# La sécurité des futures architectures convergentes pour des services personnalisés : aspect architectural et protocolaire

Ali Hammami

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2013-ENST-0039

EDITE ED 130

# Doctorat ParisTech

## T H È S E

**pour obtenir le grade de docteur délivré par**

## Télécom ParisTech

### Spécialité " Informatique et Réseaux "

*Présentée et soutenue publiquement par*


**Ali HAMMAMI**


1er JUILLET 2013

# La Sécurité des Futures Architectures Convergentes pour des Services Personnalisés : Aspects Architectural et Protocolaire

Directrice de thèse : **Pr. Noëmie SIMONI**

**Jury**

| | | |
|---|---|---|
| **M. Abdelmajid BOUABDALLAH**, | Professeur, Université de Technologie de Compiègne | Rapporteur |
| **Mme Francine KRIEF**, | Professeur, Institut Polytechnique de Bordeaux | Rapporteur |
| **M. Samir TOHME**, | Professeur, Université de Versailles | Examinateur |
| **Mme Brigitte LONC**, | Docteur, Project Manager, Renault SA | Examinatrice |
| **M. Pascal URIEN**, | Professeur, Télécom ParisTech | Examinateur |
| **Mme Noëmie SIMONI**, | Professeur, Télécom ParisTech | Directrice de Thèse |

T H È S E


**Télécom ParisTech**
**Ecole de l'Institut Mines Télécom – membre de ParisTech**
46, rue Barrault – 75634 Paris Cedex 13 – Tél. + 33 (0)1 45 81 77 77 – www.telecom-paristech.fr

**La Sécurité des Futures Architectures Convergentes pour des Services Personnalisés : Aspects Architectural et Protocolaire**

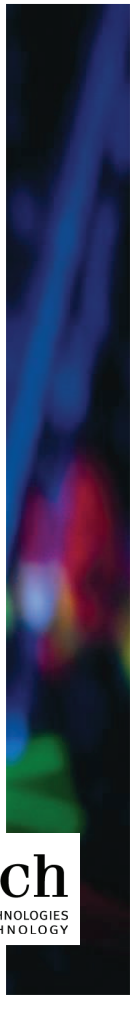

**RESUME :**

L'émergence et l'évolution des réseaux de nouvelles génération (NGN) a soulevé plusieurs défis surtout en termes d'hétérogénéité, de mobilité et de sécurité. En effet, l'utilisateur est capable, dans un tel environnement, d'avoir accès à plusieurs réseaux, à travers différents terminaux, avec un choix vaste de services fournis par différents fournisseurs. De plus, les utilisateurs finaux demandent à être constamment connectés n'importe où, n'importe quand et n'importe comment. Ils désirent également avoir un accès sécurisé à leurs services à travers une session dynamique, seamless et continue selon leurs préférences et la QoS demandée.

Dans ce contexte, la sécurité représente une composante majeure. Face à cette session user-centric sécurisée, plusieurs défis se posent. L'environnement est de plus en plus ouvert, de multiples services ne sont pas connus d'avance et nous avons une diversité de communications entre les services et les utilisateurs. L'hétérogénéité des ressources (terminaux, réseaux et services) impliquées dans la session de l'utilisateur accentue la complexité des tâches de sécurité. Les différentes déclinaisons de mobilité (mobilité de l'utilisateur, mobilité du terminal, mobilité du réseau et mobilité du service) modifient la session user-centric que l'on veut unique, sécurisée et seamless avec la délivrance d'un service continu.

Cet environnement, ouvert, hétérogène et mobile, présente des risques significatifs de sécurité et devient de plus en plus vulnérable. Pour cela, une nouvelle solution de sécurité, qui répond aux exigences user-centric dans un environnement NGN, est nécessaire. Pour ce faire, nous proposons une nouvelle architecture de sécurité qui offre une composition de service de sécurité dynamique et seamless (Securityware). Cette architecture est organisée selon quatre niveaux de visibilité (équipement, réseau, service et utilisateur) et chaque ressource (terminal, réseau et service) est considérée comme un fournisseur de service. Ainsi, nous visons à travers notre proposition basée sur l'approche de « security as a service » à assurer un accès sécurisé aux services et à préserver la continuité de sécurité dans une session sans couture. Afin de contrôler et évaluer la sécurité dans notre architecture, nous introduisons un service d'audit basé sur la QoS.

Au niveau protocolaire, nous proposons le protocole SIP+. Il s'agit d'une extension du protocole SIP (Session Initiation Protocol) qui permet d'établir une session de service personnalisée et mobile. Elle est basée sur la négociation de la QoS pour assurer et maintenir la QoS demandée de bout en bout. Afin de satisfaire les exigences de sécurité, nous devons aussi sécuriser le protocole proposé SIP+ contre les attaques et les vulnérabilités potentielles afin de fournir une session de service sécurisée en respectant les besoins user-centric. Pour ce faire, nous avons introduit un jeton de sécurité qui favorise le maintien de la continuité de la session de bout en bout et qui gère la continuité de la sécurité au cours de la mobilité et du changement de contexte de l'utilisateur. Ce jeton

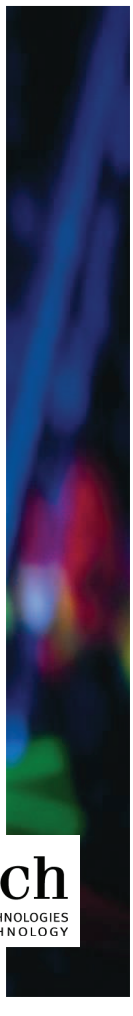



TELECOM ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

assure entre autre une authentification unique dans un environnement trans-organisationnel.

