du temps de propagation total d'un message il est fait de la manière suivante :

$$T_{Prop}{}^{Total} = T_{Prop}\ Signal + T_{Trans}\ Message$$

Légende :
T_{Prop}Total : Temps de propagation total du message
T_{Prop}Signal : Temps de propagation du signal
T_{Trans}Message : Temps de transmission du message

Le temps de propagation du signal considère les deux débits disponibles dans le réseau. Il se calcule en fonction de la distance entre l'émetteur et le récepteur ainsi que selon le type de réseau physique de diffusion, dans notre cas un réseau radio PMR. Le tableau 5.5 montre les temps considérés.

Débit	**Distance de portée maximale**	**Temps de propagation maximal** $= D_{max} / C \quad (C=3.10^8\ m.s^{-1})$
Bas Débit	3 km = 3.10^3 m	$3.10^3 / 3.10^8$ = 10 µS
Moyen Débit	50 km = 50.10^3 m	$50.10^3 / 3.10^8$ = 167 µS

Tableau 5.5. Temps de propagation maximal du signal

Nous pouvons observer que, pour un réseau bas débit, ce temps de propagation peut valoir entre 0 µS (le récepteur étant situé à côté de l'émetteur) et 10 µS (le récepteur étant situé à 3 km de l'émetteur). Pour un réseau à moyen débit, le temps de propagation peut varier entre 0 µS (le récepteur étant situé à côté de l'émetteur) et 167µS (le récepteur étant situé à 50 km de l'émetteur). En effet, la portée d'un réseau moyen débit est plus grande (50 km) que celle d'un réseau bas débit (3 km).

Le temps de transmission d'un message est calculé en fonction du débit, de la taille et du type du message. Le temps de transmission par octet dans un réseau SAFECAST est donné par le Tableau 5.6 :

Types de nœuds	Type d'info	Débit	Temps de transmission par octets
Bas débit	Voix	6 kb/s	1 302.00 µs/Octet
	Données	3.2 kb/s	2 441.00 µs/Octet
Moyen débit	Voix	100 kb/s	78.13 µs/Octet
	Données	100 kb/s	78.125 µs/Octet

Tableau 5.6. Temps de transmission d'un message

5.2.3.4.4. Eléments composant un message

En ce qui concerne la taille et la composition d'un message, le tableau 5.7 liste les éléments qui peuvent composer un message du protocole. Les tailles en octets de ces éléments sont aussi données.

ELEMENT	Taille (octet)
Numéros de Séquence	8
Type du message	8
Indice TEK	8
Indice KEK	8
Une clé asymétrique	48
Une clé symétrique	64
Clé TEK	64
Clé KEK	64
Hashing	32
HMAC	32
TEK_authentification	32
Identificateur de Membre	100
Certificats d'identité	500
Certificats d'attributs	100
Certificats d'appartenance	100
Certificats de commandement	100
Paquet de données	64

Tableau 5.7. Eléments composant un message

Les deux types de chiffrement, ainsi que la signature d'un message, n'augmentent pas la taille du message concerné.

5.3. Approche d'analyse des protocoles de communication de groupes

Pour prouver la faisabilité du protocole conçu dans le projet SAFECAST, nous effectuons une analyse temporelle des performances du protocole. Nous ne traitons pas dans ce travail la garantie des mécanismes de sécurisation face à la présence de possibles attaques mais notre intérêt est de montrer que le protocole satisfait, au niveau des performances, les exigences temporelles établies dans la phase d'analyse du système.

Nous utilisons un outil de vérification formelle appelé TURTLE dont une des principales caractéristiques est d'étudier le comportement temporel des systèmes.

Dans notre modélisation TURTLE, nous nous sommes intéressés à l'analyse des temps qui apparaissent sur le graphe d'accessibilité du système. Une méthodologie [MOT 05], [MOT 06], [SAQ 05], [FON 07a], [FON 07b] sera proposée dans la sous-section suivante pour réaliser une vérification temporelle incrémentale des protocoles de communication des groupes. Cette technique peut ensuite s'étendre vers une étude des performances des protocoles.

5.3.1. Méthodologie de modélisation des protocoles de communication

D'un point de vue méthodologique, il nous paraît important d'insister sur le caractère incrémental de la modélisation et de la vérification. La réalisation du modèle du protocole a été développée de telle façon que l'on puisse étendre au fur et à mesure les points que l'on souhaite traiter.

L'intérêt de cette méthodologie est de faire une abstraction des problèmes qui seront résolus dans le futur, en les bornant par des hypothèses simplificatrices. Nous proposons de nous focaliser sur un seul point à chaque étape d'extension du modèle. Cette abstraction doit être de plus en plus réaliste jusqu'au moment où l'on arrive à traiter et analyser, dans leur totalité, tous les points impliqués dans le protocole. Ainsi, en partant d'une abstraction à haut niveau du système, nous supprimons certaines hypothèses simplificatrices en commençant par celles incluses dans les couches inférieures de l'architecture du modèle et en terminant par celles contenues dans les couches plus hautes. A la fin du processus, l'intérêt est d'obtenir un système suffisamment détaillé, avec des hypothèses suffisamment proches de la réalité pour permettre une analyse complète et réaliste.

Dans la fig. 5.1 nous pouvons observer les étapes de la méthodologie incrémentale de modélisation des protocoles que nous proposons.

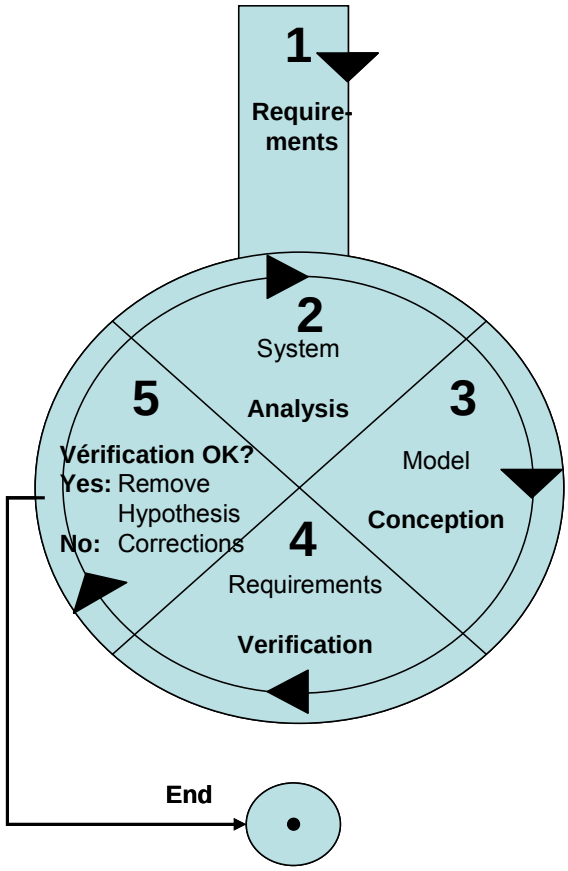


Fig. 5.1. Méthodologie de modélisation et de vérification des protocoles des groupes sécurisés

Le recueil des exigences est la première phase que nous avons effectuée. Dans cette étape, nous mettons en évidence les hypothèses à satisfaire en termes de services requis ainsi que les temps d'exécution à garantir dans le protocole.

Notre méthodologie se poursuit par la phase deux, la phase d'analyse, dans laquelle nous définissons le système à étudier. Le comportement du protocole est défini à l'aide des différents diagrammes UML. L'utilisation d'un diagramme de contexte, un diagramme non UML, nous a aidé à faciliter la description globale du système.

Avec le diagramme de contexte, nous avons déterminé les modules qui composent le système. En construisant les diagrammes de cas d'utilisation (UCD), les modules du système sont définis, leur périmètre est délimité et les acteurs externes sont isolés. Les cas d'utilisation vont guider la modélisation dans son ensemble. Cette analyse fonctionnelle et statique est ensuite complétée par un ensemble de scénarios organisés dans un diagramme d'interaction (IOD) et par des diagrammes de séquences (SD) qui décrivent le comportement dynamique de chaque fonctionnalité.

Dans la phase trois, on aborde la conception du modèle. Cette conception se compose de diagrammes de classes/objets et de diagrammes d'activités décrivant respectivement l'architecture (statique) du système et le comportement des objets. Les diagrammes d'activités intègrent la notion de temps d'exécution des services. Ces temps sont soit de durée fixe, soit de durée variable, dans le cas d'offres temporelles.

La phase de vérification correspond à la confrontation entre la couche de

protocole conçue et le service attendu. Nous cherchons à garantir que les exigences établies, pour les services en modélisation, soient satisfaites (traçabilité des exigences). Ceci fait le lien entre la phase de recueil des exigences et la vérification formelle. Lorsque le système est borné et de taille raisonnable, nous pouvons mettre en œuvre l'analyse d'accessibilité pour effectuer une vérification formelle. En effet, le graphe d'accessibilité représentant l'ensemble des états stables que le système peut atteindre à partir de son état initial, peut être marqué par des informations relatives aux actions exécutées par le système et qui sont utiles à la vérification de la propriété que l'on désire étudier.

En sélectionnant le sous-ensemble d'événements qui font l'objet de la vérification actuelle, nous pouvons minimiser le graphe d'accessibilité et réaliser, dans un contexte UML, une vérification par abstraction apte en particulier à caractériser, sous la forme d'un automate quotient, le service rendu par la couche de protocole modélisée.

Dans la phase cinq, la phase des actualisations du modèle, nous avons plusieurs possibilités : si les exigences établies et vérifiés sont satisfaites, alors un deuxième cycle peut commencer. Ce cycle démarre par une suppression de certaines hypothèses simplificatrices pour passer à une étude plus détaillée. Si ce n'est pas le cas, alors la solution proposée doit être améliorée. On reste dans le même cycle. Un dispositif de correction se met en place dans cette phase. En revenant à la phase deux, il est possible de vérifier si les corrections proposées améliorent le système étudié.

Lorsque le système a été complètement vérifié et prouvé « correct » avec des hypothèses suffisamment réalistes, alors le processus d'étude se termine.

5.3.2. Outil TURTLE

Le langage UML [OMG 03] est utilisé dans de multiples domaines où se pose le problème de la modélisation de systèmes à prépondérance logicielle. La notion de « profil » permet de personnaliser la notation UML de l'OMG pour les besoins d'un domaine d'application spécifique. C'est ainsi que l'on a vu émerger des profils orientés « sécurité » comme UMLSec [JUR 02] ou ForLysa [BUC 04] et des profils « temps réel » tels que TURTLE [APR 04] [APR 05] montrés dans le tableau 5.8. Ces trois profils ont déjà été appliqués à la vérification de protocoles de sécurité.

	Profils UML et outils de vérification		
Profil UML	UMLSec	ForLysa	TURTLE
Référence	[JUR 02]	[BUC 04]	[APR 04/05]
Outil logiciel	SPIN	LYSA	TOOL/RTL/CADP
Langage formel	Promela	Lysa	RT-LOTOS
Orientation de la vérification	Contrôle	Contrôle	Contrôle
Type d'analyse	Accessibilité	Trace	Accessibilité
Type de résultat	MSC	Trace	Automate quotient

Tableau 5.8. Vérification de systèmes de transitions dérivés de modèles exprimés dans un profil UML

En tant que « profil UML temps réel», le langage de modélisation TURTLE spécialise la notation UML (Unified Modeling Language) de l'OMG (Object Management Group) pour les besoins de la modélisation de systèmes dans lesquels le respect de contraintes temporelles est une préoccupation première. Le profil TURTLE a une sémantique formelle exprimée dans l'algèbre de processus RT-LOTOS [COU 00] à laquelle il emprunte en particulier la communication par rendez-vous entre objets et la possibilité d'introduire des actions avec des durées déterminées et de l'indéterminisme temporel dans le comportement des objets.

5.3.2.1 Quand et pourquoi choisir TURTLE ?

TURTLE n'a pas été développé dans le seul but de modéliser et de vérifier des protocoles de sécurité. Sa capacité d'adaptation à ce domaine a été testée avec succès par son application sur le protocole NSPK (Needham Schroeder Public Key protocol) [FON 06].

L'approche TURTLE se caractérise par les points suivants :

- En TURTLE, la vérification formelle n'est pas une activité isolée. Elle s'intègre dans un processus qui démarre avec une analyse (cas d'utilisation, scénarios) et embraye sur la conception (architecture, comportements).
- Une des forces du langage TURTLE réside dans le traitement des aspects temporels.
- Tant le langage de modélisation que les outils de vérification sont bien adaptés à traiter la partie « contrôle » d'une machine de protocoles (incluant ses aspects temporels). Les traitements qui s'expriment dans un paradigme états/transitions se modélisent bien en TURTLE. Par contre, les algorithmes qui manipulent des données complexes ne peuvent pas être détaillés. Le plus souvent le modèle TURTLE représentera l'algorithme par sa durée de traitement estimée et par une abstraction des résultats qu'il produit (par exemple, le succès ou l'échec du cryptage d'une donnée).
- Dans la vérification par abstraction, l'automate quotient permettra de voir si une propriété est satisfaite et dans la négative d'en comprendre les raisons. Contrairement aux approches de type « model-checking », il n'est pas nécessaire d'apprendre un langage basé logique pour exprimer les propriétés. De plus, le résultat de vérification ne se limite pas à une réponse « oui/non » vis-à-vis de la propriété à vérifier.
- TURTLE a été conçu au départ pour produire des modèles abstraits dont la taille autorise la mise en œuvre de l'analyse d'accessibilité. Le développement récent d'un générateur de code Java (non utilisé dans le cadre du projet SAFECAST) a montré que l'on peut dériver une maquette d'implantation à partir d'un modèle TURTLE.

5.3.2.2 Diagrammes support à la modélisation en TURTLE

L'utilisation des diagrammes UML pour la création d'un modèle en TURTLE permet de créer un système complet, réaliste et facile à développer grâce aux outils graphiques disponibles.

5.3.2.2.1. Diagramme de classes / d'objets (Class / Object Diagram)

Ces diagrammes font partie des diagrammes statiques UML. Un diagramme de classes a pour but de mettre en évidence les classes d'un système avec les relations qui les associent. Il formalise les relations entre les classes.

Le diagramme d'objets permet de représenter les instances des classes, c'est-à-dire les objets. Il exprime les relations qui existent entre les objets, mais aussi l'état des objets, ce qui permet de définir des contextes d'exécution. Il donne également une image statique des relations entre les objets du modèle. La fig. 5.2 montre un exemple de diagramme d'objets TURTLE avec les éléments qui le constituent.

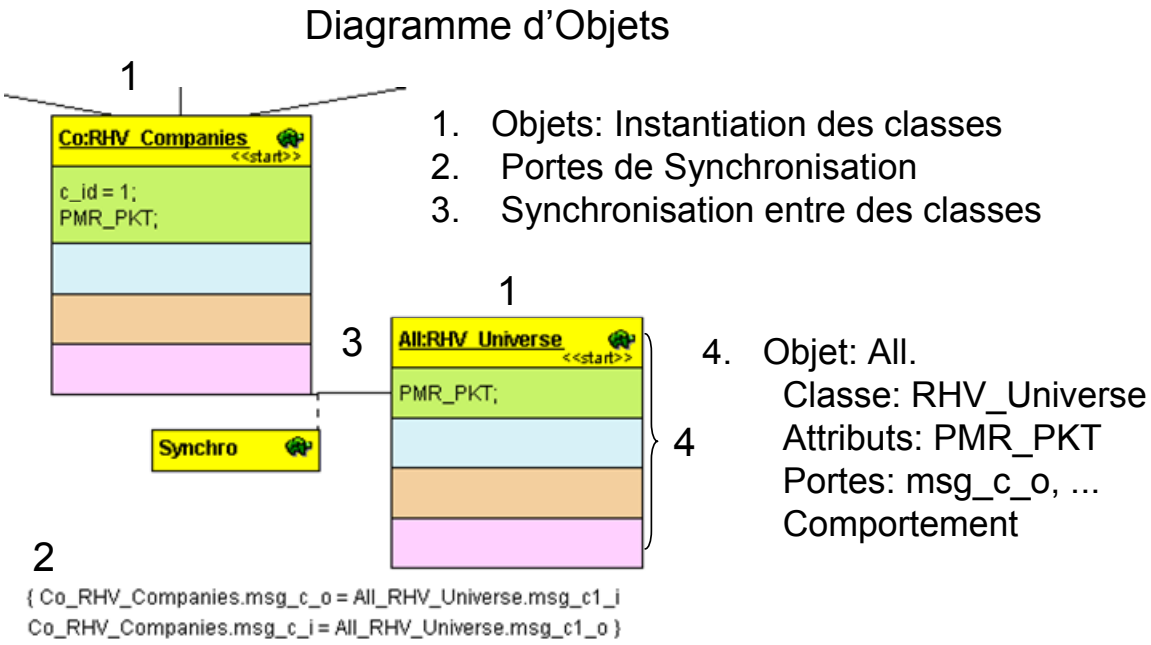


Fig. 5.2. Diagramme d'objets UML

En TURTLE, l'ensemble des diagrammes de classes et d'objets forme l'architecture du système modélisé. Le comportement du protocole sera construit sur cette architecture de base.

5.3.2.2.2. Diagramme d'activités (Activity Diagram)

Ce diagramme est une variante du diagramme *états-transitions*. Il permet de représenter l'exécution de chaque événement du système en fonction de ses états possibles et de modéliser des comportements parallélisables (multi-threads ou multi-processus). La fig. 5.3 montre les éléments que nous allons utiliser dans la construction des diagrammes d'activités TURTLE.

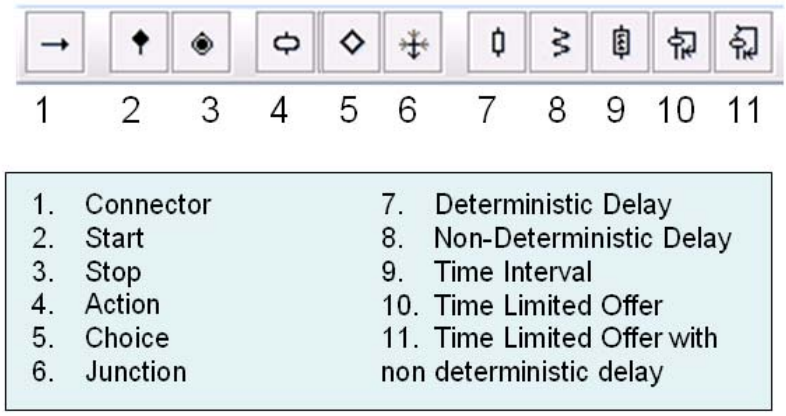


Fig. 5.3. Les éléments d'un Diagramme d'Activités UML

Utilisation des opérateurs : Le « connector » (1) sert à faire les connexions des éléments dans les diagrammes, le « start » et « stop » (2, 3) servent à commencer et arrêter l'exécution d'un diagramme. « action » (4) insère les opérations des diagrammes. Pour permettre de faire un choix, soit indéterminé, ou soit selon les valeurs d'un certain état, nous utilisons le « choice » (5). La « junction » (6) permet l'arrivée de plusieurs processus à un seul point dans le diagramme. « deterministic delay », « non-deterministic delay » et « time interval » (7, 8 et 9) introduisent des délais temporels dans l'exécution d'un processus. Soit ce délai est fixe, soit il appartient à un intervalle de temps dont les limites minimale et maximale sont fixées. La limite minimale est égale à zéro pour l'élément 8, elle peut être supérieure à zéro pour l'élément 9). « Time Limited Offer » (10) permet de faire une offre limitée dans le temps. Si l'offre temporelle n'est pas dépassée, le concepteur indique la série des activités à exécuter. Dans le cas contraire, il précise les autres activités à réaliser. « Time Limited Offer with non deterministic delay » (11) fournit la même fonction que (10) en ajoutant, avant d'effectuer l'offre, un délai non fixe.

Chacun des opérateurs précédemment listés possèdent une syntaxe graphique, facile à comprendre.

5.3.3. Vérification d'un protocole de communication

Généralement, le concepteur dispose d'informations très « floues » concernant les exigences exprimées dans le cahier des charges. Pour pouvoir vérifier ces exigences, il faut avoir des informations de très bas niveau concernant les entités impliquées dans cette exigence, les phases du protocole impliqués, etc. L'utilisateur des outils de vérification ne peut donc pas exprimer correctement les exigences pour qu'elles soient vérifiées sans avoir spécifié un modèle.

Avec les informations collectées dans le cahier des charges et après la spécification du modèle, l'utilisateur dispose des informations pour pouvoir formaliser les exigences.

Généralement, on distingue deux classes d'exigences à vérifier sur les protocoles :

- Les exigences fonctionnelles : ce sont des propriétés générales que doivent satisfaire tous les protocoles : l'absence d'interblocage (« deadlock freeness »), l'absence de fonctionnement cyclique infini (« livelock freeness ») ou encore la possibilité de revenir à l'état initial.

• Les exigences non fonctionnelles : spécifiques à la sémantique du protocole considéré et, donc au service que le protocole est censé offrir (qualité de service, garantie temporelle, propriété de sécurité, mode de fonctionnement particulier).

Pour l'utilisateur des outils de vérification, la difficulté porte essentiellement sur l'expression des exigences non fonctionnelles et sur la comparaison entre le résultat de vérification et ceux du document de spécification du service que le protocole est supposé fournir. Il existe deux approches de vérification pour vérifier ces exigences [COU 91], [VER 90] :

- Approche externe : la vérification par contrôle de modèle consiste à vérifier un modèle en termes d'assertions, par exemple des formules de logique que l'on applique sur le graphe d'accessibilité. Le résultat de la vérification est une réponse oui/non à chacune des assertions.
- Approche interne : la vérification par abstraction appliquée sur le graphe d'accessibilité. Des techniques de réduction basées sur les relations d'équivalence sont utilisées. Le choix d'une bonne relation d'équivalence est déterminant : le graphe doit rester exploitable en comportant peu d'états et avoir un bon pouvoir d'expression pour pouvoir isoler clairement les raisons des failles potentielles. La difficulté essentielle réside dans l'interprétation des automates quotients, et donc de vérifier que les automates quotients obtenus caractérisent bien les exigences informelles du concepteur.

Plus souple dans le pouvoir d'expression, c'est cette dernière approche qui sera mise en œuvre pour vérifier les exigences non fonctionnelles définies dans le cahier des charges du projet SAFECAST.

5.3.3.1 Vérification formelle, vérification par abstraction

L'approche mise en œuvre dans l'environnement TURTLE, qui a pour cible première la vérification de contraintes temporelles, est de type vérification par abstraction. Elle est illustrée dans la fig. 5.4.

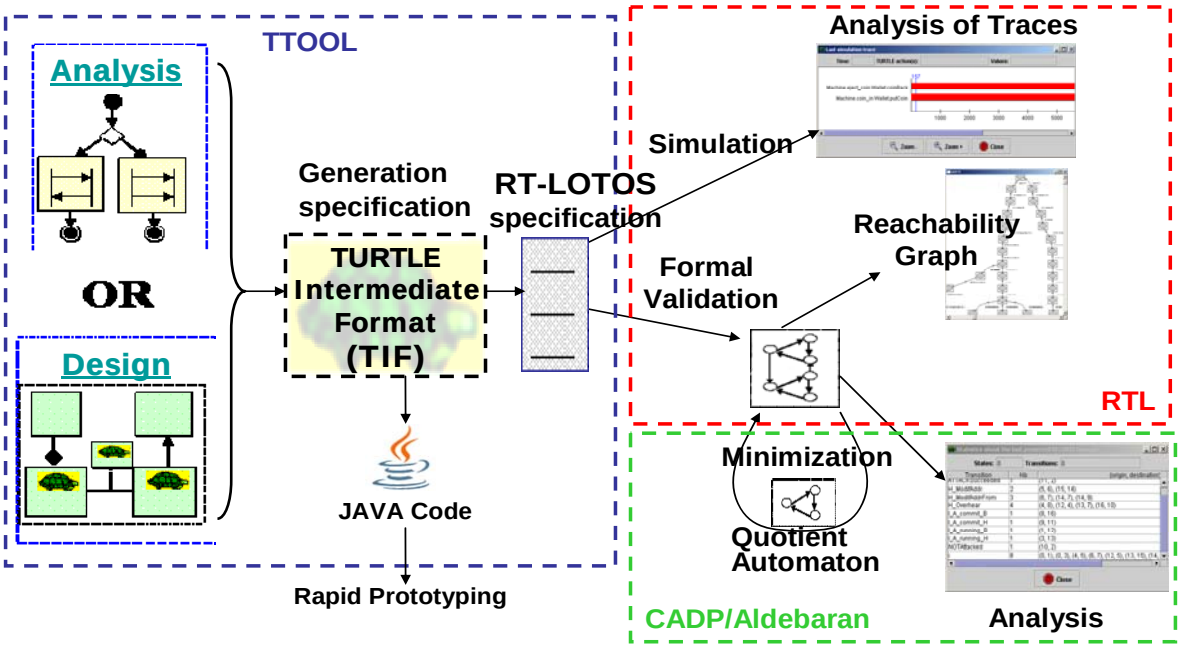


Fig. 5.4. Vérification formelle en TURTLE

A travers l'outil TTOOL, l'utilisateur crée le modèle en passant pour les étapes d'analyse et de conception du système. Un format intermédiaire (format TIF) est généré pour produire une traduction automatique de la spécification vers le formalisme RT-LOTOS.

L'étape suivante crée le Graphe d'Accessibilité à l'aide de l'outil RTL [RTL]. Ce graphe donne ainsi une vue du temps écoulé et des intervalles de temps manipulés. L'utilisation et la visualisation des « *étiquettes* » dans le graphe d'accessibilité est la base d'une vérification simple, claire et correcte. L'outil CADP pour visualiser des minimisations est aussi possible dans TURTLE.

5.3.4. Vérification temporelle

L'objectif principal est d'assurer que le protocole sous étude mette à la disposition des applications des services rapides de transmission mais aussi sécurisés. Bien entendu, leur utilisation doit être simple et l'utilisation de chaque service doit se faire en adéquation avec chaque besoin qui se présente dans un certain scénario pendant le déroulement d'une session.

En partant d'un type d'application comme SAFECAST, la sécurisation de la communication, l'administration des éléments pour la sécurisation d'échanges, et les services pour la gestion de la composition des groupes font partie de notre étude temporelle. Notre but a donc été d'identifier une architecture générique pour des protocoles de communication des groupes mais aussi de bien maîtriser l'influence, au niveau temporel, de cette sécurisation des échanges, de cette administration des éléments secrets et de ces services de gestion des groupes pendant le déroulement d'une session.

En modélisant un protocole type pour une application comme SAFECAST et en faisant une vérification temporelle sur différentes conditions, tant du réseau physique que des services offerts, nous présentons ici les points faibles pour lesquels la rapidité des réponses dans les services offerts peut diminuer.

5.3.4.1 Découpage des temps intervenant dans un protocole de communication de groupes

L'analyse que nous proposons fait une étude temporelle des protocoles de groupes. Elle se caractérise par une identification des temps intervenant selon les différents niveaux de fonctionnement.

Le temps d'exécution d'une certaine fonctionnalité du service de niveau supérieur s'appuie sur les temps qui interviennent dans l'ensemble des services qui s'exécutent au niveau inférieur. Ces services inférieurs peuvent être regroupés en faisant une abstraction.

En nous basant sur ce principe et sur l'architecture présentée lors du chapitre 4, nous pouvons donc lister les différents types de temps qui interviennent pendant l'exécution d'un service rendu à l'application :

- Temps pour la transmission des messages ;
- Temps pour les opérations élémentaires de sécurisation des échanges ;
- Temps des services de la couche GKKM ;
- Temps des services de la couche GMM ;
- Temps correspondants aux scenarii modélisés.

Remarque 1 : il peut arriver qu'un service d'une couche supérieure dans l'architecture, par exemple un service de la couche de gestion des groupes, envoie directement des messages sans passer par des services de la couche inférieure de gestion de la clé du groupe. Dans ce cas, l'influence temporelle des couches non utilisées est nulle.

Remarque 2 : Il peut arriver qu'un ensemble de services appartenant à différentes couches intervienne de façon combinée pour réaliser un service de niveau supérieur.

La fig. 5.5 montre la classification des temps qui interviennent.

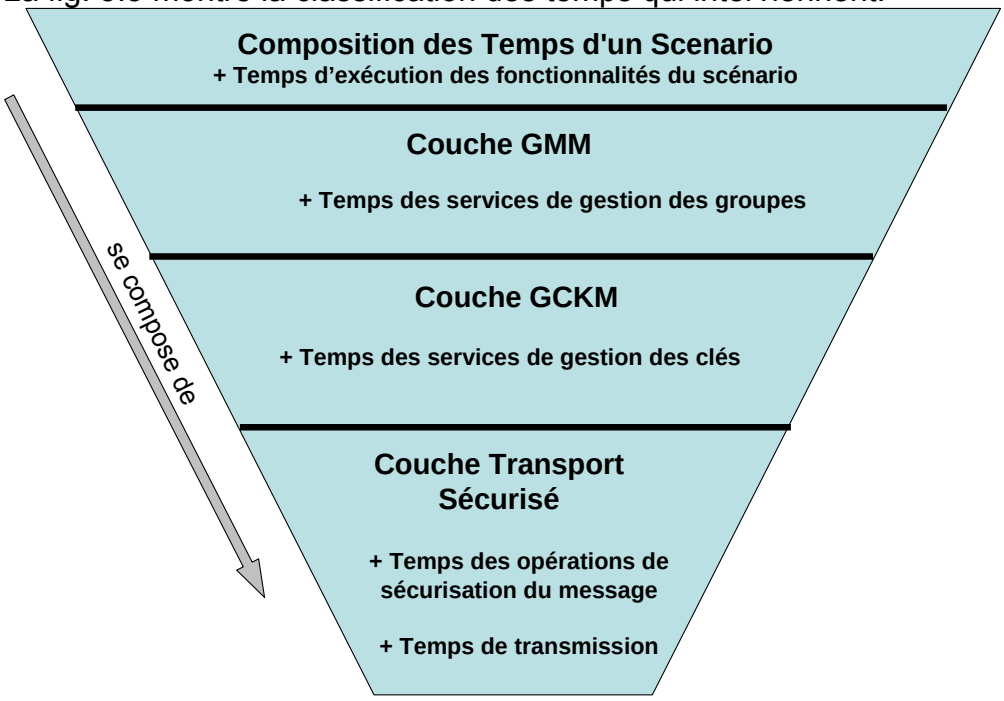


Fig. 5.5 Temps d'exécution qui interviennent selon les niveaux de l'architecture

D'après la classification par niveaux proposée par la fig. 5.5, un protocole de communication de groupes a conscience des services sous-jacents dont il a besoin.

La forme choisie pour représenter les temps qui interviennent dans le protocole est une pyramide inverse. Cette pyramide signifie que le temps qu'une couche prend pour rendre son service est composé du temps correspondant aux processus de cette même couche, auxquels on ajoute les temps que les couches inférieures mettent pour rendre leurs services de niveau inférieur.

Les temps de la couche Transport Sécurisé, la plus basse, sont donnés par les spécifications du système. Le temps correspondant à une fonctionnalité d'une couche plus haute, par exemple le temps d'exécution d'une montée en grade (upgrade), peut être composé de plusieurs temps. Le temps d'exécution de ce scénario fait intervenir les temps des services des couches inférieures.

Nous commençons par définir, dans la couche Transport Sécurisé, le temps de transport des messages pendant leur émission et leur réception. Ce temps se compose de l'addition de trois temps : (1) Le temps de sécurisation du message à l'émission ; (2) Le temps de transmission du message ; (3) Le temps de vérification des paramètres de sécurité à la réception.

Dans la couche GCKM, le temps d'exécution d'un service se compose de l'addition : (1) du temps du processus de la gestion des clés de groupes et (2) du temps nécessaire au transport sécurisé (couche Transport) du message.

Dans la couche GMM, nous considérons le temps du processus qui effectue la gestion des Groupes, auxquels deux types de temps doivent être ajoutés :

1. Le temps optionnel de la réalisation du service de la couche GCKM, si la couche GMM utilise un ou plusieurs services de la couche GCKM.
2. Le temps obligatoire correspondant au service de la couche Transport Sécurisé.

Finalement, au niveau le plus haut, nous définissons un ensemble de scenarii qui permet une étude temporelle plus globale. Un ensemble de fonctionnalités est ainsi regroupé au sein d'un scénario. Leur mise en œuvre permet d'étudier de manière globale le comportement des protocoles de groupes sous certaines conditions. Ainsi, le temps de déroulement d'un scénario peut être composé des temps correspondants aux services impliqués dans des couches inférieures.

L'analyse du comportement temporel d'un système qui suit la structure présentée dans la fig. 5.5 nous facilite les tâches suivantes :

A. Analyse temporelle unitaire de chaque couche, puis par composition.

B. La classification faite dans de cette figure, nous aide à distinguer les différentes étapes par lesquelles un protocole de groupes sécurisé doit passer pour rendre un service de communication sécurisé.

C. Nous pouvons aussi, avec ce découpage des temps, bien distinguer la chaîne des temps d'exécution des sous-services dont un certain service a besoin. Ceci est très utile pour reconnaître les services des couches inférieures qu'un certain service de la couche *N* devra utiliser. Nous pouvons aussi identifier la liste détaillée des temps intervenant tout au long de l'exécution de ce service.

   Par exemple, si nous étudions un certain service *X* de la couche « Application », ce service *X* peut appartenir à un des deux types suivants :

   1. Les services qui utilisent directement la couche transport pour des envois/réceptions directs de messages entre utilisateurs, sans avoir besoin des services de la sous-couche GMM et, par conséquence, de la sous-couche GCKM.
   2. Les services qui auront besoin des services de Gestion des Groupes et, peut être aussi des services de Gestion de la Clé du Groupe. Par exemple, ce service peut être une nouvelle entrée d'un membre dans un groupe actif pour laquelle l'application devra passer par la couche GMM pour effectuer le service de *Join*, mais aussi, par la couche GCKM pour envoyer la clé de session du groupe actif au membre.

D. Cette classification nous a aidés à l'identification des services qui doivent être améliorés pour respecter les critères de performances et/ou les critères de sécurité. Une analyse fine des fonctionnalités nous permet de sélectionner les services dont la sécurité peut être renforcée tout en quantifiant le surcoût temporel que cela implique. Le rapport entre la sécurité gagnée et l'efficacité temporelle sacrifiée servira de guide pour déterminer le protocole de communication idéal dont l'application a besoin.

En connaissant à la fois la liste des temps maximaux permis pour l'exécution des fonctionnalités du protocole (les exigences temporelles) et les conditions temporelles offertes par l'environnement (les hypothèses sur le médium et la composition du groupe), le modèle produit nous permet de vérifier le respect ou non des exigences temporelles établies, avant la réalisation finale du système.

Cette approche d'étude à priori est complémentaire d'une approche à postériori, faite une fois le système réalisé. L'approche à postériori s'appuie sur l'analyse de données réelles, extraites du système réalisé. Notre approche méthodologique, innovante et facilement accessible, peut cependant s'inscrire dans une démarche plus générale qui inclurait les deux types d'études.

5.3.4.2 Détail de la vérification par abstraction utilisée

Comme énoncé précédemment, l'étude des modélisations TURTLE se base sur une analyse par bisimulation du graphe d'accessibilité, analyse appelée aussi vérification par abstraction. Les graphes d'accessibilité générés lors de la vérification, graphes de grande taille et complexes, sont réduits avec l'outil CADP/Aldebaran [CADP].

Cette minimisation permet de présenter à chaque fois, une vision partielle correspondant à la fonctionnalité qui nous intéresse. De façon plus précise, pour rendre dans le graphe résultant les informations plus facilement analysables, nous avons donné, dans le modèle, des noms représentatifs à certains ports [FON 06]. Nous avons aussi ajouté des ports dédiés exclusivement à la présentation des résultats. Par exemple, pour représenter le début et la fin de montée en grade d'un certain membre dans un groupe en communication, nous avons ajouté les ports AskUpgrade_Req et AskUpgrade_Conf. Ces ports représentatifs apparaissent dans le graphe d'accessibilité. Ils facilitent la compréhension des résultats.

La recherche finale des erreurs sur des graphes minimisés reste encore manuelle. Des travaux pour automatiser cette observation et l'affichage des résultats ont été proposés par Benjamin Fontan, puis exposés dans sa thèse [FON 07b]. Les améliorations portent sur l'utilisation des observateurs pour la vérification. Ces observateurs permettent en particulier de « factoriser » les événements au niveau de l'automate quotient résultant de la minimisation du graphe d'accessibilité et d'avoir ainsi des résultats de vérification aisés à interpréter et à transmettre.