**Mots clés :** “Security as a service, NGN/NGS, mobilité, hétérogénéité, QoS, session “user-centric”, continuité de la sécurité, SIP+ basé sur le jeton, SIP+ sécurisé



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

**Security of Converging Future Architectures for Personalized Services: Architectural and Protocol Aspects**


**ABSTRACT :**

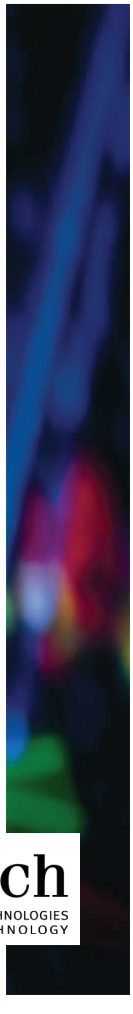
The emergence and evolution of Next Generation Networks (NGN) have raised several challenges mainly in terms of heterogeneity, mobility and security. In fact, the user is able, in such environment, to have access to many networks, via multiple devices, with a vast choice of services offered by different providers. Furthermore, end-users claim to be constantly connected anywhere, anytime and anyhow. Besides, they want to have a secure access to their services through a dynamic, seamless and continuous session according to their preferences and the desired QoS.

In this context, security represents an important concern. In fact, this user-centric session should obviously be secured. However, many challenges arise. In such environment, system boundaries, which were well delimited, become increasingly open. Indeed, there are multiple services which are unknown in advance and multiple communications between services and with users.

Besides, heterogeneity of involved resources (terminals, networks and services) in the user session increases the complexity of security tasks. In addition, the different types of mobility (user, terminal, network and service mobility) affect the user-centric session that should be unique, secure and seamless and ensure continuity of services.

This open, heterogeneous and mobile environment presents significant risks in terms of security and becomes increasingly vulnerable. Therefore, a new security solution, which responds to user-centric requirements in NGN environment, should be provided. For this purpose, we propose a novel security architecture which ensures a dynamic and seamless security service composition (Securityware). This architecture is organized in four visibility levels (equipment, network, service and user) and each resource (terminal, network and service) is considered as a service component. Thus, we aim through our proposal based on security as a service approach to ensure a secure service ac-cess and to guard security continuity within a seam-less session. To control and evaluate security in our architecture, an audit service based on QoS is proposed.

In the protocol level, we propose SIP+ protocol which represents an extended version from Session Initiation Protocol (SIP) that permits to establish a personalized and mobile service session. It is based on QoS (Quality of Service) negotiation to maintain the required end-to-end QoS. To achieve security requirements, we should also secure the proposed SIP+ protocol against the potential attacks and vulnerabilities in order to provide a secure service session while respecting our user-centric needs. Therefore, we introduce a Token Security which furthers end-to-end session continuity and manages security continuity during mobility and user context change. This

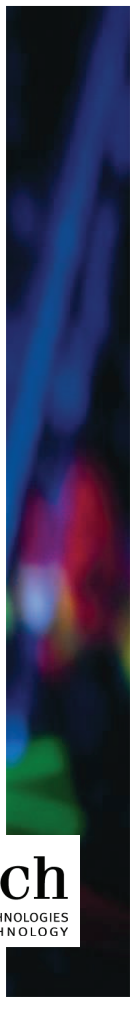
TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

Token permits to have a unique user authentication in a cross-organizational environment.

**Keywords** : security as a service, NGN/NGS, mobility, heterogeneity, QoS, user-centric session, security continuity, token based SIP+, secure SIP+



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

# Remerciements

*En tout premier lieu, je tiens à remercier ma directrice de thèse, Madame Noëmie SIMONI, pour avoir accompagné mes travaux de recherche et mes réflexions par ses précieux conseils depuis mon stage de master et durant les années de ma thèse. Je vous assure ma profonde reconnaissance pour votre aide, vos encouragements et le temps que vous m'avez consacré.*

*Je remercie Mesdames Francine KRIEF et Brigitte LONC, et Messieurs Abdelmajid BOUABDALLAH, Samir TOHME et Pascal URIEN pour m'avoir fait l'honneur d'accepter de faire partie de mon jury et pour l'intérêt qu'ils ont porté à mon travail.*

*Mes gratitudes s'adressent également à Mesdames Martha DWEYER, Florence BESNARD, Hayette SOUSSOU et Nazha ESSAKKAKI, et Monsieur Dominique ROUX pour leur aide durant mon cycle doctoral.*

*Un grand merci à Monsieur Hassane AISSAOUI, à tous mes collègues et tous les membres du projet UBIS pour les moments que nous avons partagé durant l'avancée de nos travaux. Je remercie également tous mes amis pour leur soutien et les moments que nous avons vécu ensemble.*

*Enfin, je dédie ce travail*

*A la mémoire de mon père qui m'a tant donné pour faire de moi ce que je suis,*

*A ma mère qui nulle expression ne puisse exprimer ce que je lui dois pour son amour, ses sacrifices et son dévouement,*

*A mes sœurs et mon frère et leurs petites familles pour leur amour, leur soutien et leur confiance.*

*Ali HAMMAMI*

# Table des matières

# Table des figures

# Liste des tableaux

# 1 Introduction Générale

## 1.1. Contexte d'étude

Le secteur de télécom n'a cessé de se développer dans l'objectif de satisfaire les nouveaux besoins de communications. L'évolution majeure se manifeste par la migration vers les réseaux de nouvelle génération (Next Generation Network : NGN). En plus, le degré de concurrence entre les fournisseurs de service a augmenté d'une manière remarquable. Leurs objectifs principaux sont de maximiser leur RoI (Return on Investment) et de satisfaire leur clientèle. Ainsi, ils s'investissent à offrir des services orientés utilisateurs. Ceci nécessite l'adoption d'une nouvelle vision des services qui permet d'assurer le paradigme « *anywhere, anytime, anyhow, every service, everyone* ». Il s'agit des services de nouvelle génération (NGS). Cet environnement est dynamique et ouvert, les technologies déployées sont relativement complexes, et les éléments interconnectés sont mobiles et hétérogènes. Ceci influence la sécurité et expose le système à des menaces potentielles.