5.4. Modélisation et résultats de vérification du protocole SAFECAST

L'expérience montre que plus l'on cherche à analyser de matière pointue et exhaustive une des « facettes » d'un système - par exemple pour apprécier le niveau de sécurité offert par ce système ou encore pour vérifier qu'il respecte un certain nombre d'exigences temporelles - il faut avoir recours à un langage de modélisation spécialisé. Le pouvoir d'expression de ce langage est en général très grand dans la « facette » qu'il permet d'aborder mais très limité dans les autres facettes du système. On peut dire que les langages formels dotés de techniques d'analyse puissantes sont souvent « hyper spécialisés ». Ainsi, il est rare qu'un même outil traite à un même niveau la partie « contrôle » d'un

protocole (échanges de messages et respect des contraintes temporelles) et la partie « données » (format des messages et sémantique des données véhiculées).

Ceci nous amène à dire que l'étude d'un système complexe tire bénéfice de « modélisations croisées » impliquant plusieurs langages et outils de modélisation. C'est l'approche retenue dans la cadre du projet SAFECAST qui « croise » une modélisation orientée « données » et « sécurité » à l'aide de l'outil AVISPA [ARM 05] avec une modélisation orientée « contrôle » et « contraintes temporelles » mise en œuvre par l'outil TURTLE.

On retiendra in fine que tout langage de modélisation (formel ou pas) a un pouvoir d'expression forcément spécialisé et que de ce fait un modèle ne va adresser qu'un sous-ensemble des exigences exprimées en phase amont du cycle de vie. La facette que nous avons étudiée, dans le cadre de ce mémoire, correspond à une modélisation orientée contrôle. Une étude des performances des protocoles, à travers la vérification temporelle des hypothèses établies est réalisée.

Ayant connaissance de la méthodologie de modélisation et des techniques de vérification que nous proposons pour le développement du modèle et la vérification des hypothèses établies, cette sous-section montre l'application de cette méthodologie lors de l'application de l'architecture générique du chapitre 4 à une modélisation TURTLE. Les résultats de vérification de cette sous-section ont été générés sur la base de la vérification formelle déjà introduite, qui analyse et étudie de façon fine les contraintes temporelles qui interviennent.

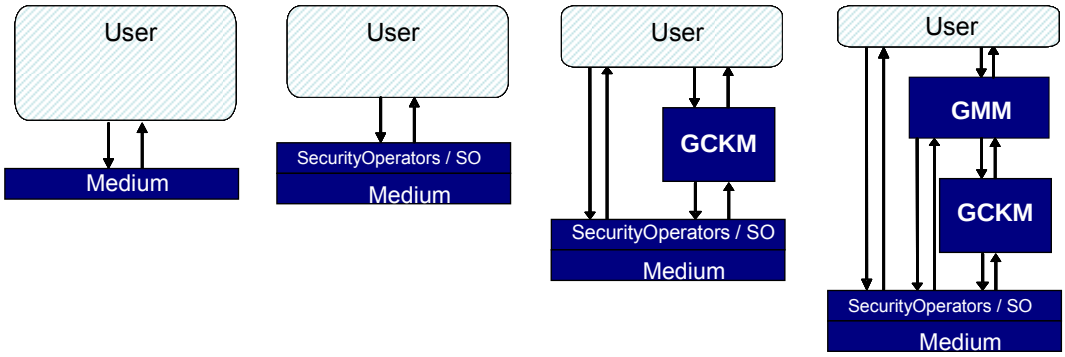
5.4.1. Modélisation du protocole

L'analyse préliminaire de notre système a été faite au moyen de diagrammes comportementaux ou dynamiques d'UML. De façon plus précise, les diagrammes de séquence considérés ont été utilisés pour détailler le comportement de chacune des fonctionnalités du protocole que nous avons développé. Ils ont servi de base pour le développement des diagrammes de classe, d'objets et de comportement du modèle.

En suivant notre méthodologie, la première étape a porté sur le développement et l'analyse du médium. Nous avons commencé par une architecture réduite à deux couches : 1. La couche *Medium*. 2. La couche *User,* qui abstrait le système et l'application. Cette première architecture est montrée dans fig. 5.6 (a). L'hypothèse simplificatrice majeure qui correspond à ce modèle est que les messages à échanger ont été déjà construits et sécurisés. A ce niveau de détail du modèle, nous ne nous sommes pas intéressés au contenu du message. Nous avons juste identifié le type de message et ses caractéristiques physiques, afin de le sécuriser et de l'envoyer ou le recevoir.

L'étape suivante a été le développement des aspects de sécurisation des messages dans le modèle. En enlevant des hypothèses simplificatrices concernant la sécurité des échanges, nous avons construit un module de plus. La nouvelle architecture du système est montrée dans la fig. 5.6 (b). L'aspect sécurité des données ne fait pas partie de l'étude. Nous avons étendu l'aspect contrôle temporel des opérations de sécurisation en vue de l'étudier. Nous traitons principalement, les temps qui interviennent pendant des transmissions sécurisées des messages. Dans cette étape-là l'architecture est composée des

sous-couches suivantes : 1.1. La couche *Medium*. 1.2. La couche *Security Operators* et la couche 2, *User.*



(a) (b) (c) (d)

Fig. 5.6.Développement incrémental du modèle

Ainsi de suite, la construction du modèle a été réalisée progressivement. Les couches *« Gestion de la communication des groupes (GCKM) » et « Gestion des groupes »* ont été développées pour mettre à disposition de l'application une pile de fonctionnalités prête à être utilisée. Voir fig. 5.6 (c), (d).

Cette modélisation incrémentale, associée à la vérification par abstraction par étapes, est une méthodologie qui peut s'appliquer à tout langage de modélisation où les entités (ici les objets) structurant l'architecture communiquent par rendez-vous, lorsque les modèles sont « compatibles » avec des systèmes de transitions.

5.4.1.1 Architecture TURTLE du protocole SAFECAST

Les opérations que nous pouvons représenter, l'inclusion de la notion du temps des activités, et la création d'une architecture du réseau très semblable au réseau réel des utilisateurs et des groupes en communication, sont des points forts de TURTLE qui ont influencé notre décision de l'utiliser comme outil de modélisation. La possibilité de faire une vérification formelle sur le protocole sous étude a été aussi importante dans notre décision.

La fig. 5.7 montre une partie significative du modèle implémenté. La structure générale de l'architecture construite en TURTLE étend l'architecture générique définie lors du chapitre 4, avec des sous-couches qui servent à fournir, de manière modulaire, tous les services de la couche à laquelle ils appartiennent. Ces sous-couches TURTLE sont dépendantes du problème à modéliser. Dans notre cas, elles implantent les spécificités du système SAFECAST en termes de groupes manipulés. Ainsi, une couche de l'architecture générique, peut être composée d'un ensemble de sous-couches dans l'architecture TURTLE. Par exemple, la couche User/Application, se compose de trois sous-couches (scenarii, fonctionnalités et rôles) dans l'architecture TURTLE. La couche Physique, elle aussi, est composée de trois sous-couches (classes, compagnies et univers).

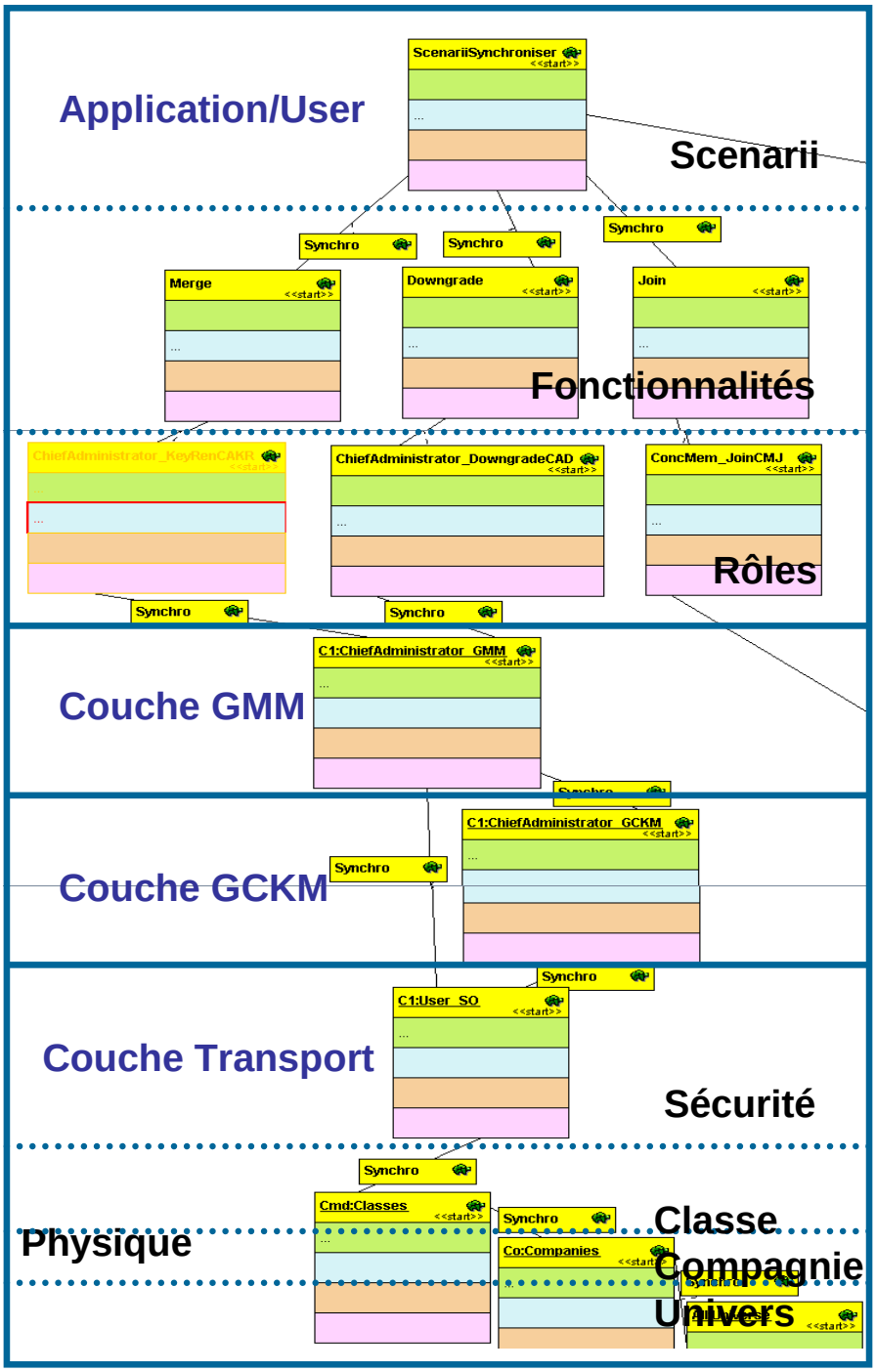


Fig. 5.7. Architecture du protocole SAFECAST développée en TURTLE

5.4.1.2 Couche Transport

La modélisation du réseau physique a été développée dans la couche Transport –Physique (membres + classes + compagnies + univers) de l'architecture TURTLE. Nous avons fait de cette couche physique une représentation des éléments dont nous avons besoin pour la vérification.

Nous insistons sur deux éléments importants à prendre en compte pendant la conception de la structure du modèle de réseau physique :

- La nature des applications telles que des interventions des pompiers, de secouristes et de gendarmes n'a pas comme priorité une communication massive. Nous n'avons donc pas à étudier en priorité le

problème de surcharge du médium. La vérification que nous faisons n'est pas donc focalisée sur l'étude d'augmentation du trafic, mais sur la rapidité de la réponse du protocole.

- Dans un réseau physique du type radio PMR et pour des missions comme celles de SAFECAST, l'augmentation des membres actifs en communication ne sacrifie pas l'efficacité dans la transmission car le médium utilisé (l'air) n'est pas limité et la taille des messages échangés n'est pas importante.

Afin de ne pas avoir un graphe d'accessibilité peut être impossible à construire, nous avons regroupés certains comportements identiques. En effet, étant donné que plusieurs membres ont le même comportement dans le modèle, nous avons identifié les utilisateurs type du système. Nous avons modélisé au niveau de ces types.

Nous avons ensuite limité la représentation du système à des groupes de composition minimale, mais représentatifs de tous les membres sur terrain. On trouve dans ces scenarii tous les membres type et les seuls utilisateurs indispensables. Les membres qui n'ont pas d'influence dans le scénario étudié ne sont pas représentés, ou bien sont représentés de manière abstraite et agrégée.

Le modèle du médium est donc formé de deux groupes. Chaque groupe possède trois classes. Chaque classe comprend deux membres. Seuls les membres types qui participent à une session ont été modélisés.

Les services de sécurisation des messages font partie de la couche Transport.

Les couches GCKM et GMM du modèle TURTLE contiennent, comme dans notre architecture, les fonctionnalités décrites dans le chapitre 3 et 4 pour effectuer des services isolés de gestion des clés ou de gestion de groupes.

5.4.1.3 Couche Application : modélisation orientée vérification

Une fois les fonctionnalités du protocole dans les couches Transport + CGKM + GMM modélisées, conformément à notre méthodologie, nous avons étendu le modèle de vérification avec les composants de la couche *« Application ».*

Cette couche est composée de trois niveaux. Nous considérons que, pour présenter les résultats de vérification de manière à faciliter l'identification des problèmes, il est important de bien distinguer les scenarii, les fonctionnalités sous étude et les personnes participantes. Ainsi, nous proposons de décomposer la couche Application selon les trois sous-couches suivantes (montrées dans la fig. 5.8) :

1. Séquenceur de scénarii : Dans ce niveau, nous avons spécifié les scénarii incluant toutes les fonctionnalités étudiées. Ces fonctionnalités apparaissent dans les scenarii via des appels aux méthodes.

   Dans un premier temps, nous avons modélisé et vérifié individuellement chaque fonctionnalité du système SAFECAST. Puis, en fonction des besoins et des hypothèses à vérifier dans [LIV 1.3] [LIV 1.4], nous avons identifié des scenarii représentatifs des conditions réelles de mise en œuvre sur les théâtres d'opération. Ces scenarii regroupent des ensembles de fonctionnalités. Le scénario construit pour vérifier l'hypothèse établie sur le temps moyen d'une entrée tardive dans une communication chiffrée est un

exemple où plusieurs fonctionnalités participent à cette vérification : celle du *« join »,* du *« reconnect »* et du *« reinstal ».*

De façon plus spécifique, le *« Séquenceur des scénarii »* a été très utile pour représenter à tout moment, la partie de l'application que l'on cherche à vérifier. Nous avons changé ce séquenceur, autant de fois que l'on avait besoin, sans toucher ni les fonctionnalités du protocole modélisé, ni le comportement des membres actifs dans la session.

2. Fonctionnalités : on représente chaque fonctionnalité par une classe. Le niveau au dessus (Séquenceur de scenarii) invoque les différentes classes de fonctionnalités pour que celles-ci puissent être exécutées. A partir de là, la classe représentant une fonctionnalité invoquée se synchronisera et s'exécutera lors de chaque activation du rôle utilisateur concerné.

3. Rôles correspondants aux scénarii : à chaque fonctionnalité correspondent des rôles d'utilisateurs différents (par exemple le chef de groupe n'aura pas le même rôle pour un renouvellement de clé dans un groupe de base et dans un groupe fusionné). Les classes qui correspondent aux différents rôles d'utilisateurs sont ensuite connectées aux entités de protocoles de la couche GMM ou de la couche GCKM.

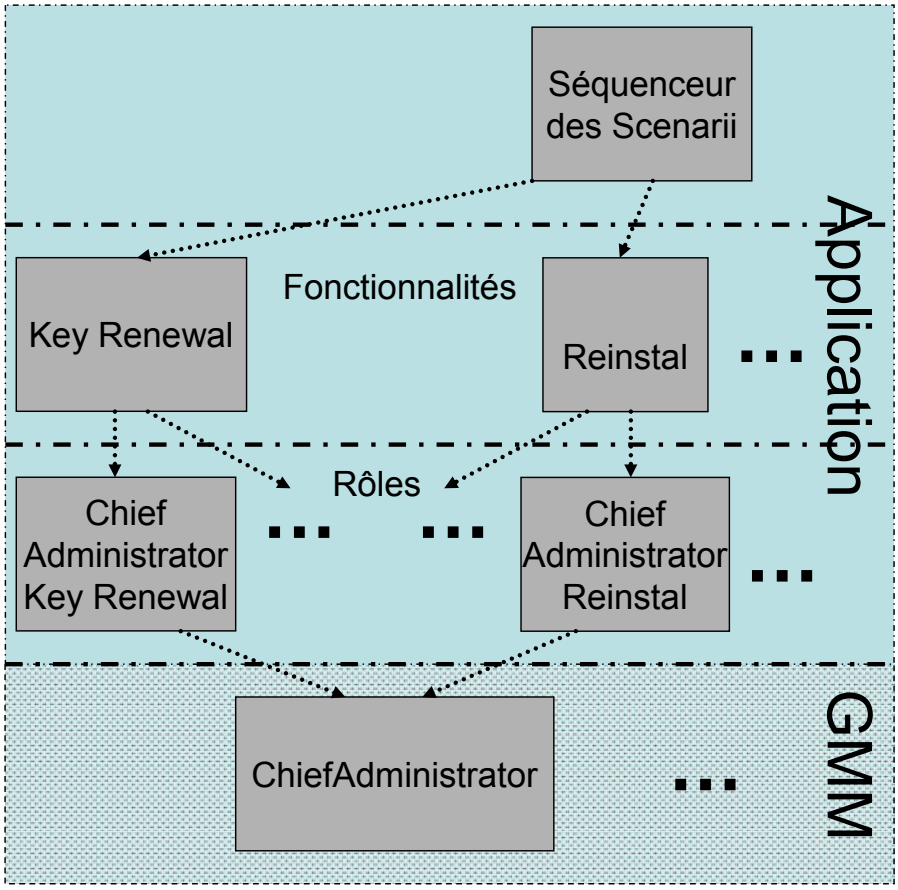


Fig. 5.8. Composition de la couche Application

La fig. 5.8 montre les trois sous-couches proposées pour la couche d'application. Dans la sous-couche la plus haute, nous pouvons observer le *Séquenceur des Scenarii*, dans la sous-couche suivante deux exemples de fonctionnalités du protocole, *Key Renewal* et *Reinstal*, sont présentés et dans la sous-couche la plus basse, l'instanciation des deux rôles différents, reliés à la fonctionnalité qui les utilise : *ChiefAdministratorKeyRenewal* et *ChiefAdministratorReinstal*. Ces deux rôles correspondent à un seul membre *ChiefAdministrator*. Notons que, dans la couche GMM, seul l'utilisateur *ChiefAdministrator* est présent. Cependant, deux rôles lui sont attribués : celui du renouvellement de la clé et celui de la réinstallation d'un membre antérieurement exclu. Chaque rôle instancié et lié à une seule personne, en réponse aux fonctionnalités qui l'utilisent, donne à cette personne un ensemble de droits correspondants à ce rôle. La couche GMM de cette fig. 5.8 nous permet aussi d'observer qu'un seul membre (*ChiefAdministrator*) est à la fois en charge des opérations de l'Administrateur (pour la réinstallation) et du Chef (pour le renouvellement de la clé de session).