C'est dans ce nouveau paysage NGN/NGS que se situe notre contexte d'étude. Nous nous intéressons aux besoins « user-centric » qui constituent, entre autre, l'objectif de notre projet UBIS (« User-centric »: uBiquité et Intégration de Services) (§1.1.1). Nous traitons également les problématiques d'hétérogénéité et de mobilité (§1.1.2) imposées par ce contexte. Finalement, nous évoquons surtout les besoins de sécurité de la session de service établie dans un tel contexte (§1.1.3) qui présentent le point focal de notre étude.

### 1.1.1 UBIS: « User-centric », uBiquité et Intégration de Services

De nos jours, l'utilisateur final demande un accès sécurisé et continu sans contraintes à ses services. Il désire une session dynamique et personnalisable selon son contexte, ses besoins et ses préférences. Cette session doit être également unique, continue, transparente et sécurisée malgré l'hétérogénéité et la mobilité imposées par l'environnement NGN qui constitue le nouveau paysage des télécommunications.

C'est dans ce cadre que se situe le projet UBIS qui propose une architecture de service offrant des services personnalisés d'une façon ubiquitaire et continue et pouvant s'adapter à l'environnement de l'utilisateur (localisation, type du terminal, préférences, etc.). Ces services sont désignés par « *Next Generation Services* : NGS ». Inspiré par l'évolution du NGN, le projet UBIS cherche à supporter les défis de mobilité, d'hétérogénéité et principalement l'approche « *user-centric* ».

Cette approche consiste à procurer à l'utilisateur une position privilégiée qui se situe au centre du dispositif. Ceci implique qu'aucune contrainte ne peut lui être imposée par le

système (System-centric) ou par l'application (Application-centric). Il se déplace bien sûr, mais il peut aussi changer de terminal, définir ses préférences, solliciter des services auprès de plusieurs fournisseurs.

L'approche « *user-centric* » permet également à l'utilisateur d'avoir une session de service unique et personnalisée. Elle est unique car elle regroupe l'accès à tous les services que l'utilisateur désire solliciter durant l'espace temps qui lui convient. Ce qui veut dire que la gestion et le contrôle de la session en termes de QoS et de sécurité doivent être transparents vis-à-vis de l'utilisateur, et que la continuité de services doit être garantie. Elle est personnalisée, ce qui permet à l'utilisateur de faire sa propre composition de services selon une logique qui correspond au mieux à son contexte ambiant, ses préférences fonctionnelles et ses exigences en termes de QoS et de sécurité. Cette session peut contenir tous types de services (Telco, Web et IT) qui pourraient être multi-domaines et multi-fournisseurs, d'où les objectifs d'intégration de service et du contexte trans-organisationnel du projet UBIS. L'utilisateur peut ainsi choisir librement ses services et avoir une composition dynamique qui s'adapte à tout changement durant sa session.

C'est dans le cadre de ce projet UBIS que s'inscrit la présente thèse qui s'intéresse particulièrement aux aspects de la sécurité.

### 1.1.2 NGN : Mobilité et Hétérogénéité

Next Generation Network (NGN) est une nouvelle architecture de réseaux de communication dont le principe est d'utiliser les technologies de transport en mode paquet, réservé jusqu'alors pour les données, pour transporter l'ensemble des services de télécommunications. Elle assure la convergence entre les réseaux fixes, mobiles et d'Internet en offrant un accès aux services indépendant du réseau d'accès et du terminal avec la mobilité requise. De plus, les réseaux NGN convergent vers une structure horizontale en séparant entre les services et les technologies de transport sous-jacentes, pour permettre une évolutivité plus importante du réseau.

Pour accompagner cette évolution des réseaux vers le NGN, l'évolution des services s'avère impérative. La motivation globale est de faciliter l'intégration des usages afin que l'utilisateur ait un « service tout compris » et de permettre l'intégration des offres de tous les acteurs du marché. L'utilisateur d'aujourd'hui désire avoir l'accès à tout type de services sans interruption de session et avec un accès sécurisé. Le contexte qui nous intéresse est celui qui associe la convergence de solution des problématiques de mobilité, d'hétérogénéité et des besoins centrés sur l'utilisateur.

- **Hétérogénéité**

Actuellement, l'utilisateur vit dans un monde hétérogène multi-terminal, multi-accès, multi-opérateurs, multifournisseurs, multi-services, etc. En effet, l'utilisateur est capable d'avoir accès à différents réseaux, à travers des terminaux multiples (Smartphone, PDA, etc.), avec un choix vaste des services qui sont fournis par plusieurs fournisseurs et déployés sur différentes plates-formes.

Tous les éléments de ces environnements hétérogènes doivent collaborer et participer à la session de l'utilisateur pour lui fournir un service global tout en respectant ses préférences et les contraintes de qualité de service (QoS) et de sécurité.

Afin d'avoir une perception homogène, toutes ces ressources sont conçues d'une manière uniforme selon une approche SOA et un modèle de QoS, pour traduire leur comportement, préconisé dans UBIS.

Cependant, l'hétérogénéité de différentes ressources impliquées dans la session de l'utilisateur augmente la complexité en termes de sécurité. C'est au niveau de l'analyse des dysfonctionnements de comportement et de l'évaluation des risques qu'il faudra repérer les impacts de ce nouveau contexte.

- **Mobilité**

Dans ce nouveau paysage de nouvelle génération de réseaux et de service, l'utilisateur a la possibilité de se déplacer et d'accéder à ses services au gré de ses besoins et selon ses préférences. En effet, la grande couverture et la diversité des réseaux d'accès a permis à l'utilisateur d'accéder à ses services n'importe où sans tenir compte de sa localisation qui n'est plus considérée comme un obstacle.