L'objectif de la modélisation incrémentale proposé dans la sous-section 5.3.1 et de son application au protocole SAFECAST lors de la sous-section 5.4.1, est d'aboutir à un protocole de communication des groupes selon l'architecture *SGCva* proposée dans le chapitre 4 (fig. 4.3). Cependant, afin de permettre la construction d'un protocole spécifique à l'application sous étude, nous avons encore proposé de sous-diviser les couches dont nous avons besoin : la couche physique et la couche application. La fig. 5.8 montre un exemple de cette particularisation dans la couche application.

5.4.1.4 Un exemple de modélisation, fonctionnalité « upgrade »

Une des exigences temporelles que nous devons vérifier dans notre modèle est que *« le temps de détection d'une violation d'intégrité dans le protocole doit être inférieur à 10 secondes »*. Pour mieux clarifier notre approche, nous allons présenter la vérification détaillée de cette exigence dans cette section. Le *« Upgrade »* représente un exemple de modélisation pour la vérification que nous avons suivi de manière générale lors de l'étude du système SAFECAST.

Le *« Upgrade »* d'un membre peut se présenter lorsqu'un autre membre a besoin de quitter la session. Si le membre qui s'en va a des responsabilités spécifiques au sein de son groupe, il devra donc être remplacé par un membre *um* qui provient d'un niveau inférieur. Le membre *um* transmet une requête de montée au chef du groupe *(cm : ChiefMemberCom)*, qui peut avoir aussi le rôle d'administrateur et donc s'appeler *ca : ChiefAdministrator.* En faisant une vérification des certificats d'identité et d'attributs du membre *um,* le chef *cm/ca* accepte où refuse la montée de *um*. Le scénario *« Upgrade »* est montré dans la fig. 5.9.

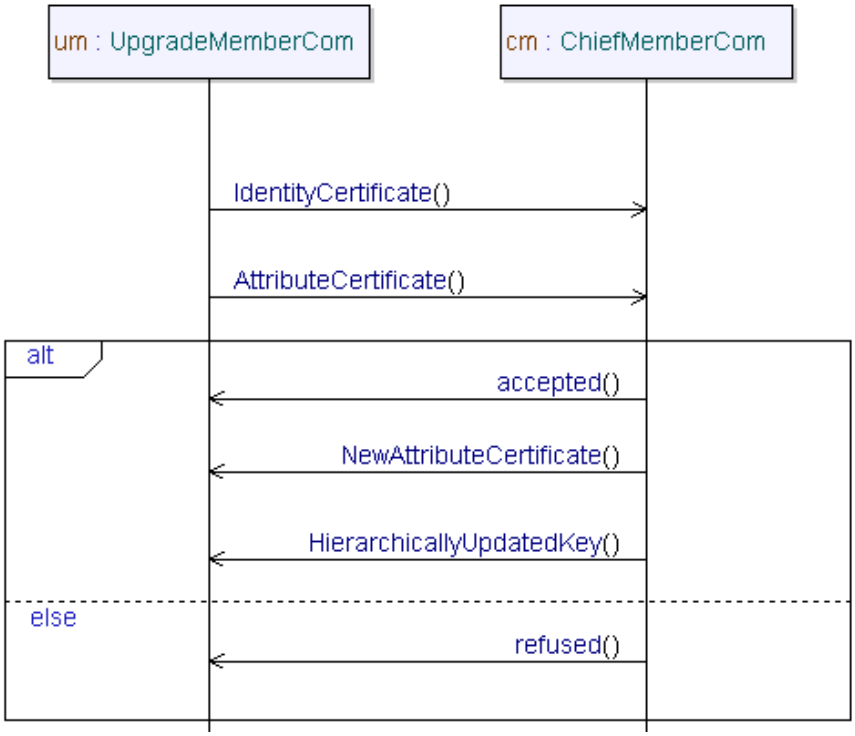


Fig. 5.9. Diagramme de séquence de la fonctionnalité Upgrade

Dans ce diagramme, la demande est effectuée en transmettant les messages qui contiennent le certificat d'identité et le certificat d'attributs. L'acceptation est obtenue avec le message d'acceptation correspondant, suivi du nouveau certificat d'attributs et de la clé de communication de la nouvelle classe. Le refus est signalé avec le message *refused( )*.

Le tableau 5.9 montre l'implantation de la couche Application pour effectuer l'*Upgrade* d'un membre. Il est composé des trois colonnes : la première colonne montre l'objet implémenté ; la deuxième montre le comportement de cet objet ; la troisième colonne détaille de façon textuelle cet objet, son comportement ainsi que les activités que nous pouvons inclure dans cet objet.

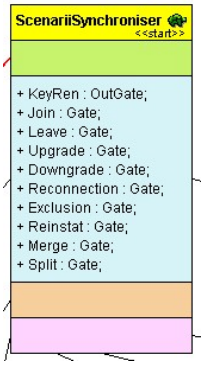
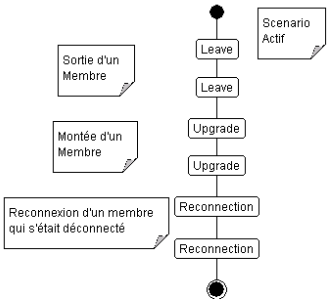
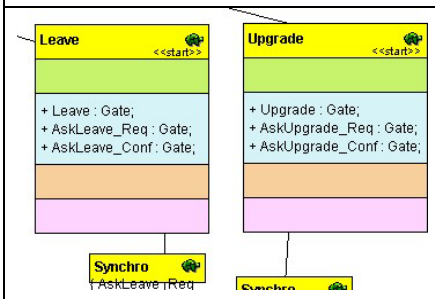
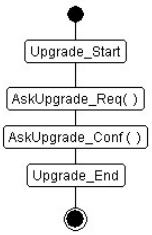
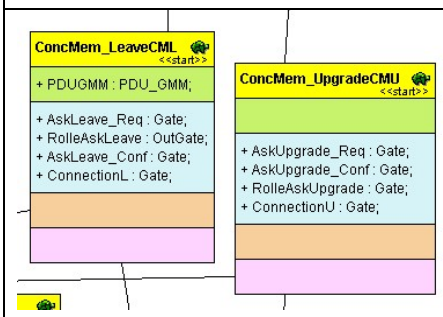
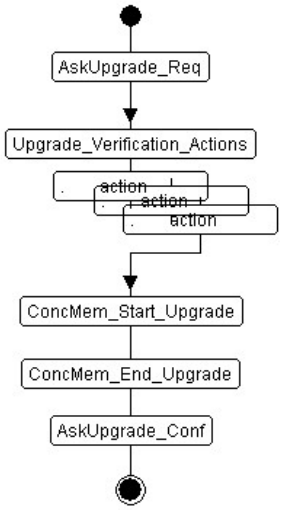
Couche Application. Montée d'un membre.		
Objet	Diagramme de Comportement	Commentaire
		L'objet ScenariiSynchroniser (dans la première colonne) contient tous les ports qui sont nécessaires pour construire des scenarii. Les ports de la partie bleue ont une connexion par rendez-vous avec leurs fonctionnalités correspondantes. Le diagramme de comportement (dans la deuxième colonne) montre le scénario construit afin de tester le *upgrade*. La sous-fonctionnalité *Leave* démarre l'exécution de l'*Upgrade.* La dernière sous-fonctionnalité (*Reconnection)* intervient à la suite de ce scénario.
		Le comportement de l'objet *Upgrade* a été abstrait. Il fait appel au Rôle spécifique du Membre qui doit exécuter cette fonction. Notre intérêt ici est de matérialiser, à travers l'objet *Upgrade* l'appel à la fonctionnalité de la montée d'un membre afin d'utiliser cette représentation lors de l'étape de vérification. En effet, chaque fonctionnalité est associée à un objet qui porte le même nom qu'elle.
		Le nom composé de l'objet *Conc_Mem_UpgradeCMU* met en évidence que le demandeur du *Upgrade* est un simple membre de rôle *Concerned Member (Conc_Mem)*. Le rôle *ConcMem* apparaît aussi dans la colonne 1, avec l'étiquette *ConcMem_LeaveCML*. Cet exemple montre la façon d'instancier les différentes actions par rôle d'utilisateur.

Tableau 5.9. Objets et diagrammes de comportement pour l'Upgrade dans la couche Application

Dans la couche GMM nous avons implanté les membres *type* en charge d'exécuter toutes les activités qui correspondent à la gestion de groupes (voir tableau 5.10). En utilisant la même approche que pour la couche application, nous avons implanté une classe et instancié un objet pour chaque rôle/fonction qu'un utilisateur pouvait prendre. Dans la couche GMM, un utilisateur de la couche GMM doit être synchronisé avec tous les objets représentant ses propres rôles instanciés dans la couche Application.

Nous avons identifié, d'un côté un membre type appelé *ConcernedMember* qui correspond à un membre sans responsabilité pouvant être appelé par n'importe lequel de ses rôles/fonctionnalités du niveau application. Le *join,* l'upgrade et le *leave* sont des exemples des fonctionnalités qu'il peut exécuter. Un second membre *type,* appelé *ChiefAdministrator* a été implanté pour prendre toutes les responsabilités d'un chef/administrateur. Il sera appelé lorsqu'une certaine fonctionnalité de niveau application a besoin d'une autorisation ou d'un ordre.

Le tableau 5.10 montre les objets et le comportement des deux membres mentionnés pendant qu'ils participent à l'exécution de l'*upgrade* dans la couche GMM.

La fig. 5.10 montre un objet qui représente le *ConcernedMember (M1G1S1C1 : ConcMem_GMM)* dans la couche de gestion de groupes GMM. Les lignes qui arrivent sur cet objet matérialisent les relations vers les objets provenant de la couche Application.

Les étiquettes (Upgrade, Join, Leave et Reinst) de la fig. 15.11 matérialisent les choix possibles qui correspondent à l'exécution des fonctionnalités de gestion de groupes.

Par souci de clarté, dans la fig. 5.11, nous expliquons le comportement du *ConcernedMember_GMM* pendant le déroulement de l'*Upgrade*. Comme le diagramme d'activité de ce membre est le plus complexe, nous allons nous focaliser sur l'explication de la fonctionnalité *Upgrade.* L'ovale étiqueté **ConcMem1** regroupe les activités que le *ConcernedMember* réalise pour construire et envoyer au *ChiefAdministrateur* le PDU de demande *d'Upgrade*. L'ovale **ConcMem2** représente l'indication que la clé du nouveau groupe a été donnée au *ConcernedMember*, et donc l'acceptation de la montée. L'obtention de la nouvelle clé se déroule dans la couche GCKM. L'ovale **ConcMem3** décrit la non acceptation de la montée. Un PDU de refus est reçu dans ce cas là.

La fig. 5.12 montre un objet qui représente le *ChiefAdministrateur (C1 :ChiefAdministrator_GMM)* dans la couche de gestion de groupes GMM. Dans la fig. 5.13, nous expliquons son comportement pendant le déroulement de l'*Upgrade*. Comme pour le *ConcernedMember,* nous allons juste expliquer la partie du diagramme qui concerne la fonctionnalité *Upgrade*.

L'ovale étiqueté **ChiefAdm1** représente l'arrivée d'une demande (PDU venant d'un *ConcernedMember*) de changement de connexion pour une certaine classe (soit une connexion, soit une déconnexion), suivie de l'évaluation de cette demande. Dans le cas d'un *upgrade*, le PDU contient une demande d'entrée dans la nouvelle classe. L'ovale **ChiefAdm2** contient les instructions vers la couche GCKM pour transmettre vers le membre la clé de sa nouvelle classe, ce qui correspond à accepter la montée du *ConcernedMember*. La transmission de la nouvelle clé se déroule dans la couche GCKM. L'ovale **ConcMem3** décrit la non-acceptation de la montée. Un PDU de refus est envoyé au demandeur.

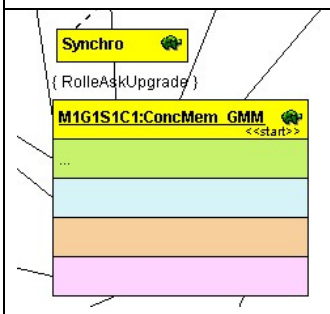
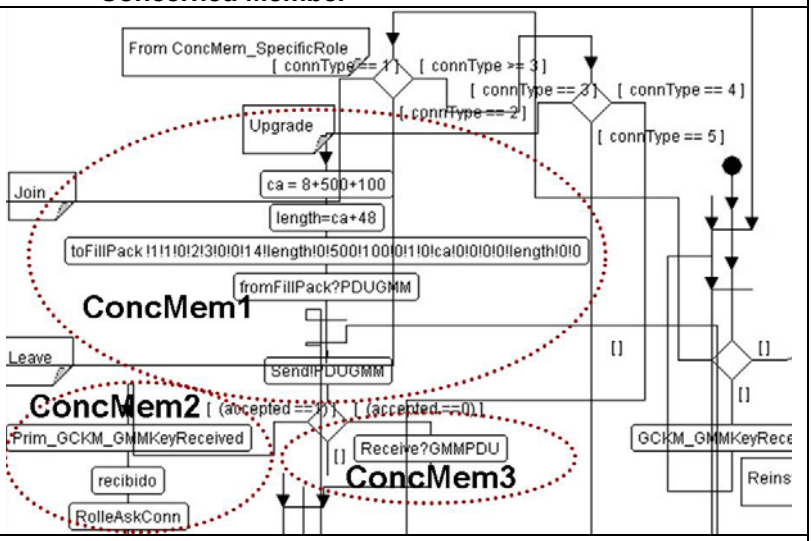
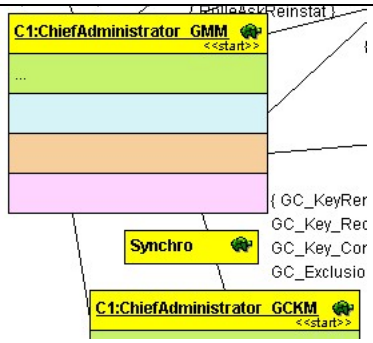
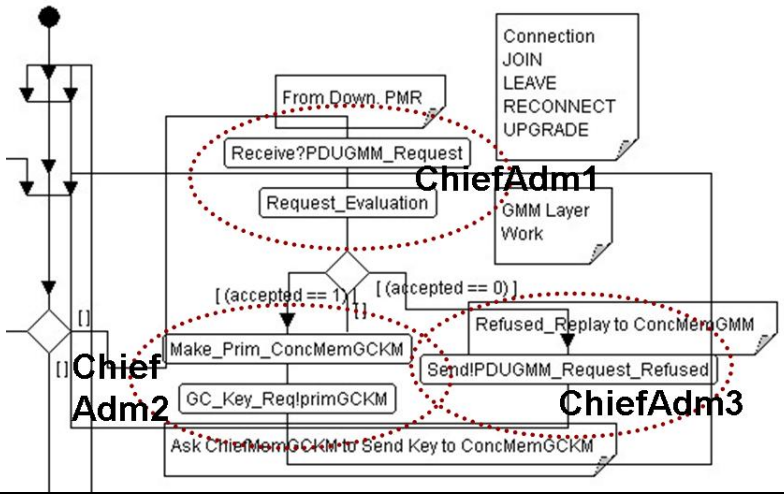
<table>
<tr><th colspan="2">Group Membership Management. GMM</th></tr>
<tr><td colspan="2">Concerned Member</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Fig. 5.10. Classe Concerned Member_GMM</td><td>Fig. 5.11. Diagramme des Activités. Concerned Member_GMM</td></tr>
<tr><td colspan="2">Chief-Administrateur</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Fig. 5.12. Classe Chief Administrator_GMM</td><td>Fig. 5.13. Diagramme des Activités. Chief Administrator _GMM</td></tr>
</table>

Tableau 5.10. Objets et diagrammes de comportement pour l'Upgrade dans la couche GMM

Le tableau 5.11 montre les objets et le comportement des membres participants dans la couche GCKM pendant l'exécution de l'*upgrade*.