Ceci impacte la délivrance du service de bout en bout sécurisé. En effet, la mobilité rend la continuité de service et la sécurité de la session de l'utilisateur de plus en plus complexes.

Cette mobilité se décline en deux types : spatiale et temporelle. La mobilité spatiale englobe la mobilité de terminal, la mobilité de l'utilisateur et la mobilité de service. La mobilité temporelle est représentée par la mobilité de session.

- *Mobilité de terminal :* cette mobilité représente le fait que le terminal change de localisation, c'est-à-dire, le terminal se déplace soit à travers différents points d'accès d'un réseau, soit à travers différents réseaux d'accès, tout en maintenant l'accès à un même ensemble de services. Ceci veut dire que l'utilisateur doit continuer à accéder à toute la liste des services auxquels il a souscrit.

  Pour permettre un déplacement de terminal sans coupure, il existe actuellement deux solutions : *le Roaming,* qui permet le passage entre deux domaines administratifs ou d'un opérateur à un autre et *le handover,* qui est un processus qui permet à un terminal mobile d'effectuer le passage entre deux points d'attachement de façon transparente. Le changement de point d'attachement entraîne parfois une déconnexion momentanée du terminal mobile et des perturbations des communications en cours. Néanmoins l'objectif est d'assurer des communications sans coupure.

- *Mobilité de l'utilisateur :* Actuellement, un utilisateur peut avoir plusieurs terminaux à partir desquels il peut accéder et utiliser ses services. On parle de mobilité de l'utilisateur lorsque ce dernier change de terminal durant la même session qui doit maintenir son niveau de sécurité. Pour résoudre les problèmes que pose la mobilité de l'utilisateur, nous devons gérer cette jointure à travers la connaissance de son parc de terminaux et de sa localisation pour qu'il puisse, dans toutes les occasions (lieu et temps) et indépendamment des terminaux utilisés, être accessible et ceci en tenant en compte de ses préférences et des contraintes de QoS et de sécurité.

- *Mobilité de service :* Les deux types de mobilité que l'on vient de citer ci-dessus représentent des mobilités d'ordre spatial qui sont relatifs à des aspects de connectivité. Dans un monde « user-centric », il faut également considérer la mobilité de service qui permettra l'accès aux services ubiquitaires. Avec l'évolution de la fourniture de services, un même service demandé par l'utilisateur peut être déployé

dans différentes plates-formes et fourni par plusieurs fournisseurs de services. La mobilité de service a comme objectif de respecter les préférences de l'utilisateur et d'assurer la QoS et la sécurité de bout en bout.

- *Mobilité de session :* La combinaison des quatre types de mobilité cités ci-dessus induit une mobilité d'ordre temporel qui est la mobilité de session. L'objectif principal est de fournir à l'utilisateur une session « sans coupure » et « sans couture ». Il faudra utiliser des mécanismes permettant de garder l'unicité et la continuité de la session qui doit être sécurisée et assurer la QoS de bout en bout.

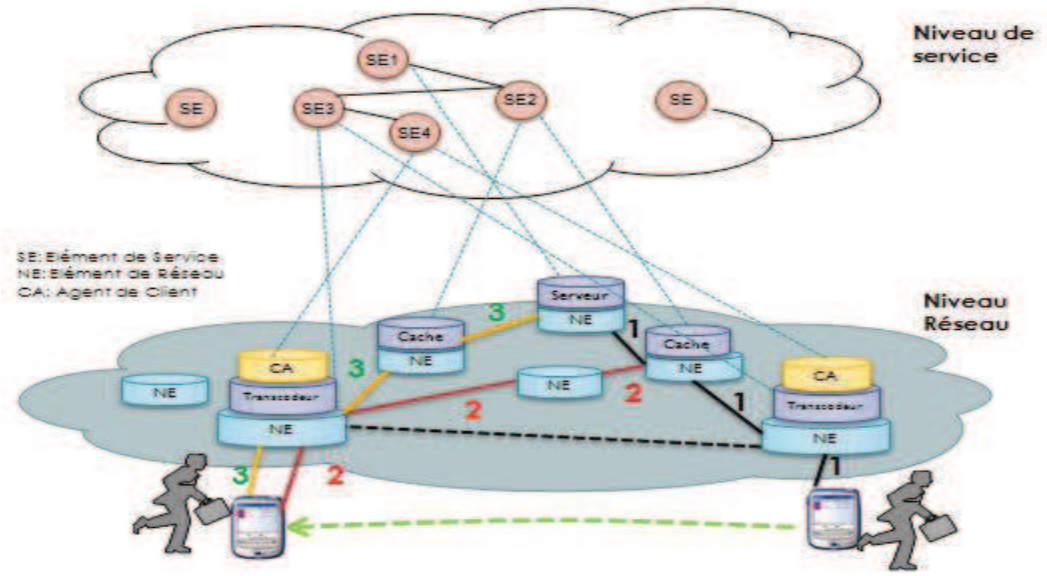



***Figure 1 :*** *Mobilité de la session.*

Prenons le cas d'une vidéo à la demande (VoD). Le service (simplifié) est rendu avec les composants de service (voir Figure 1) : agent client (SE4), transcodeur (SE3), cache (SE2), diffuseur (SE1). La première route (1) est trouvée, en fonction de la logique de service, des capacités des composants de service et des capacités des bus de services (supportés par un réseau d'acheminement). L'utilisateur se déplace (mobilité du terminal), donc nouveau point d'accès, la route (2) est modifiée. Si la QoS n'est plus respectée, une route (3) avec des composants de services fonctionnellement équivalents sera empruntée. Nous venons de décrire la mobilité de session. L'automatisation des processus favorisera la continuité de service.

### 1.1.3 La sécurisation de la session

La session regroupe l'ensemble des composants mis en relation à partir du terminal de l'utilisateur jusqu'aux plates-formes de services, en passant par les composants du réseau. Afin d'assurer la sécurisation de cette session, on doit mettre en place les mécanismes nécessaires de sécurité. Fondamentalement, les enjeux de la sécurité restent les mêmes : authentifier les utilisateurs, gérer les autorisations, assurer l'intégrité et la confidentialité des données, et imposer un caractère non répudiable aux actions réalisées.

En fait, la sécurité à un spectre très large, mais dans notre contexte mobile et hétérogène, il s'agit de s'assurer de la cohérence et de la fiabilité de trois fonctions importantes

permettant un accès sécurisé à la session. Il s'agit de l'identification, l'authentification et l'autorisation. En effet, il nous faut d'abord avoir les moyens d'identifier correctement notre utilisateur. Une identification s'appuie sur une simple déclaration comme la réception ou la lecture d'un code d'identification (identifiant, numéro de série, code barre …). Ce code d'identification n'est pas supposé secret. C'est une donnée publique. Puis, au moment de l'utilisation d'un terminal, de l'accès à un réseau ou à un service, il faut authentifier cet utilisateur, s'assurer que c'est la bonne personne. L'authentification s'appuie sur un élément de preuve comme un secret partagé ou un secret asymétrique. Elle permet de s'assurer avec un niveau de confiance raisonnable de l'identité de l'utilisateur. Pour finir, l'utilisateur, est-il autorisé à utiliser le terminal, le réseau ou le service ? A-t-il le droit ? Ce droit est à mettre en regard avec les permissions signifiées par les différentes offres. En conclusion, l'autorisation est chargée d'évaluer les droits effectifs sur la base des informations (identité et preuves d'authenticité) fournies par l'authentification. L'autorisation est donnée cas par cas à un utilisateur, par application, en fonction des droits associés au rôle (ou par ressource en fonction des privilèges).

Pour contrôler et avoir un suivi continu des différentes fonctions de la sécurité, une fonction d'audit est nécessaire.

## 1.2. Périmètre de la thèse

Après la description de notre contexte d'étude, nous délimitons, comme le montre la Figure 2, le périmètre de notre thèse. Il se situe à l'intersection des trois contextes abordés ci-dessus : le contexte du projet UBIS, la mobilité et l'hétérogénéité des réseaux NGN et la sécurité de la session.

Alors, nous avons besoin de repenser l'architecture de sécurité de telle sorte qu'elle permette de sécuriser la session de l'utilisateur en respectant les besoins « user-centric » dans un environnement, où la mobilité et l'hétérogénéité sont omniprésentes. Cette architecture de sécurité doit répondre principalement aux besoins suivants :

- Un accès sécurisé à l'ensemble des services demandés par l'utilisateur.
- Une sécurité d'accès appliquée à des services convergents multi-fournisseurs.
- Une session unique en préservant la continuité de services et de sécurité dans un contexte mobile.
- Une sécurité de bout en bout.

Afin de satisfaire ces besoins, nous nous intéressons dans notre thèse aux aspects architectural et protocolaire lors de la conception de notre proposition de sécurité.

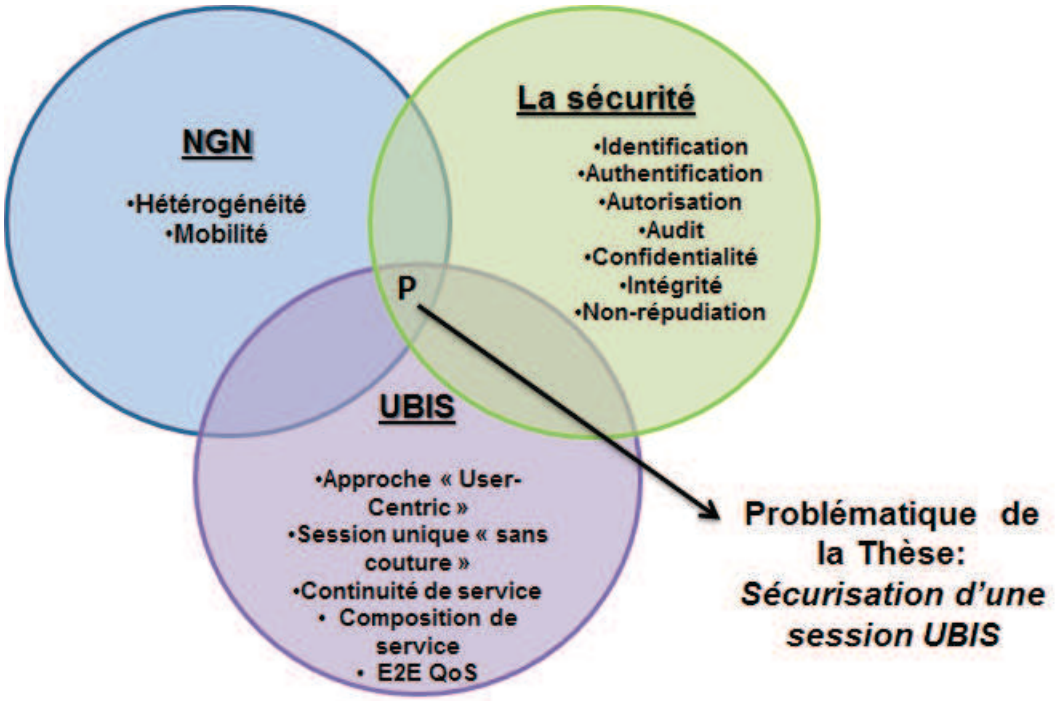



***Figure 2 :*** *Le périmètre de la thèse.*

### 1.3. Problématiques

L'interconnexion du « Monde UBIS » et de celui de la sécurité introduit un certain nombre de verrous. En effet, l'utilisateur d'aujourd'hui, et plus encore celui de demain, est nomade. Il réclame l'accès à n'importe quel service sans aucune barrière technique, temporelle, économique ou géographique, afin d'obtenir toujours la meilleure session, en adéquation avec son contexte et/ou ses préférences. Ces besoins « user-centric » introduisent une nouvelle perception du paysage des télécommunications. L'industrie de télécommunications doit proposer des nouvelles solutions, afin de répondre à ces besoins de : « anywhere, anytime, anyhow, every services, everyone », en assurant la continuité de service tout en étant sécurisé.