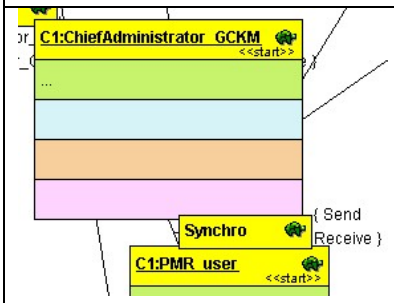
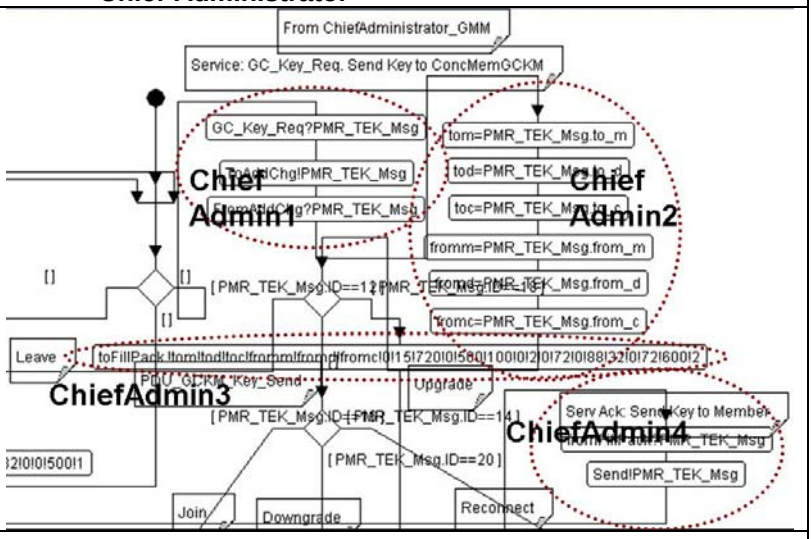
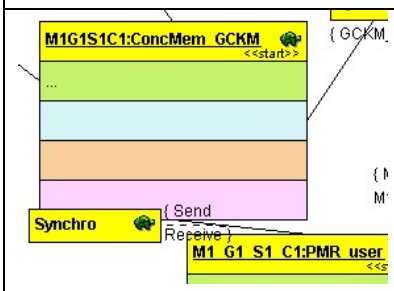
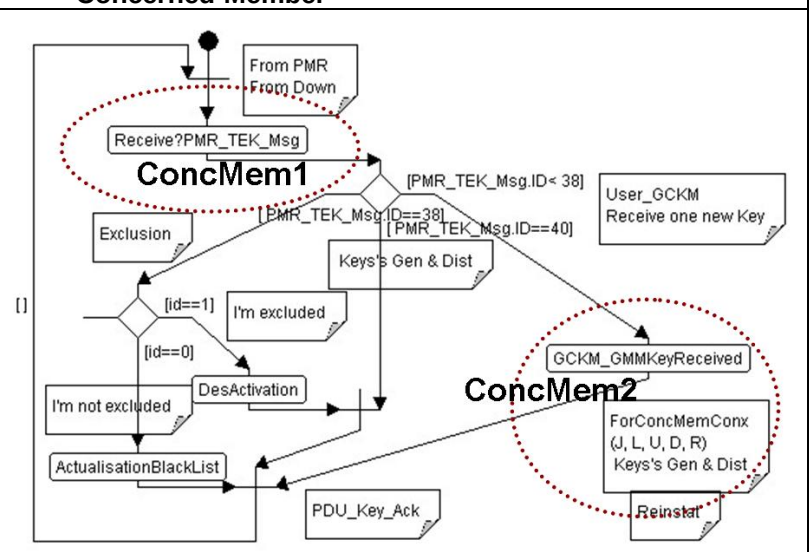
<table>
<tr><th colspan="2">Group Communication Key Management. GCKM</th></tr>
<tr><td colspan="2">Chief-Administrator</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Fig. 5.14. Classe ChiefAdministrator_GCKM</td><td>Fig. 5.15. Diagramme des Activités. ChiefAdministrator_GCKM</td></tr>
<tr><td colspan="2">Concerned-Member</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Fig. 5.16. Classe ConcernedMember_GCKM</td><td>Fig. 5.17. Diagramme des Activités. ConcernedMember_GCKM</td></tr>
</table>

Tableau 5.11. Objets et diagrammes de comportement pour l'Upgrade dans la couche GCKM

La fig. 5.14 montre un objet qui représente le *ChiefAdministrator (C1 : ChiefAdministrator_GCKM)* dans la couche de gestion de clés GCKM. Cet objet est normalement appelé par l'objet de la couche GMM de la même personne. Il implante une partie des services de gestion de clés offerts par la couche GCKM.

Le diagramme de la fig. 5.15 contient une partie du comportement du *ChiefAdministrator.* L'ovale étiqueté **ChiefAdm1** représente la demande, au *ChiefAdministrator_GCKM*, de distribution de la nouvelle clé vers le *ConcernedMember_GCKM* qui a demandé la montée. L'ovale **ChiefAdm2**

représente la préparation du message qui contient la clé. L'ovale **ConcMem3** montre le contenu d'un paquet de contrôle pour envoyer la clé de session demandée pendant le *upgrade*. L'ovale **ConcMem4** montre l'envoi de la clé vers le *Concenrnedmeber*.

Les étiquettes de la partie inférieure de la figure (*Leave, Join, Downgrade, Reconnect)* mettent en évidence les différents messages qui seront envoyés selon la fonctionnalité qui initie l'envoi.

La fig. 5.16 montre l'objet du *ConcernedMember (M1G1S1C1 : ConcernedMember_GCKM)* pour la couche de gestion de clés GCKM.

Son comportement, décrit dans la fig. 5.17, montre qu'une clé peut arriver en fonction de différentes possibilités. Dans notre cas, dans l'ovale **ConcMem1**, si la clé arrive grâce à une demande qui provient d'un *update*, on transmet une annonce de l'arrivée de cette clé vers la couche GMM du même utilisateur. Les actions associées nécessaires seront prises dans les couches supérieures.

5.4.2. Présentation des résultats

Etablie par le partenaire EADS dans le cahier de charges, nous nous sommes focalisés sur la vérification des exigences du tableau 5.12 [LIV 1.3] [LIV 1.4].

La modélisation TURTLE prend en compte l'ensemble des huit fonctionnalités de sécurité et de gestion intra-groupes suivantes : Key Generation and Distribution, Join, Leave, Upgrade, Downgrade, Reconnection, Reinstat, Exclusion. Les fonctions inter-groupes modélisées sont Merging et Splitting.

Les fonctionnalités Join, Leave, Upgrade, DownGrade, Reinstat et Reconnection impliquent une communication entre deux membres. Les autres fonctionnalités, Generation, Merging, Splitting et Exclude, impliquent une communication vers tous les membres du groupe, voire vers tous les membres de tous les groupes.

5.4.2.1 Exigences temporelles à vérifier

Le tableau 5.12 classe par fonctionnalité les exigences temporelles que le système doit respecter. Chaque ligne du tableau décrit une exigence temporelle qui doit être vérifiée lors d'un changement dans la composition des groupes. Une croix entre une ligne et une colonne indique à quelle fonctionnalité cette exigence s'applique. Les exigences du tableau 5.12 sont classées en deux groupes. Dans le premier groupe, dont les exigences sont étiquetées par *1i*, nous rassemblons les exigences qui font référence à la dynamique des groupes, soit pour limiter le temps des entrés des membres dans des groupes (1a et 1ab), soit pour limiter les temps de changements dans la structure du groupe existant (1ab et 1b). Dans le deuxième groupe, dont les exigences sont étiquetées par *2j*, nous rassemblons les exigences qui se réfèrent à la sécurité du protocole, soit pour limiter le temps de détection des violations des messages échangés (2a), soit pour limiter le temps d'exclusion des membres détectés comme des intrus (2b).

	Exigence \ Fonctionnalités	Gen&Dist	Join	Leave	Upgrade	Downgrade	Reinstal	Exclude	Recomm	Merging	Spliting
1a	Temps d'établissement d'une communication chiffrée < 350 ms	X									
1a	Temps moyen d'une entrée tardive dans une communication chiffrée < 1s		X				X		X		
1a	Temps maximum d'une entrée tardive dans une communication chiffrée < 2s		X				X		X		
1ab	Temps d'accès au groupe Multimédia < 350 ms		X		X	X	X		X		
1ab	Temps d'accès au groupe Messagerie < 1 mn		X		X	X	X		X		
1b	Temps de fusion < 1 s									X	
1b	Temps de partition < 1mn										X
2a	Temps de détection d'une violation d'intégrité < 10 s	X	X	X	X	X	X	X	X		
2a	Temps de détection de rejeu < 10 s	X	X	X	X	X	X	X	X		
2b	Temps maximum pour exclure un membre par un administrateur < 1 h							X			
2b	Temps minimum pour exclure un membre par un autre membre < 1 mn							X			

Tableau 5.12. Exigences à vérifier dans le protocole SAFECAST

La signification des exigences temporelles à vérifier dans le tableau 5.12 est la suivante :

- Temps d'établissement d'une communication chiffrée : il s'agit du temps qu'un groupe d'utilisateurs prend pour établir une clé de session dans le groupe.
- Temps moyen et maximum d'une entrée tardive dans une communication chiffrée : il s'agit des temps correspondant à l'entrée d'un membre qui arrive après que la session de communication ait déjà commencé. On trouve notamment le cas d'un *join* mais aussi le cas d'une *reconnexion* et d'une *réinstallation*.
- Temps d'accès au groupe multimédia ou messagerie : ceci représente les temps qu'un utilisateur prend soit pour entrer la première fois dans la session, soit pour être accepté lors d'une demande de montée ou de descente dans une classe. Pour ces deux cas, étant donné que le processus d'accès dans une classe est indépendant du type de données à échanger, un même scénario de vérification sera utilisé.
- Temps de fusion : ce temps représente la durée nécessaire pour qu'un ensemble de groupes s'organise dans un nouveau groupe fusionné.
- Temps de séparation ou partition : ce temps représente la durée nécessaire pour qu'un groupe fusionné revienne à sa configuration originale avant fusion.
- Temps de détection de rejeu ou d'une violation d'intégrité : ceci représente le temps que le protocole prend pour détecter qu'un message déjà envoyé a été réutilisé malicieusement ou modifié par une attaque contre l'intégrité du message.
- Temps maximum pour exclure un membre par un administrateur : il s'agit du temps maximal qu'un administrateur prend pour exclure un membre malicieux.
- Temps minimum pour exclure un membre par un utilisateur : il s'agit du temps qu'un utilisateur prend pour exclure un membre qui a été découvert dans la réalisation d'une action malicieuse.

Les délais à vérifier sont exprimés dans les exigences de la deuxième colonne de ce tableau. Ils sont assez variables. Certains se situent dans l'échelle de la

milliseconde, d'autres utilisent des délais proches de la seconde, les derniers, les plus lâches, mettent en œuvre des exigences de l'ordre de l'heure. Le premier groupe d'exigences (exigences 1i) contient les contraintes temporelles les plus dures, puisque les durées des quatre premières exigences vont de quelques millisecondes à un maximum de deux secondes. Dans le deuxième groupe nous observons que les exigences temporelles ne sont pas trop restrictives. Leurs valeurs sont comprises entre 10 secondes et 1 heure.

Lors de la modélisation, deux types de réseau radio sont étudiés : le réseau bas débit pour lequel le temps de transmission est de 6 kbps et le réseau moyen débit avec un temps de transmission de 100 kbps. Le modèle TURTLE du système a été enrichi d'un chronomètre qui permet de comptabiliser les durées en obtenant ainsi le temps consommé par chaque activité dans le contexte d'un scénario global. Les résultats ci dessous suivent la technique d'analyse temporelle proposée dans la section 5.3.4.1.

5.4.2.2 Vérification pour un réseau bas débit

La fig. 5.18 décrit les temps que nous avons relevés pendant l'exécution d'un *upgrade* pour un réseau bas débit. Les terminaisons **Min** et **Max** indiquent le temps minimal de transmission, si le membre qui monte est à côté du chef et le temps maximal, si le membre qui monte est à la portée maximale permise (3km).

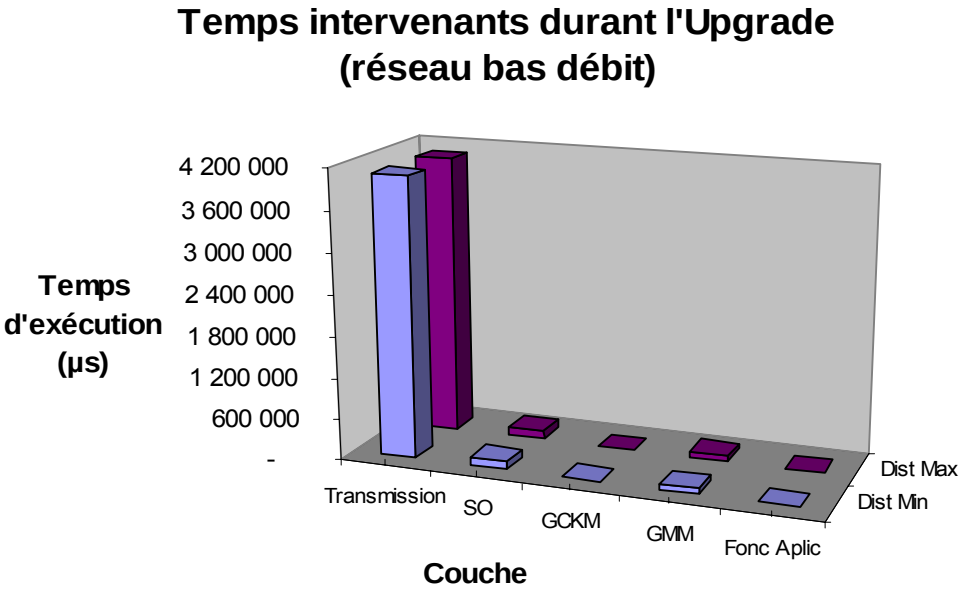


Fig. 5.18. Temps intervenants pendant l'exécution d'un Upgrade sur un réseau bas débit

Sur cette figure, nous voyons que le temps de transmission des messages, c.à.d. **« Transmission »** des courbes correspondant à **« Bas débit Min »** et à **« Bas débit Max »,** n'est pas du tout dans la même gamme de valeurs que les autres temps intervenants (**« SO », « GCKM », « GMM » et « Fonctionnalité applicative »)**. Le temps « Transmission » pour « Bas_débit_Min » est major que 2 400 000 µs (4 071 588 µs), pendant que le temps « SO » est égal à 120 270 µs et que le temps « GMM » est égal à 80 300 µs. Nous trouvons que le problème ne réside pas dans le protocole sous étude, il se trouve plutôt dans le réseau physique qui le supporte. Ce réseau apparaît comme trop lent vis-à-vis des contraintes à respecter pour les fonctionnalités.

Dans le cas du bas débit, seules les exigences les plus lâches sont respectées (de l'ordre d'une heure ou d'une minute, voire quelques secondes). Les résultats de vérification temporelle obtenus sont montrés dans le tableau 5.13.

Les délais à vérifier sont repris dans la deuxième colonne du tableau. La ligne « Fonctionnalité » montre les fonctionnalités qui interviennent à chaque exigence. La ligne « Durée obtenue » montre les temps donnés par l'environnement TURTLE par fonctionnalité du protocole, exprimé en ms. La valeur **« V »** indique que la propriété s'applique à la fonctionnalité étudiée, et que son exigence temporelle est satisfaite. De la même façon, la valeur **« F »** indique que la propriété s'applique à la fonctionnalité étudiée, mais que son exigence temporelle n'est pas satisfaite. Une case vide indique qu'une propriété ne s'applique pas à une fonctionnalité.

	Fonctionnalité	Gen Clé	Join	Leave	Upgrade	Downgrade	Reinstal	Exclude	Reconn	Demandé Merging	Ordonné Merging	Split
	Durée obtenue (ms)	2 189	4 008	3 284	4 272	6 198	6 198	2 677	4 272	6 949	8 632	1 643
	Exigence											
1a	Temps d'établissement d'une communication chiffrée < 350 ms	F										
1a	Temps moyen d'une entrée tardive dans une communication chiffrée < 1 s		F				F		F			
1a	Temps maximum d'une entrée tardive dans une communication chiffrée < 2 s		F				F		F			
1ab	Temps d'accès au groupe Multimédia < 350 ms		F		F	F	F		F			
1ab	Temps d'accès au groupe Messagerie < 1 mn		V		V	V	V		V			
1b	Temps de fusion < 1 s									F	F	
1b	Temps de partition < 1 mn											V
2a	Temps de détection d'une violation d'intégrité < 10 s	V	V	V	V	V	V	V	V			
2a	Temps de détection de rejeu < 10 s	V	V	V	V	V	V	V	V			
2b	Temps maximum pour exclure un membre par un administrateur < 1 h							V				
2b	Temps minimum pour exclure un membre par un autre membre < 1 mn							V				

Tableau 5.13. Respect d'exigences pour un réseau bas débit

Pour un ensemble important d'exigences où la borne temporelle à ne pas dépasser est assez contraignante (allant de 350 ms jusqu'à 2 s), le protocole SAFECAST sur un réseau bas débit ne remplit pas les besoins en termes de performance. Les cas de la *génération et* de la *distribution de la clé* de session : 2.19 s, du *downgrade* d'un membre : 6.2 s et du *merge* des deux groupes : 6.95 s pour une fusion sur ordre, ou 8.6 s pour une fusion suite à une demande, sont des exemples de fonctionnalités qui sont très loin de remplir leurs exigences.

Plus un nombre important de messages est échangé dans une fonctionnalité, plus elle va se trouver loin de satisfaire les exigences établies. Par exemple, le temps moyen d'une entrée tardive dans une communication chiffrée doit être au maximum d'une seconde. Dans le cas du *join*, qui nécessite avec deux étapes d'échange des messages, cette entrée tardive prend *4 s*. Dans le cas du *reinstal*, qui nécessite trois étapes d'échange de messages, elle prend *6.2 s* (avec trois étapes d'échange de messages). Aucun des deux cas ne respecte la limite temporelle établie.

Finalement, seules les exigences qui sont de l'ordre d'une dizaine de secondes, d'une minute et d'une heure sont respectées.

5.4.2.3 Vérification pour un réseau moyen débit

La facilité de réutilisation des modèles TURTLE a permis de passer très facilement du système SAFECAST pour un réseau bas débit à un réseau moyen débit. Nous avons juste changé les caractéristiques temporelles du médium, pour procéder à une nouvelle étude du protocole.

Dans le cas du moyen débit, l'analyse temporelle du modèle TURTLE s'avère plus intéressante puisque les temps intervenants pour la transmission sont très inférieurs aux exigences des fonctionnalités du protocole.

La fig. 5.19 décrit les temps que nous avons relevés pendant l'exécution d'un *upgrade* pour un réseau moyen débit.