*Soit **V1** le premier verrou de la spécification des mécanismes de sécurité dans un tel contexte. Il faut qu'ils deviennent des services pour que l'on puisse les composer selon les besoins de lieu et de temps (spatial et temporel) et les préférences de l'utilisateur.*

Pour assurer une continuité de service et de sécurité, avec la chaîne des mobilités spatiales, le système doit gérer dynamiquement la session de l'utilisateur et son unicité de bout en bout (seamless) en temps réel. En plus de la continuité de service, on doit garantir la sécurité d'accès aux services afin d'avoir une session sécurisée.

*Ceci nous a amené à identifier le deuxième verrou **V2** qui traite la sécurisation de la session mobile et dynamique basée sur une composition de service.*

Le principal problème de toutes les ressources qui vont devoir collaborer pour fournir un service global est celui de l'hétérogénéité. Or notre objectif est d'avoir une QoS de bout en bout, à laquelle participe chaque composant de service. C'est pourquoi, la vision UBIS est d'avoir une perception homogène de toutes ces ressources à travers un modèle de QoS qui représente le comportement et les caractéristiques du service demandé. Le modèle s'appliquera à tout objet du système, que cela soit un équipement, un réseau d'accès ou un composant de service qu'il soit applicatif ou de sécurité.

Le suivi de la QoS et l'analyse des dysfonctionnements de comportement des services de sécurité sont –ils suffisants pour le point de vue de la sécurisation.

*Soit alors le troisième verrou **V3** : l'Audit de la sécurité.*

Maintenant d'autres verrous apparaissent au niveau de la sécurité tout au long de la session.

Les communications personnelles induisent des mobilités essentiellement d'ordre spatial comme la « mobilité du terminal », la « mobilité de l'utilisateur » et la « mobilité du réseau ». La « mobilité du terminal » affecte la continuité de la connexion. Lorsque l'utilisateur passe d'un terminal à un autre, cela relève de la « mobilité de l'utilisateur », nous devons alors résoudre les adaptations pour préserver la personnalisation. La « mobilité du réseau » concerne le déplacement de l'infrastructure du support de transport. Ces trois types de mobilité sont relatifs à des aspects de connectivité.

On peut dire que la mobilité du terminal ne pose pas trop de problème du point de vue de la sécurité car l'utilisateur est confondu avec son terminal, il est identifié à travers ce terminal. Il n'en est pas de même avec la mobilité de l'utilisateur. En effet, l'utilisateur change de terminal mais pas de service ! Qu'en est-il de son authentification ? Et comment assurer la transparence à sa mobilité ?

*Ainsi apparait le quatrième verrou **V4** lié à la mobilité de l'utilisateur (changement de terminal) et à son authentification.*

Dans le monde réel, il faut également considérer l'évolution des services et la multitude des offres. Un même service demandé par l'utilisateur peut être offert par plusieurs fournisseurs et supporté par différentes plates-formes de service. Et si nous faisons notre « marché » auprès de différents fournisseurs, comment assurer la continuité de la sécurité ?

*L'espace de confiance introduit un cinquième verrou **V5** qu'expose cet environnement trans-organisationnel (présence de plusieurs fournisseurs de service).*

Face à cette diversité de services, quand l'utilisateur se déplace, comment peut-il faire son choix pour avoir la meilleure adéquation à ses besoins ? Nous avons dénommé ce

quatrième type de mobilité: la «mobilité de service». Cette «mobilité de service» induit, en fait, une autre mobilité d'ordre temporel que nous avons dénommé «mobilité de session».

Selon le contexte et les déplacements de l'utilisateur, une nouvelle composition de service pourrait être adoptée. Si la QoS n'est plus respectée, une route avec des composants de services fonctionnellement équivalents sera empruntée en respectant toujours la logique de service, On parle alors de la mobilité de session. L'automatisation des processus favorise la continuité de service.

Mais quel est l'impact de cette mobilité sur la sécurité ? L'utilisateur doit-il s'authentifier à chaque service sollicité ? D'autre part, comment l'autorisation peut-elle être validée pour chaque service demandé de façon transparente vis-à-vis de l'utilisateur ?

*Par conséquent, notre sixième verrou **V6** porte sur l'authentification et l'autorisation pour tous les composants de services sollicités.*

UBIS a fait le choix de SIP pour l'établissement de la session. Mais est-il robuste aux attaques ? La sécurisation de la session de service « user-centric » consiste aussi à protéger les informations confidentielles et critiques, relatives à l'utilisateur et à sa session, échangées durant cette session. Ceci revient principalement à sécuriser le protocole permettant l'établissement et le maintien de cette session.

*C'est pourquoi, se pose le septième verrou **V7** de la sécurité du protocole de signalisation dans la session (dans notre cas, il s'agit du protocole SIP+).*

## 1.4. Organisation

Ce rapport de thèse est organisé en chapitres réparties en trois principales parties précédées par une introduction générale et terminées par une conclusion.

- Le **Chapitre 1** est la présente introduction générale qui a pour objectif d'introduire notre contexte d'étude, de définir le périmètre de notre thèse et d'identifier les problématiques à traiter. Nous avons commencé par la description des conditions et des besoins imposés par notre contexte qui se situe dans un environnement NGN caractérisé par la mobilité et l'hétérogénéité. Ce contexte fait partie du projet UBIS (Ubiquité et Intégration de Services) dans lequel nous nous intéressons principalement aux aspects de la sécurisation de la session. Sept verrous sont identifiés. Ceux d'ordre architectural sont étudiés dans la partie 1 et ceux d'ordre protocolaire sont traités dans la partie 2. La partie 3 couvre les aspects de faisabilité.