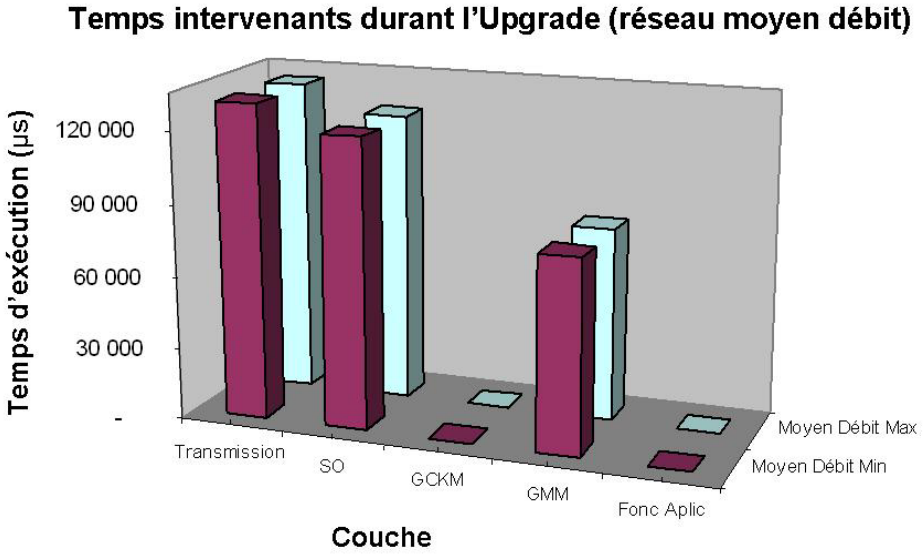


Fig. 5.19. Temps intervenants pendant l'exécution d'un Upgrade sur un réseau moyen débit

Sur cette figure, nous voyons que le temps de transmission des messages, c.à.d. **« Transmission »** des courbes correspondantes à **« Moyen débit Min »** et **« Moyen débit Max »,** est dans la même gamme de valeurs que les autres temps intervenants dans la fonctionnalité Upgrade. Le temps « **Transmission** » pour « Moyen_débit_Min » est égal à 130 312 µs, le « Temps SO » est égal à 120 270 et le « Temps GMM » est égal à 80 300 µs. Ce réseau est suffisamment rapide pour respecter les contraintes des fonctionnalités.

Comme les temps de transmission sont compatibles avec la durée des services des niveaux supérieurs, nous poursuivons notre analyse par une comparaison plus fine et plus détaillée des temps qui interviennent.

La fig. 5.20 montre le temps que chaque couche prend dans le temps total d'exécution de certaines fonctionnalités du protocole SAFECAST. Ces fonctionnalités ont été choisies pour leur représentativité. L'étude que nous avons faite nous donne des différences de temps négligeables entre le moyen débit min et le moyen débit max. Pour des questions de lisibilité, nous ne présentons que les temps obtenus pour le moyen débit max sur la figure 5.20.

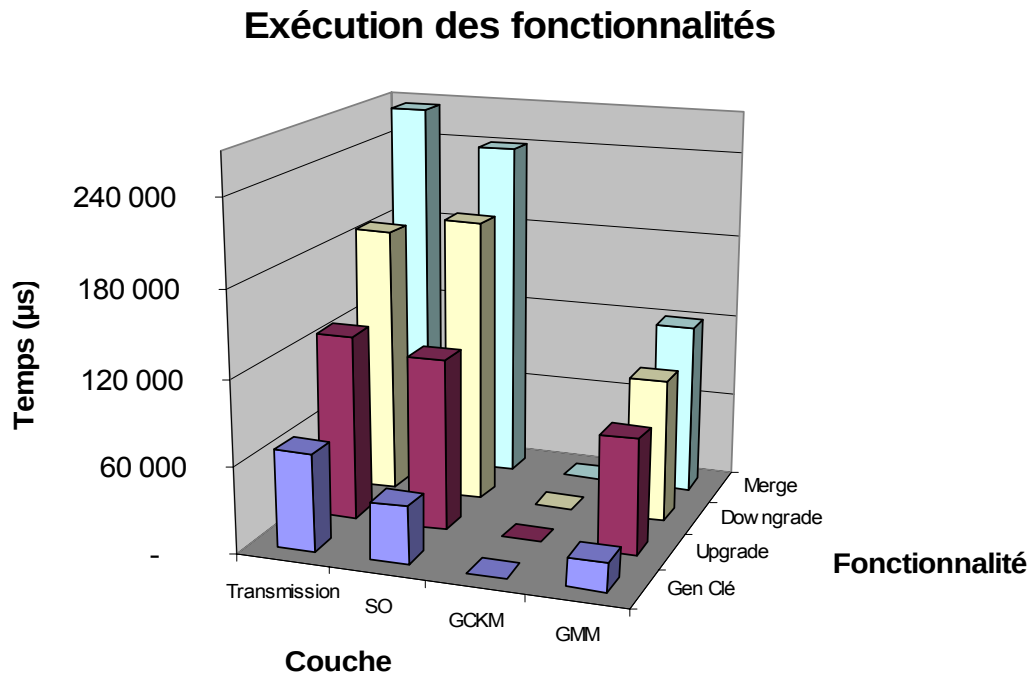


Fig. 5.20. Comportement temporel des fonctionnalités du protocole SAFECAST dans un réseau moyen débit

Le schéma de la fig. 5.20 montre quatre fonctionnalités du protocole SAFECAST : Génération de clés, Downgrade, Upgrade et Merge. Chacune des couches qui intervient est représentée en abscisse : Temps de transmission ; SO, GCKM, GMM. L'ordonnée donne le temps nécessaire à l'exécution des activités de chaque couche.

La fonction pour générer une clé de session et la distribuer « *Gen Clé* » dans un groupe semble être la plus simple. Elle est composée d'un seul message envoyé vers tous les utilisateurs d'un groupe pour leur distribuer la clé de session. La montée d'un membre dans un groupe *« Upgrade »* (courbe jaune) est plus compliquée. Cette monté implique l'échange de deux messages (un pour demander à monter, un autre pour transmettre une réponse). La descente d'un membre *« Downgrade »* représente un exemple de fonctionnalité qui se réalise au moyen de trois messages. Le premier correspond à l'indication qui est émise par le chef/administrateur, et qui est reçue par le membre, pour descendre de niveau hiérarchique. Les deux autres messages correspondent à l'entrée du membre dans sa nouvelle classe et à la réponse du chef responsable de cette classe.

Finalement, le schéma de la fig. 5.20 nous montre l'exemple d'une fonctionnalité dont le comportement est plus complexe que celui des fonctionnalités précédentes : Le *« Merge »*. Sa réalisation consomme plus de temps, car quatre échanges de messages sont nécessaires. Le premier correspond à l'ordre (multicast) envoyé vers tous les chefs qui fusionnent. Le deuxième message procède à l'envoi (multicast) par le nouveau chef de la nouvelle clé du groupe fusionné vers tous les chefs participants. L'envoi de la nouvelle clé de session à l'intérieur de chaque groupe (plusieurs envois multicast en parallèle) correspond au troisième ensemble d'échanges. Le dernier groupe

d'échanges correspond au retour, vers l'administrateur qui a demandé la fusion, des acquittements de fusion de la part des chefs des groupes.

De façon plus précise, le modèle réalisé en TURTLE nous a fourni les durées suivantes : la *« Generation et distribution »* de la clé de session consomme un temps de 128 707 µs. Le *« Upgrade »* d'un membre dans un groupe prend 331 217 µs. Le *« Downgrade »*, réalisée au moyen des trois messages, consomme 490 170 µs. Le *« Merge »* consomme 626 740 µs pour s'effectuer, car quatre échanges de messages sont nécessaires. A partir de ces résultats, nous pouvons observer que, une augmentation dans la complexité des fonctionnalités du système cause une augmentation prédictible sur les temps obtenus lors de l'exécution de cette fonctionnalité. Les opérations destinées au transport et à la sécurisation des échanges ainsi que celles destinées à la gestion des groupes augmenteront de manière proportionnelle à la complexité de la fonctionnalité considérée.

Finalement, en faisant une deuxième analyse des courbes de la fig. 5.20, nous pouvons aussi préciser le pourcentage de temps que chaque couche prend pendant l'exécution des fonctionnalités du système SAFECAST. Ainsi, le temps *« Transmission »* de chaque courbe consomme environ 40% du temps total d'exécution. Le temps *« SO »*, c.à.d. celui nécessaire aux opérations de sécurisation des échanges, consomme aussi une quantité importante du temps total, de 30% à 40%. L'addition de ces deux couches, qui forment la transmission sécurisée, représente environ 80% du temps total d'exécution. C'est pour cette raison que de nombreux travaux de recherche ont été ciblés sur l'étude de la sécurisation des échanges dans des protocoles de gestion des groupes, en vue de minimiser le coût de ces échanges.

Le temps de génération des clés GCKM, est négligeable dans SAFECAST.

Le temps GMM, c.à.d. les opérations de gestion des groupes, consomme de 15% à 20% du total, ce qui est assez faible.

Cette analyse des temps des actions des fonctionnalités nous permet de prendre conscience de l'importance de chaque couche du protocole.

Les travaux qui étudient les performances des groupes de communication sécurisés [AMI 05], [BAN 02], [WAN 05], ont mis en avant que les opérations de gestion des clés et de gestion de la dynamique des groupes sont coûteuses en temps et en consommation des ressources. Ces travaux considèrent ces opérations de gestion de façon globale, c.à.d. qu'ils ne font pas la différence entre les temps pour les transmissions sécurisés et les temps pour les traitements locaux nécessaires aux algorithmes de gestion. Nos conclusions vont dans le même sens que ces travaux. Cependant, grâce à notre analyse plus fine par couche, nous avons pu montrer que, dans le cas de notre système SAFECAST pour un réseau moyen débit, la transmission sécurisée inhérente à cette gestion est l'élément qui consomme le plus de temps : 80% du temps global. Ce pourcentage est élevé lorsqu'on le compare aux 20% restants qui ne concernent que les traitements locaux.

Le tableau 5.14 synthétise les résultats de l'analyse temporelle du protocole SAFECAST confronté aux exigences établies pour un réseau moyen débit.

	Fonctionnalité	Gen Clé	Join	Leave	Upgrade	Downgrade	Reinstal	Exclude	Reconn	Merging Demandé	Merging Ordonné	Split
	Exigence \ Temps d'exécution (ms)	129	303	222	331	490	490	145	331	496	627	111
1a	Temps d'établissement d'une communication chiffrée < 350 ms	V										
1a	Temps moyen d'une entrée tardive dans une communication chiffrée < 1 s		V				V		V			
1a	Temps maximum d'une entrée tardive dans une communication chiffrée < 2 s		V				V		V			
1ab	Temps d'accès au groupe Multimédia < 350 ms		V		V	F	F		V			
1ab	Temps d'accès au groupe Messagerie < 1 mn		V		V	V	V		V			
1b	Temps de fusion < 1 s									V	V	
1b	Temps de partition < 1 mn											V
2a	Temps de détection d'une violation d'intégrité < 10 s	V	V	V	V	V	V	V	V			
2a	Temps de détection de rejeu < 10 s	V	V	V	V	V	V	V	V			
2b	Temps maximum pour exclure un membre par un administrateur < 1 h							V				
2b	Temps minimum pour exclure un membre par un autre membre < 1 mn							V				

Tableau 5.14. Résultats de la vérification temporelle du protocole SAFECAST

Dans le tableau 5.14, nous constatons que toutes les exigences temporelles sont satisfaites excepté le temps d'accès à un groupe multimédia.

Un analyse plus poussée nous laisse entrevoir que pour certaines exigences le protocole SAFECAST semble ne pas avoir des problèmes au niveau temporel, pour d'autres, la satisfaction des exigences se fait de justesse. Par exemple, l'exigence *« Temps de partition < 60 s »* nous demande une durée maximale de 60 s pour effectuer un *split* et nous avons obtenu une durée de *111 ms*. Nous avons donc une grande latitude pour ajouter des mécanismes pour améliorer cette fonctionnalité. Un autre exemple est l'exigence *« Temps de détection d'une violation d'intégrité < 10 s »,* pour laquelle nous avons un temps à respecter, dans le pire de cas, de 10 s. Nous avons obtenu un temps de maximum de 0.49 s. Nous avons donc 19 fois plus de temps à utiliser pour le renforcement des fonctionnalités nécessaires à cette exigence. Par contre l'exigence *« Temps de fusion < 1 s »* nous indique que le temps maximal d'exécution du *merge* doit être de *1 s*. Nous avons obtenu une durée de *0.627 s* dans le pire des cas (le *merge* provoqué sur ordre). Dans le cas de cette fonctionnalité, le temps restant laisse une latitude plus faible qui nécessitera une analyse fine des temps consommés par tout mécanisme que l'on souhaiterait ajouter.

La négociation entre la sécurité que le protocole garantit et le prix à payer en termes de satisfaction temporelle est le centre de notre étude. En effet, il nous intéresse de déterminer les fonctionnalités dont la sécurité peut être renforcée sans sacrifier la satisfaction des contraintes temporelles établies.

5.5. Conclusion

Ce chapitre a présenté l'approche que nous avons proposée pour l'étude et la validation des contraintes temporelles de mécanismes de gestion des groupes et de sécurité, au sein de systèmes de communication de groupes. Un cas d'utilisation comme celui de SAFECAST nous semble un cas complet dans le sens où il contient un degré de difficulté qui nous permet d'analyser différentes facettes d'un protocole de communication de groupes. Il présente, dans sa composition, différents niveaux hiérarchiques ainsi qu'une structure de groupes dynamique qui oblige au changement en ligne des ressources pour la protection de la communication.

Les réseaux radio PMR permettent, avec leurs différentes technologies, de tester le protocole sous différentes conditions de déploiement. Ainsi, nous pouvons montrer que ce qui semble une solution, pour certaines cas, ne peut pas être appliqué efficacement pour d'autres cas. Les mécanismes de sécurité proposés sont trop coûteux en temps dans le cas d'utilisation du réseau radio à bas débit, obligeant ainsi à faire des compromis sécurité versus temps pour obtenir un système réaliste.

Nous avons aussi montré que, l'application d'une méthodologie incrémentale de modélisation est très utile pour le développement de modèles dont la complexité exige de traiter à chaque étape une problématique différente. La facilité que l'environnement de l'outil TURTLE nous donne pour produire des modèles orientés objets nous a aidés pendant l'implémentation et l'étude d'un système complexe comme celui de SAFECAST. Le cadre architectural proposé dans le chapitre quatre a été ainsi instancié.

Pour l'étape de vérification, l'analyse par abstraction que nous avons menée a été guidée par l'utilisation de ports pro-vérification. Ces derniers, au travers de l'utilisation de techniques de minimisation fournies par CADP/Aldébaran, ont permis d'aboutir à des graphes d'accessibilité clairs et exploitables. Enfin la vérification formelle menée avec TURTLE a montré que toutes les exigences temporelles sauf une, sont satisfaites dans le cas d'un médium moyen débit [MOT 07]. Ces premiers résultats sont très encourageants. Ils confirment que le protocole SAFECAST, tel qu'il a été défini, répond bien à la majorité des contraintes et des besoins définis pour l'application PMR au tout début du projet.

Cependant, en faisant une distinction entre tous les types des activités qui composent le protocole (analyse par couches), nous pouvons maîtriser leur influence temporelle lors de l'exécution du protocole global. Ceci nous permet de proposer des mécanismes correctifs ou additionnels au protocole en ayant conscience du coût de ces modifications ou ajouts, d'après la couche modifiée et d'après l'impact que cette couche a sur la performance totale du système.

Dans un réseau moyen débit, les exigences temporelles établies sont respectées avec peu de marge de manœuvre pour les opérations de changements dans le groupe. Dans ce cas, il est nécessaire de regarder au plus juste quelles sont les modifications que nous pouvons apporter, et dans quel degré nous pouvons les utiliser, si nous souhaitons renforcer la sécurité de ces opérations.

Dans un deuxième temps, pour les opérations de détection des violations du système, nous avons une marge de manœuvre beaucoup plus importante. Nous pouvons dans ce cas utiliser des mécanismes beaucoup plus coûteux en temps si nous souhaitons renforcer la sécurité de ces opérations.

6 Conclusion et perspectives

Face à l'évolution des applications déployées sur des systèmes de communication de groupes, des plates-formes avancées comme Spread et Antigone émergent avec des solutions qui répondent bien aux derniers besoins en termes de communication : sécurisation, performance et adaptabilité. Nous avons cependant remarqué que leurs analyses et leurs propositions manquent d'un effort de standardisation et de documentation pour leur éventuelle reprise pour de futures utilisations dans le domaine de la communication de groupes.

D'un autre côté, nous avons trouvé une grande quantité de travaux [BAN 02], [KIM 04a], [HOR 02], [KU 03], [SHE 03], [WAL 98], [CAR 98], [NOU 98], [WON 98], tous orientés vers l'administration de la gestion de la sécurité de la communication des groupes (gestion de la clé de session) pour la protection de l'information échangée entre les membres d'une session. Des analyses mathématiques sur la charge des processus de gestion des clés ou bien des simulations, seront en général trouvées comme preuve de l'efficacité de la gestion de cette sécurité dans ces travaux. Ces évaluations n'attesteront pas de la validité des résultats pour des cas réels, car elles ne sont pas appliquées à des conditions et à des exigences imposées par une certaine application.

Nous sommes donc confrontés au besoin d'analyser et de documenter de tels types de systèmes, de considérer un champ bien défini de problématiques impliquées, puis de proposer une technique de modélisation et de vérification d'un protocole de communication, dont l'efficacité a été testée vis-à-vis d'exigences temporelles réelles établies lors de la phase d'analyse de l'application. Notre analyse UML contient un ensemble complet d'activités concernées par la communication des groupes. Notre architecture propose une démarche intéressante pour la standardisation des architectures de communication sécurisée de groupes.

Enfin, l'instanciation de cette analyse, l'implémentation de cette architecture, nous mène à la présentation de résultats en relation avec un cas d'application réel. La détection des points faibles du protocole proposé dans le cas du projet industriel SAFECAST montre l'applicabilité de nos travaux et ouvre un nouveau champ d'analyse.

6.1. Bilan des contributions

Nos contributions ont donc porté principalement sur les points suivants :

- Analyse des systèmes de communication de groupes en utilisant le profil UML 2.0. Définition des fonctionnalités intrinsèques de tels systèmes

Nous avons choisi de faire une analyse UML (Unified Modeling Language) des systèmes de Communication de Groupes, avec comme objectif de contribuer à une intégration la plus complète possible des fonctionnalités impliquées dans la communication de groupes. Cette documentation standard nous a facilité l'identification et l'organisation de ces fonctionnalités. Ainsi, en nous basant sur certains diagrammes de la notation UML 2.0, nous avons analysé et documenté les composants d'un protocole de communication sécurisé de groupes.

Pour effectuer la gestion des membres pour des groupes, nous définissons l'ensemble de fonctionnalités suivant : *join, leave, upgrade, downgrade, exclude, reconnect, reinstal, merge* et *split.* Une fonctionnalité commune aux fonctionnalités précédentes a été définie, *Connection_Actualisation*. Cette fonctionnalité prend le rôle du gestionnaire de la connexion. Ce gestionnaire est utilisé par toutes les fonctionnalités, avec une adaptation à chaque fonctionnalité, selon les valeurs des paramètres. Avec la proposition de ce module (*Connection_Actualisation)* nous simplifions de façon importante la tâche de modélisation du module de gestion du groupe.

Pour effectuer la gestion de la communication sécurisée, nous avons besoin de fonctions pour demander des renouvellements des clés mais aussi pour la distribution de messages lors de ces renouvellements. La liste des fonctions nécessaires à cette gestion est la suivante : distributeKey, distributeKeyHierarchical, distributeKeyPlane, renewHierarchicalKey, renewBasePeriodicKey, renewMergedGroupKey, renewPeriodicMergedGroup.

- Proposition d'une architecture de modélisation et de vérification de protocoles pour la communication sécurisée de groupes

L'architecture générique appelée SGCva (Secure Group Communication verification architecture) [MOT 06] [MOT 07] que nous avons proposée et développée dans ce mémoire peut être utilisée dans différents contextes pour des protocoles de communication de groupes. Sa définition s'appuie sur les fonctionnalités précédemment identifiées dans ce type de communications. Le modèle de référence OSI (Open System Interconnection) [TAN 96] a été la référence pour la structuration de notre architecture en couches. Les niveaux de déploiement de notre architecture s'inscrivent dans les couches Application, Session et Transport de ce modèle.

L'architecture proposée, en partant du niveau le plus bas et en terminant par le niveau le plus haut, comprend les services suivants : 1). Transport sécurisé de l'information échangée entre les membres d'une session (SO / Médium). 2). Gestion de la communication sécurisée des groupes (GCKM). 3). Gestion des groupes (GMM). 4). Application utilisatrice.

Pour chacune des couches, l'identification des primitives de service a été faite. Les PDUs échangés à l'intérieur d'une couche ont été aussi définis.

- Instanciation de notre architecture à travers une méthodologie incrémentale dans un projet industriel.

Le projet RNRT SAFECAST, dédié à l'étude, à la conception et à la validation

de fonctionnalités et de mécanismes garants de communications sécurisées à l'intérieur de groupes dynamiques [EAD 04], [SAFECAST], représente un cas réel et complexe de l'application d'un protocole sécurisé dans des communications de groupes. Dans [FON 07c], nous avons détaillé la complexité de ce système.

A travers l'application d'une méthodologie incrémentale [MOT 05], [MOT 06] de modélisation et de vérification, nous avons instancié notre architecture pour modéliser le système Safecast [LIV 2.5]. Nous avons apporté au partenaire EADS (leader du projet) et aux autres partenaires du projet : LORIA, UTC et ENST les résultats des preuves de satisfaction fonctionnelle et d'exigences temporelles établies lors de l'étape d'analyse du système [LIV 3.4].

Notre contribution aux bien livrables de projet se trouve dans [LIV 2.5], [LIV 4.1], [LIV 4.2], [LIV 4.3] et [LIV 4.4].

En coordination avec l'équipe LORIA, nous avons présenté une technique de vérification formelle pour des systèmes de groupes dite « croisée » [FON 07c]. Avispa [AVISPA] (équipe LORIA) a été utilisé pour vérifier SAFECAST dans le cadre sécuritaire. Turtle [TURTLE] a été utilisé pour s'assurer que les exigences temporelles sont bien respectées.

- Définition d'une nouvelle technique d'analyse de protocoles de groupes sécurisés

  Une contribution originale a été le fait de proposer une solution dont les qualités (fonctionnelles et de performances) auront été démontrées par des techniques de validation formelle. Nous avons traité en même temps et dans un même modèle de vérification des problématiques qui, à notre connaissance, avaient été auparavant adressées séparément, à savoir : (1) la sécurité avec la protection des messages et l'utilisation des certificats ; (2) la gestion des groupes qui varient dans leur composition au cours de l'évolution du système ; (3) la gestion de ces groupes, qui est régie selon une organisation hiérarchique qui vient complexifier les opérations de base sur un groupe ; (4) les contraintes temporelles ; (5) la combinaison étroite entre des exigences de sécurité et des contraintes temporelles.

Le travail développé dans cette thèse ouvre des possibilités pour l'analyse des performances de protocoles de communication des groupes à travers une étude temporelle de ses fonctionnalités.

Une étude des limites des propriétés de sécurité que le protocole peut respecter, se fait en termes de garanties temporelles établies durant la phase d'analyse du système.

Dans cette étude, notre contribution majeure a été de proposer que cette analyse temporelle soit en plus modulaire et en fonction des couches définies dans notre architecture. Avec ce type d'étude modulaire, nous pouvons préciser le pourcentage de temps que chaque couche prend pendant l'exécution des fonctionnalités d'un système de communication, tout en sachant que les activités impliquées dans un tel protocole sont nombreuses et complexes.

Par exemple, dans le cas du projet SAFECAST, un analyse guidée par notre proposition, nous laisse entrevoir que pour certaines exigences le protocole semble ne pas avoir des problèmes au niveau temporel, pour d'autres, la satisfaction des exigences se fait de justesse. La négociation entre la sécurité

que le protocole garantit et le prix à payer en termes de satisfaction temporelle est le centre de notre étude. Il nous intéresse de déterminer les fonctionnalités dont la sécurité peut être renforcée sans sacrifier la satisfaction des contraintes temporelles demandées.

Ceci nous permet de proposer des mécanismes correctifs ou additionnels au protocole en ayant conscience du coût de ces modifications ou ajouts, d'après la couche modifiée et d'après l'impact que cette couche a sur la performance totale du système.

6.2. Perspectives

6.2.1. De l'utilisation d'outils de vérification séparés pour une convergence des conceptions

La complexité des protocoles mis en œuvre pour gérer les échanges de messages au sein d'un système réparti justifie le recours à des techniques de modélisation supportées par des outils de simulation et de vérification formelles.

La description à un niveau « système » conduit aujourd'hui le plus souvent à utiliser des outils qui couvrent plusieurs « facettes » ou « points de vue » d'un même système. Le modèle résultant s'avère impossible à vérifier formellement dans son ensemble et la méthode retenue consiste à se focaliser sur chacune des facettes du système et à sélectionner des outils de vérification formelle adaptés à traiter chaque facette.

Un prolongement possible serait de proposer un langage de conception « commun » dans le sens où l'on produirait un seul modèle de Systèmes de Communications de Groupes Sécurisés (SCGS) d'où seraient ensuite extraites des sources destinés aux différents outils de vérification.

La faisabilité des résultats proposés pour chaque facette doit être validée après l'analyse de la complémentarité des solutions proposées pour chaque facette évaluée.

6.2.2. Approche d'utilisation conjointe d'outils de simulation et de vérification

Nous pourrions proposer une méthode pour la validation d'un système distribué qui profite des bénéfices des deux techniques afin d'offrir des résultats complémentaires.

Utiliser des techniques de simulation pour réaliser une approche préliminaire globale et à haut niveau d'un système de groupes donné, peut nous aider, dans une première instance, à avoir une vue générale du système testé. Utiliser ensuite des méthodes de vérification formelle pour l'étude détaillée des mécanismes composant le protocole étudié, nous permettra d'apporter des résultats plus précis sur des particularités de l'application.

Par exemple, des tests de simulation pourraient être envisagés pour évaluer le comportement global du protocole sous étude vis-à-vis de la dynamique d'un groupe pour un certain type d'application. Dans le monde de la simulation, l'utilisation de modèles de comportement concernant la composition des groupes peuvent être facilement intégrés. L'obtention des résultats issus de cette simulation du système peut guider pendant la phase de modélisation d'un système dans le monde de la vérification formelle. On utilise les résultats issus de la simulation pour identifier les points durs, dont on peut étudier plus finement

le comportement par des techniques de validation.

6.2.3. Application de notre méthode de vérification dans des systèmes autres que Safecast

Modéliser des systèmes avec des conditions différentes peut obliger à ce que la vérification s'oriente vers l'observation d'éléments non étudiés précédemment.

Les axes principaux que nous pourrions étudier sont les suivants :

1. Le réseau physique
2. L'algorithme de gestion de clés
3. Les techniques de sécurité utilisées, selon le niveau de sécurité requis

Un réseau radio qui s'appuie sur une infrastructure réseau de type haut débit peut permettre au système de communication de négliger certaines exigences de performance et de temps de réponse, alors que, pour un réseau avec des caractéristiques physiques minimales, ces exigences ne sont pas négligeables. Ceci donne comme conséquence le besoin de se focaliser sur des aspects autres que la composition d'un protocole de communication bien sécurisé. L'algorithme de gestion de clés est un thème pour de nombreuses études et propositions. Par contre, modéliser des améliorations de l'algorithme de Diffie-Hellman pour des clés de groupe et vérifier ces améliorations selon des approches de vérification formelle et temporelle, est une activité qui reste encore à faire. Avec des ressources physiques limitées, il est évident que nous ne pouvons pas faire une utilisation quelconque des techniques de sécurité pour la protection de la communication des groupes. Appliquer notre méthode à l'étude des nouvelles techniques dans la protection de la communication et dans l'authentification au sein des groupes fait aussi partie des travaux à réaliser.

6.2.4. Analyse du passage à échelle sous l'optique d'une méthode de vérification formelle

Dans le cadre du projet SAFECAST, la question a été contournée du fait de l'utilisation d'un médium PMR. En effet, le mécanisme de diffusion de PMR a été ainsi interprété : tous les récepteurs de la zone de couverture reçoivent en même temps un message donné et le nombre de récepteurs dans cette zone n'intervient pas. De fait, les modèles TURTLE présentés dans les rapports du projet sont construits sur des configurations « minimales » en nombre de membres ; ces configurations mettent principalement en exergue les différents types de membres (membres de base, chef, …) en identifiant les rôles qui participent à toute session de communication et qui apparaissent dans le SCGS SAFECAST.

Dans un SCGS avec un médium autre que PMR, on peut légitimement escompter que le nombre de groupes et de membres par groupes modélisés influent sur les résultats de vérification. Il s'agit là du problème de passage à l'échelle. Cette problématique est proposée comme un thème à étudier comme suite des résultats présentés dans le chapitre 5.

6.2.5. Perspectives liées au projet SAFECAST

La vérification du protocole proposé au sein du projet SAFECAST en présence de scenarii plus complexes est un travail qui reste à faire. Les sessions que nous

avons envisagées sont dynamiques en termes d'entrées et de sorties de membres et de changements de rôle au sein de la structure de session. Une extension de nos travaux pourrait affiner cette notion de dynamique selon deux directions :

1) Caractériser la fréquence des changements. Dans cette première gamme d'études, cette fréquence, approximativement constante, pourrait comprendre des sessions avec des changements peu fréquents, moyennement fréquents ou très fréquents.

2) Etudier des sessions avec des fréquences variables. Dans cette deuxième gamme d'études, les sessions pourraient voir leur fréquence de changement évoluer, par exemple passer de changements peu fréquents à très fréquents. Cela permettrait de caractériser plus finement le rapport entre les fréquences de changement maximum possibles avec les exigences temporelles des mécanismes de gestion des groupes et de sécurité utilisés. Si les changements dans la composition des groupes présentent une dynamique importante ajouter aux fonctionnalités des mécanismes de sécurité coûteux en termes de performances, peut sévèrement invalider la satisfaction des exigences temporelles établies. Mais, si le comportement de l'application ne nécessite pas beaucoup de mouvement des membres lorsqu'une session a été démarrée, augmenter la sécurité du protocole ne pose pas de problème, les exigences resteront satisfaites.

Pour la vérification des exigences temporelles, il faudrait en plus, envisager des mécanismes d'alerte ou correctifs déclenchés dynamiquement. Ces derniers cas, les plus complexes, font partie des extensions à long terme de nos travaux.

7 Références

[AMI 98] Y Amir, J Stanton. The Spread Wide Area Group Communication System. Technical Report CNDS-98-4. 1998.

[AMI 05] Y Amir, C Nita-Rotaru, J Stanton, G Tsudik. Secure Spread : An integrated Architecture for Secure Group Communication. IEEE Transactions on Dependable and Secure Computing, sept 2005.

[APR 04] L. Apvrille, J.-P. Courtiat, C. Lohr P. de Saqui-Sannes. TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit. IEEE Transactions on Software Engineering, Vol. 30, No. 7, pp.473-487, july 2004.

[APR 05] L. Apvrille, P. de Saqui-Sannes, F. Khendek. Synthèse d'une conception UML temps-réel à partir de diagrammes de séquences. Colloque Francophone sur l'ingénierie des protocols (CFIP '05), Bordeaux, France, mars 2005.

[ARM 05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. H´eam, J. Mantovani, S. Modersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Vigano, L. Vigneron, K. Etessami, S. Rajamani. The AVISPA Tool for the automated validation of internet security protocols and applications. 17th International Conference on Computer Aided Verification, CAV'2005, volume 3576 of Lecture Notes in Computer Science, pages 281–285, Edinburgh, Scotland, 2005.

[ASO 00] N. Asokan, P. Ginzboorg. Key Agreement in ad-hoc networks. Computer Communications. pp. 1627-1637. 2000.

[AVISPA] Project Avispa. Page Web http://www.avispa-project.org/

[BAG 06] W. Bagga, R. Molva. Collusion-Free Policy-Based Encryption. ISC 2006, LNCS 4176, pp. 233–245, Berlin 2006.

[BAN 02] S. Banerjee, B. Bhattacharjee. Scalable Secure Group Communication over IP Multicast. IEEE Journal on Selected Areas in Communications, 20(8):1151-1527, 2002.

[BIR 90] K. Birman, R. Cooper. The ISIS project: real experience with a fault tolerant programming system. Proceedings of the 4th workshop on ACM SIGOPS European workshop. pp. 1-5. Italy 1990.

[BIR 97] K. Birman, W. Vogels, K. Guo, M. Hayden, T. Hickey, R. Friedman, S. Maffeis, R. van Renesse, A. Vaysburd. Moving the Ensemble Groupware System to Windows NT and Wolfpack. Proc. of USENIX Windows NT Workshop, Seattle, WA. August 1997.

[BIR 00] K. Birman, R. Constable, M. Hayden, C. Kreitz, O. Rodeh, R. van Renesse, W. Vogels. The Horus and Ensemble Projects: Accomplishments and Limitations. Proc. of the DARPA Information Survivability Conference & Exposition (DISCEX '00), South Carolina, January 2000.

[BLU 93] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung. Perfectly-secure key distribution for dynamic conferences. CRYPTO ’92. LNCS. Vol. 740, Springer-Verlag. pp. 471–486. Berlin, 1993.

[BUC 04] C. M. Buchholtz, C. Montangero, L. Perrone, and S. Semprini. Priami and P. Quaglia. For-LySa: UML for authentication analysis. Proceedings of the second workshop on Global Computing, LNCS 3267, pp. 92–105, 2004.

[BUR 94] M. Burmester, Y. Desmedt. A secure and Efficient Conference Key Distribution System. Proc. conf. advances in Cryptology. EUROCRYPT 94. May 1994.

[CACTUS] Project Cactus. Page Web. http://www.cs.arizona.edu/projects/cactus/

[CADP] CADP. Page Web. http://www.inrialpes.fr/vasy/cadp/

[CAN 99a] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas. Multicast security : a taxonomy and somme efficient constructions. Proceedings of INFOCOM ’99 : Conference on Computer Communications. 1999, pp. 708-716.

[CAR 98] G. Caronni, K. Waldvogel, D. Sun, B. Plattner. Efficient security for large and dynamic multicast groups. Proceedings of the Seventh IEEE International Workshop on Enabling Technologies : Infrastructure for Collaborative Entreprises (WETICE ’98). 1998, pp. 376-383

[CHI 06] Chik How Tan; Teo, J.C.M., Energy-efficient ID-based group key agreement protocols for wireless networks. Parallel and Distributed Processing Symposium 2006. IPDPS 2006. 20th International.

[COU 91] J.P Courtiat, M. Diaz, V.B Mazzola, P. de Saqui-Sannes, « Description formelle de protocole et de services OSI en Estelle et Estelle* - Expérience et Méthodologie ». CFIP’91 Ingénierie des protocoles. Septembre 1991 France.

[COU 00] J.P. Courtiat, C.A.S. Santos, C. Lohr., B. Outtaj. Experience with RT-LOTOS, a Temporal Extension of the LOTOS Formal Description Technique. Computer Communications, Vol. 23, No. 12, p. 1104-1123, 2000.

[DIF 76] W.Diffie, M.E.Hellman. New Directions in Cryptography. IEEE Trans. Inform. Theory, IT-22, 6, pp.644-654. 1976.

[DOL 96] D. Dolev and D. Malki, The Transis Approach to High Availability Cluster Communication. Communications of ACM 39, 4, April 1996.

[DON 99a] L.R. Dondeti. Efficient private group communication over public networks. Phd Dissertation, CSE UNL.

[DON 99b] L.R. Dondeti, S. Mukherjee, A. Samal. A dual encryption protocol for scalable secure multicasting. Fourth IEEE Symposium on Computers and Communications, pp. 2-8. 1999.

[DON 00] L.R. Dondeti, S. Mukherjee, A. Samal. DISEC. A distributed framework for scalable secure many-to-many communication. Proceedings of fifth IEEE Symposium on Computers and Communications (ISCC 2000), 2000, pp. 693-698.

[DU 99] F. Du, L.M. Ni, A. H. Esfahanian. Towards solving multicast key management problem. Eight International Conference on Publication Communications and Networks, 1999. pp. 232-236. 1999.

[EAD 04] EADS. Projet National SAFECAST. L1.1 Dossier de Définition de l'application de communication de groupe, sept. 2004.

[EADS] EADS. Page Web. http://www.eads.com/1024/fr/Homepage1024.html

[ELL 03] C. Ellison, S. Dohrmann. Public-Key Support for Group Collaboration. ACM Transaction on Information and System Security, Vol. 6, No. 4, November 2003, pp. 547-565.

[FIA 94] Amos Fiat, Moni Naor. Broadcast Encryption. Advances in Cryptology. CRYPTO '93, Proceedings, LNCS, Vol. 773, 1994, pp. 481-491.

[FIS 92] M.J. Fischer, R.N. Wright. Multiparty secret key exchange using a random deal of cards. CRYPTO '91. LNCS, Vol. 576, Springer-Verlag, Berlin, 1992. pp. 141–155.