La **Partie 1** adresse donc l'aspect architectural de la sécurité en présentant le concept du « Security as a Service ». Cette partie se compose de trois chapitres :

- Le **Chapitre 2** est un chapitre introductif de cette partie et décrit sa structure.
- Le **Chapitre 3** analyse la sécurité dans les architectures orientées service existantes, à savoir les Web Services, SOA et le Cloud. Nous discutons alors leurs avantages et leurs inconvénients face aux besoins qui régissent notre contexte NGN/NGS.

- Le **Chapitre 4** présente notre proposition au niveau architectural de la sécurité. Nous introduisons le concept « Security as a Service » sur lequel se base notre architecture de sécurité. Nous spécifions alors nos services de sécurité répartis en trois catégories : les services de base de sécurité à savoir l'identification, l'authentification et l'autorisation ; les services de support de sécurité permettant la composition sécurisée de services au niveau terminal (VPDN) et au niveau services (VPSN) ; et les services de gestion de sécurité à savoir le service d'audit qui permet de contrôler et d'analyser les dysfonctionnements relatives à la sécurité.

Dans la **Partie 2**, nous traitons l'aspect protocolaire où nous proposons un protocole SIP+ basé sur le jeton.

- Suite à une introduction de la partie dans le **Chapitre 5**, nous déterminons les besoins liés à cet aspect dans le **Chapitre 6.**
- Nous menons ensuite une analyse de l'existant dans le **Chapitre 7**. D'une part, nous discutons le mécanisme du « Single Sign On » permettant une authentification unique, et d'autre part, nous étudions les protocoles utilisés dans notre contexte durant les différentes phases de la session, à savoir SIP, HTTP et SOAP, en s'intéressant particulièrement aux aspects de sécurité.
- Le **Chapitre 8** consigne notre proposition qui adresse la sécurité de la session au niveau protocolaire. Notre contribution adresse deux axes majeurs de sécurité dans un environnement NGN/NGS. Le premier consiste à sécuriser l'accès aux services en assurant une authentification unique permettant de fournir à l'utilisateur une session de service sans couture et sans coupure en toute sécurité. Pour cela, nous décrivons d'abord notre session UBIS et ses exigences en termes de sécurité. Puis, nous décrivons le protocole SIP+, proposé par notre groupe de travail dans le cadre du projet UBIS, dans lequel notre contribution a consisté à définir le « Token-based SIP+ ».

  Le second axe de notre contribution s'oriente vers la sécurisation du protocole proposé SIP+ pour faire face aux attaques potentielles de SIP+.

La **Partie 3** permet de valoriser et montrer la faisabilité de nos propositions.

- Elle est constituée du **Chapitre 9** qui présente la plate-forme d'expérimentation et nos scénarii de tests.
- Le **Chapitre 10** décrit le modèle informationnel des différents services de securité à savoir l'identification, l'authentification et l'autorisation.

A la fin,

- Le **Chapitre 11** contient la conclusion de ce manuscrit de thèse qui résume l'ensemble de nos contributions et donne des éventuelles perspectives de recherche.

*Partie 1*

# Aspect Architectural: « Security as a Service »

# 2

# Introduction

Avec l'évolution des nouvelles générations des réseaux et des services (NGN/NGS), l'utilisateur peut facilement et constamment être connecté à ses services pendant ses déplacements, par n'importe quel réseau de sa préférence dont il possède les droits d'accès et à partir de n'importe quel terminal et n'importe quand. Il peut composer des services de nature différents (IT, Web, Telecom) fournies par différents fournisseurs de service. Dans ce nouveau contexte, la délivrance des services personnalisés à l'utilisateur est étroitement liée à une session unique et continue qui doit être sécurisée et maintenue durant la mobilité et les changements de préférences ou du contexte ambiant de l'utilisateur. Pour pouvoir répondre aux besoins de l'utilisateur moderne, il nous faudra harmoniser entre la mobilité, l'hétérogénéité de l'environnement, le contexte ambiant, les préférences de l'utilisateur ainsi que la sécurité.

La gestion de la sécurité dans un environnement mobile hétérogène et ouvert n'implique pas de réinventer la roue. En effet, les enjeux de la sécurité restent les mêmes : Il est important de préserver la confidentialité des données privées et de veiller à ce que les messages ne soient pas altérés (intégrité). Il est tout aussi important d'authentifier le demandeur, de décider du niveau d'autorisation octroyé à l'application ou à l'utilisateur demandeur et d'assurer le suivi de tout ce qui s'est passé et se passe au sein de l'environnement, d'un point de vue sécurité (audit/reporting).

Pour offrir une gestion et un contrôle de sécurité efficaces, flexibles et distribués qui répondent au mieux aux exigences de notre contexte et aux attentes de l'utilisateur, il nous a fallu réfléchir autrement à la sécurité, elle ne pouvait plus garder son image classique. Elle doit être fournie comme service en se déclinant sous forme de composants de sécurité permettant à l'utilisateur d'avoir un accès sécurisé et sans couture à l'ensemble de ses services.

Le but est de lever les verrous de la spécification des services de sécurité, de la sécurisation de l'accès aux services dans un environnement mobile et trans-organisationel, de la composition dynamique et sécurisée des services durant la session « user-centric », et du contrôle automatisé et une auto-surveillance de la sécurité durant les changements du contexte.