[FON 06] B. Fontan, S. Mota Gonzalez, T. Villemur, P. de Saqui-Sannes, J.P. Courtiat. UML-Based Modeling and Formal Verification of Authentication Protocols, IEEE ISSSE International. Symposium on Secure Software Engineering, mars 2006, Washington DC, USA.

[FON 07a] B. Fontan, S. Mota, P. de Saqui-Sannes, T. Villemur. Temporal Verification in Secure Group Communication System Design. International Conference on Emerging Security Information, Systems and Technologies SECURWARE'07. October 2007, Valencia, Spain.

[FON 07b] B. Fontan. Méthodologie de conception de systèmes temps réels et distribués en contexte UML/SysML. Memoire du doctorat, Institut National Polytechnique, Toulouse, Janvier 2008.

[FON 07c] B. Fontan, P. de Saqui-Sannes, S. Mota Gonzalez, T. Villemur, M.S. Bouassida, N. Chridi, I. Chrisment, S. Vigneron. Formal verification of secure group communications using AVISPA and TURTLE. Rapport LAAS N°07552, Octobre 2007.

[GOL 01] O. Goldreich. Foundations of Cryptography. ISBN 0-521-79172-3. Cambridge University Press. 2001.

[GON 96] L. Gong. Enclaves: Enabling secure collaboration over the internet. Proceedings of the Sixth USENIX Security Symposium, pp. 149-159. USENIX Association, July 1996.

[GUI 88] L. C. Guillou, J.J. Quisquater. A “paradoxical” identity-based signature scheme resulting from zero-knowledge. In advances in Cryptology – Crypto ’88, volume 0403 LNCS, pp. 216 – 231. 1990.

[HAR 97a] H. Harney, C. Muckenhirn. Group Key Management Protocol (GKMP) Architecture. IETF RFC 2094, 1997.

[HAR 97b] H. Harney, C. Muckenhirn. Group Key Management Protocol (GKMP) Specification. IETF RFC 2093, 1997.

[HEL 02] M. E. Hellman. An overview of Public Key Cryptography. IEEE Communications Magazine. 50th Anniversary Commemorative Issue. May 2002.

[HIL 97] M. A. Hiltunen, X. Han, and R. Schlichting. Real-time issues in Cactus. In Proceedings of the IEEE Workshop on Middleware for Distributed Real-Time Systems and Services, pp. 214-221, San Francisco, CA, Dec 1997.

[HIL 00] M. A. Hiltunen, R. D. Schlichting, The Cactus approach to building configurable middleware services, Proceedings of the Workshop on Dependable System Middleware and Group Communication, DSMGC 2000.

[HOR 02] G. Horng. Cryptanalysis of a key management scheme for secure multicast communications. IEICE Transactions in Communications, pp. 1050-1051. 2002.

[ING 82] I. Ingemarsson, D. Tang, C. Wong. A Conference Key Distribution System. IEEE Transactions on Information Theory, Vol. 28, pp. 714-720. 1982.

[IRR 03] J; Irrer, A. Prakash. Antigone: Policy-based Secure Group Communication System and AMirD: Antigone-based Secure File Mirroring System. IEEE Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX’03). 2003.

[JUR 02] J. Jürjens. UMLsec: Extending UMLfor secure systems development. Proc.of UML 2002- The Unified Modeling language, LNCS 2460, pp. 412-425, 2002.

[KAM 99] M. Kamata, T. Kawase. A. Watanabe, I. Sasase. Proposal and Analysis of Multicast Communication Method over the Department VPN. Trans. IEICE, Vol. J82B, No. 11, pp. 2061 – 2073, 1999.

[KER 06] A. D. Keromytis, J. L. Wright, T. de Raadt, M. Burnside. Cryptography As An Operating System Service: A Case Study. ACM Transactions on Computer Systems, Vol 24, No 1, February 2006.

[KIH 98] K.P. Kihlstrom, L.E. Moser, P.M. Melliar-Smith. The SecureRing protocols for securing group communication. IEEE 1998. Proceedings of the 31st Annual Hawaii International Conference on System Sciences, pp. 317-326. 1998.

[KIM 00] Y. Kim, A. Perrig, G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. Proceedings of the 7th ACM Conference on Computer and Communications Security (ACM CCS 2000). pp. 235-244.

[KIM 04a] Y. Kim, A. Perrig, G. Tsudik. Tree-based group key agreement. ACM Transactions on Information System Security. Pp. 60-96. 2004.

[KIM 04b] Y. Kim, A. Perrig, G. Tsudik. Group Key Agreement Efficient in Communication, IEEE Trans. Computers, vol. 33, no. 7, 2004.

[KOY 87] K. Koyama, K. Ohta, Identity-based conference key distribution systems. CRYPTO ’87, LNCS Vol. 293, Springer-Verlag, Berlin, 1987, pp. 175–185.

[KU 03] W.C. Ku, S.M. Chen. An improuved key management scheme for large dynamic groups using one-way function trees. Proceedings of the International Conference of Parallel Processing (ICPP), pp. 391-396. 2003.

[LIV 1.1] Projet National. SAFECAST Livrable L1.1. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Dossier de définition de l’application de communication de groupe »

[LIV 1.3] Projet National SAFECAST Livrable 1.3. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Dossier de définition des exigences et des besoins de sécurité et temporelles »

[LIV 1.4] Projet National SAFECAST Livrable 1.4. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Dossier de spécification des services de sécurité »

[LIV 2.2] Projet National SAFECAST Livrable 2.2. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Définition de certificats spécifiques aux applications de groupe »

[LIV 2.3] Projet National SAFECAST Livrable 2.3. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Conception d’infrastructure de confiance basée sur les certificats spécifiques aux applications de groupes »

[LIV 2.5] Projet National. SAFECAST Livrable L2.5. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Contribution à l’analyse du service de gestion de clés de groupe sur la base d’une modélisation UML 2.0 »

[LIV 3.1] Projet National SAFECAST Livrable L3.1. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Rapport sur l’analyse des algorithmes de cryptographie multicast version 1 »

[LIV 3.4] Projet National SAFECAST Livrable L3.4. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Evaluation des performances pour la validation du facteur d’échelle »

[LIV 4.1] Projet National SAFECAST Livrable L4.1. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Validation fonctionnelle de l’architecture »

[LIV 4.2] Projet National SAFECAST Livrable L4.2. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Rapport sur l’identification des points durs. V.2 »

[LIV 4.3] Projet National SAFECAST Livrable L4.3. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Identification et mise en œuvre des outils de simulation et vérification »

[LIV 4.4] Projet National SAFECAST Livrable L4.4. EADS, ENST Paris, LAAS-CNRS, LORIA, UTC Compiègne. « Bilan des contributions originales du projet SAFECAST et perspectives de recherche ouvertes »

[LOW 96] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. Tools and Algorithms for the Construction and Analysis of Systems, vol. 1055 of LNCS, pages 147-166. Springer-Verlag, 1996.

[McD 99] P. McDaniel, A. Prakash, P. Honeyman. CITI Technical report 99-2. Antigone: A Flexible Framework for Secure Group Communication. Center for Information Technology Integration. University of Michigan, Sept 1999.

[McD 00] P. McDaniel, A. Prakash. Antigone: Implementing Policy in Secure Group Communication. Technical Report CSETR -426-00, University of Michigan, May 2000.

[MER 78] R. C. Merkle. Secure Communication over an Insecure Channel. Commun. Ass. Comp. March., vol. 21, Apr. 1978, pp. 294-99.

[MIT 97] Mittra S. Iolus: A framework for scalable secure multicasting. ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communication. pp. 277-288. 1997.

[MOL 00] R. Molva, A. Pannetrat. “Scalable Multicast Security with Dynamic Recipient Groups”. ACM Transaction on Information and System Security, Vol. 3, No. 3, August 2000, pp. 136-160.

[MOT 05] S.MOTA GONZALEZ , B.FONTAN, UML-based modeling and formal verification of security protocols. 1st International Conference on Emerging Networking Experiments and Technologies (CoNEXT'2005). Student Workshop, Toulouse (France), Octobre 2005, pp.282-283.

[MOT 06] S.MOTA GONZALEZ , Conception de services de communications sécurisées dans des groupes multi utilisateurs. 7ème Congrès des Doctorants de l'Ecole Doctorale Systèmes (EDSYS), Tarbes (France), 11 Mai 2006, 6p.

[MOT 07] S. Mota, T. Villemur. Une approche UML pour l’analyse temporelle de communications de groupes sécurisées. Notere 2007. 7ème Conférence Internationale sur les NOuvelles TEchnologies de la REpartition. Marrakech, Marroc. 2007.

[MOS 95] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, C. A. Lingley-Papadopoulos and T. P. Archambault. The Totem system. Proceedings of the 25th Annual International Symposium on Fault-Tolerant Computing, Pasadena, CA, pp. 61-66. June 1995.

[NOU 98] G. Noubir. Multicast Security. European Space Agency, Projet : Performance Optimisation of Internet Protocol Via Satellite. 1998.

[OKA 00] Okazaki, M. Park, A. Watanabe, S. Seno, T. Ideguchi, M. Yabe. Realization Method of Flexible Private Network System. Trans IEEJ Vol. 120C, no. 8/9, pp 1242-1249. 2000.

[OMG 03] Object Management Group, Unified Modeling Language Specification. Version 1.5, mars 2003. http://www.omg.org/docs/formal/03-03-01.pdf, March 2003.

[ÖNE 07] M. Önen, R. Molva. Secure data aggregation with multiple encryption. 4th European conference on Wireless Sensor Networks (EWSN 2007), LNCS Volume 4373, January 2007.

[PAR 04] M. Park, N. Okazaki, S. Seno, K. Kashima, K. Oshima. A proposal and its evaluations of a re-keying method for dynamic secure group communications. Advanced Information Networking and Applications, 2004. AINA 2004.

[REC 901/DISSI/SCSSI] Recommandation pour la protection des systèmes d'information traitant des informations sensibles non classifiées de défense No. 901/DISSI/SCSSI le 2 mars 1994.

[REN 96] R. van Renesse, K. P. Birman, S. Maffeis. Horus: A flexible group communication system. Communications of the ACM, 39(4):76-83, April 1996.

[REI 95] M. K. Reiter. The Rampart Toolkit for Building High-Integrity Services. Lecture Notes in Computer Science 938, Springer-Verlag 1995, pp. 99-110.

[RHE 05] K. H. Rhee, Y.H. Park et G. Tsudik. A group key management architecture for mobile ad-hoc wireless networks. Journal of information science and engineering. 2005. pp. 415-428.

[RIV 78] R. Rivest, A. Shamir, L. Adelman. "On Digital Signatures and Public-Key Cryptography". Communications of the ACM, v. 27, n. 7, July 1978.

[ROD 01] O. Rodeh, K. P. Birman, D. Dolev. "The Architecture and Performance of Security Protocols in the Ensemble Group Communication System: Using Diamonds to Guard the Castle". ACM Transaction on Information and System Security, Vol. 4, No. 3, August 2001, pp. 289-319.

[RTL] Project RT-LOTOS. Page Web http://www.laas.fr/RT-LOTOS

[SAE 98] S. Saeednia, R. Safavi-Naini. Efficient identity-based conference key distribution protocols. Information security and privacy. 3rd conf. Vol. LNCS. Pp; 320-331. 1998.

[SAFECAST] Projet RNRT SAFECAST. Page Web http://rnrt-safecast.org/

[SAK 00] R. Sakai, K. Ohgishi, M. Kasahara. Cryptosystems based on pairing. SCIS 2000. 2000.

[SAQ 05] P. de Saqui-Sannes. Conception basée modèle des systèmes temps réel et distribués. Habilitation à diriger des recherches. ENSICA. LAAS-CNRS. Juillet 2005.

[SHA 01] D. Shands, J. Jacobs, R. Yee, E. J. Sebes. “Secure Virtual Enclaves: Supporting Coalition Use of Distributed Application Technologies”. ACM Transaction on Information and System Security, Vol. 4, No. 2, May 2001, pp. 103-133.

[SHE 03] A. T. Sherman et D. A. McGrew. Key establishment in large dynamic groups using one-way function trees. IEEE transactions on Software Engineering. 2003, pp. 444-458.

[SPREAD] SPREAD Toolkit. Page Web http://www.spread.org/

[STE 90] D. G. Steer, L. Strawczynski, W. Diffie, M. Wiener. A secure audio teleconference system. Proceedings on Advances in Cryptology (Santa Barbara, California, United States). S. Goldwasser, Ed. Springer-Verlag New York, New York, NY. 1990. pp. 520-528.

[STE 96] M. Steiner, G. Tsudik and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. Proceedings 3rd ACM Conference on Computer and Communications Security, 1996, pp. 31-37.

[STE 97] M. Steiner, G. Tsudik, M. Waidner. CLIQUES: A new approach to group key agreement. IEEE International Conference on Distributing Computing Systems. 1997. pages. 380-387.

[STE 00] M. Steiner, G. Tsudik, M. Waidner. Key agreement in dynamic peer groups. IEEE Transactions on Parallel and Distributed Systems. 2000. 11 (8): 769-780.

[TAN 96] Tanenbaum, Andrew S. Computer networks. Upper Saddle River, N.J. Prentice Hall PTR, 1996.

[TEM 01] J. Templemore-Finlayson, S. Budkowski. Group Communication and Multicast. ICN 2001. LNCS 2093 pp. 649-656. Berlin 2001.

[TETRA] TETRA. Page Web http://www.tetramou.com/

[TETRAPOL] TETRAPOL. Page Web http://www.tetrapol.com/

[TURTLE] Profil TURTLE. Page Web http://www.laas.fr/TURTLE

[UML] Object Management Group, “Unified Modeling Language Specification”, Version 1.5, page Web http://www.omg.org/docs/ formal/03-03-01.pdf, March 2003.

[VER 90] J.C.Lloret, P. Azema, F.Vernadat. Compositional design and verification of communication protocols using labelled Petri nets. DIMACS'90. Workshop on Computer Aided Verification, New Brunswick (USA), 18-21 Juin 1990, 17p.

[WAL 98] D. Wallner, E. Harder, R. Agee. Key management for multicast: Issues and architectures. Internet Draft, draft-wallner-key-arch-01.txt, Internet Eng. Task Force. RFC 2627. 1999.

[WAN 05] W. Wang, B. Bhargava. Key Distribution and Update for Secure Inter-group Multicast Communication. SASN 05. Virginia USA. 2005.

[WHE 94] B. Whetten, S. Kaplan, and T. Montgomery. A high performance totally ordered multicast protocol. Available from research.ivv.nasa.gov by ftp /pub/doc/RMP/RMP dagstuhl.ps, August 1994.

[WIK 07] Wikipédia. L'encyclopédie libre. Wikimedia Foundation, Inc. Page Web http://fr.wikipedia.org/wiki/Accueil

[WON 98] C. K., Wong, M. Gouda, S.S. Lam. Secure group communication using key graphs. SIGCOMM '98. University of Texas at Austin, Computer Science Technical report TR 97-23, pp. 68-79. 1998.

[WU 05] Bing Wu, Jie Wu, Eduardo B. Fernandez, Mohammad Ilyas, Spyros Magliveras. Secure and efficient key management in mobile ad hoc networks. ELSEVIER Journal of Network and Computer Applications. July 2005.

[ZIM 80] H. Zimmerman. "OSI Reference Model, the ISO Model of Architecture for Open Systems Interconnection". IEEE Transactions on Communications, vol. COM-28, Avril 1980.

[ZOU 04] X. Zou, B. Ramamurthy. A block-free tree-based group Diffie-Hellman key agreement for secure group communications. Proceedings of International Conference on Parallel and Distributed Computing and Networks, Innsbruck, Austria. Pp. 288-293. 2004.

[ZOU 05] X. Zou, B. Ramamurthy, S. Magliveras. Secure Group Communications Over Data Networks. Springer 2005.

[ZOU 07] B. Zouari, M. Kallel, A. Cavalli. An Incremental Authentication Study using SIM-IP Cards for IEEE 802.11 Wireless LANs. 4th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, SETIT 2007, March 2007.

AUTHOR: Sara MOTA

TITLE: Modeling and Verifiying Protocols for Secure Group Communication.

PhD SUPERVISORS: Thierry VILLEMUR, Michel DIAZ

DATE AND PLACE: LAAS–CNRS, 5 juin 2008

ABSTRACT:

Systems that implement communications in the form of group multicast have increasingly raised security problems. The protection mechanisms applied to that communication rely on symmetrical and asymmetrical key exchanges, and the way these mechanisms are selected does influence the system's efficiency. Following an in depth analysis of the needs captured by these systems, we defined a model for representing the dynamics of groups, as well as communication among group members. We defined one system architecture which focuses on key creation, exchange and management functions. The system was modeled in UML 2.0 and checked against security and temporal properties. The approach we followed to investigate temporal requirements may be extended to a broad variety of distributed systems.

KEY WORDS:
Group Communication, Security, Key Management, UML Modeling, Temporal Verification, Performance.

DOMAIN: Computer Sciences

AUTEUR : Sara MOTA

TITRE : Modélisation et vérification de protocoles pour des communications sécurisées de groupes

DIRECTEURS DE THESE : Thierry VILLEMUR, Michel DIAZ

LIEU ET DATE DE SOUTENANCE : LAAS–CNRS, 5 juin 2008

RESUME :

Dans le monde des systèmes qui utilisent des communications sous forme de diffusion de groupes, le critère de sécurité devient un facteur de plus en plus important. Le choix des mécanismes pour la protection de cette communication, mécanismes basés sur des échanges de clés symétriques et asymétriques, influe sur l'efficacité du système.
Nous avons procédé à l'analyse des besoins et nous avons défini un modèle qui permet de représenter la dynamique des groupes et la communication entre leurs membres. Nous avons défini l'architecture d'un système dont l'élément central est la fonction de création, d'échange et de mise en place correcte des clés. La modélisation de ce système dans un environnement UML 2.0 a permis son analyse en termes de garantie de propriétés temporelles et de sécurité. L'approche suivie pour l'étude des exigences temporelles est généralisable à de nombreux systèmes distribués.
La valorisation de nos études a été faite dans le cadre du projet national RNRT SAFECAST.

MOTS CLES :
Communications de groupes, Sécurité, Gestion de clés, Modélisation UML, Vérification temporelle, Performance.

DISCIPLINE : Informatique