Nous commençons cette partie par une analyse de la sécurité dans les architectures de services existantes à savoir les web services, SOA et le Cloud (§3). Nous les avons décrites et ensuite, nous avons discuté leurs avantages et leurs inconvénients, particulièrement en termes de sécurité, afin de tirer des leçons à retenir pour appréhender ces nouveaux besoins qui régissent le nouveau contexte NGN/NGS et user-centric. Par la suite, nous proposons

notre nouvelle architecture de sécurité de services (§4). Cette dernière est basée sur le concept « Security as a Service » et sur un modèle architectural à base des composants de services mutualisables, auto-gérables, interopérables et stateless. Après, nous allons décrire les services de base de sécurité selon ce modèle et les services de support de sécurité qui permettent une composition personnalisée et sécurisée des services du côté du terminal (VPDN) et du côté des services (VPSN) lors de la session « user-centric » en tenant compte des défis provenant de l'émergence NGN. Nous proposons aussi le service d'audit basé sur la QoS qui permet de gérer et contrôler la sécurité pendant l'exécution des services.

# 3

# Analyse de la sécurité dans les architectures de service existantes

## 3.1. Introduction

Ces dernières années, une majorité des grandes entreprises dans le monde ont déjà commencé à utiliser une approche SOA/WS (architecture orientée service/services Web) ou prévoient de le faire dans un futur proche. Les architectures orientées services et les services Web semblent aujourd'hui se démarquer, pour devenir les nouvelles architectures d'applications populaires au sein des entreprises dont l'activité est centrée sur les technologies de l'information. De plus, l'émergence et la popularité croissante du déploiement du Cloud renforcent encore la tendance de l'approche SOA/WS en matière d'intégration des applications.

Toutefois, comme dans le cas de toute nouvelle architecture et en particulier celles qui sont fortement distribuées, la gestion de la sécurité peut être un problème majeur qui doit être résolue pour qu'un déploiement à grande échelle soit possible. Sans une architecture adaptée, la sécurité des applications est souvent organisée en silos. Malheureusement, ceci augmente le risque de fuite d'informations, les coûts d'administration de la sécurité et la difficulté de mise en conformité vis-à-vis des réglementations ayant une incidence sur les systèmes d'information.

En général, les solutions de sécurité conçues pour n'importe quelle architecture doivent prendre en charge les concepts suivants:

- L'authentification : Consiste à valider l'identité de l'utilisateur ou de l'interface qui compte accéder au service ;
- L'autorisation d'accès au service ;
- L'intégrité et la confidentialité pour garantir que l'échange des messages est confidentiel et intact ;
- La non-répudiation qui garantit qu'un acteur ne puisse renier la dernière opération qu'il a effectuée ;
- L'audit et la trace des évènements de sécurité dans le système.

Dans cette partie, nous étudions les solutions existantes de sécurité dans ces architectures de service à savoir les Web Services (§3.2), SOA (§0) et le Cloud (§0).

## 3.2. La sécurité et les Web services

### 3.2.1 Description de l'architecture des Web services

Le concept des « Web Services » est devenu la technologie la plus populaire et la plus connue dans le monde industriel et académique pour la mise en place d'architectures à services. Le W3C (World Wide Web Consortium) [1] définit un «Web Service » comme un système logiciel conçu pour supporter l'interopérabilité entre les différentes machines dans un réseau. Il dispose d'une interface décrite dans un format WSDL (Web Service Description Language). D'autres systèmes interagissent avec le service Web de la manière précisée par sa description en utilisant les messages SOAP (Simple Object Access Protocol), et qui sont transportés via HTTP (Hypertext Transfer Protocol) avec une sérialisation XML (eXtensible Markup Language) et avec la contribution d'autres normes et standards liés à l'approche web.

Il existe plusieurs définitions similaires d'un service Web qui adoptent un point de vue général sans préciser les standards utilisés pour la découverte, l'invocation, le transport et la publication. Par exemple, IBM définit un Web Service comme étant une application logicielle autosuffisante et auto-descriptive qui peut être localisé et appelé à travers d'un réseau, en particulier le World Wide Web et dont ses composantes décrivent ses propres entrées et sorties de façon que d'autres logiciels invoquant ce service web puissent interpréter ce qu'il fait, comment invoquer sa fonctionnalité ainsi à quel résultat peut-on s'attendre. En effet, IBM considère que les web services sont les bases permettant de construire des systèmes distribués et ouverts sur Internet, grâce à leur interface asynchrone utilisant des technologies indépendantes des plates-formes et de leurs composantes réutilisables appelées services. Pour Gartner Group, il décrit les web services comme des composants logiciels faiblement couplés qui interagissent les uns avec les autres d'une manière dynamique en utilisant les standards et les protocoles internet.

Il s'agit maintenant d'identifier les éléments sur lesquels se base une architecture Web Services et de comprendre, par la suite, comment ils interagissent entre eux. Comme le montre la Figure 3 ci-dessous, les trois acteurs les plus importants dans une architecture Web Service sont : le fournisseur de service, le demandeur de service et le registre de services [1].

- Le fournisseur du service crée le service, définit sa description et publie toutes ses caractéristiques dans un annuaire de services.

- Le demandeur de service, quant à lui, invoque une opération de recherche à l'annuaire pour trouver la description du service dont il a besoin. Il peut ainsi envoyer ses requêtes au service désiré et obtenir les réponses qu'il pourra analyser.

- Le registre de services joue le rôle d'un annuaire qui permet de rendre disponible les interfaces d'accès aux services tout en cédant le contrat et l'architecture employée d'un Web Service pour permettre les interactions entre le fournisseur et le demandeur.

Les interactions entre les différents acteurs dans une architecture Web Service se fait par l'intermédiaire des trois primitives de communications : la publication *(Publish),* la recherche et la découverte (*Find)*, et la liaison et l'invocation *(Bind).*

- *La publication de services* : les fournisseurs de services enregistrent leur service dans le registre.
- *La recherche et la découverte d'un service :* Les demandeurs de service consultent le registre pour trouver le service désiré.
- *La liaison et l'invocation de service* : Une fois le service est choisi, le demandeur peut établir une connexion avec le fournisseur et par la suite utiliser le service.

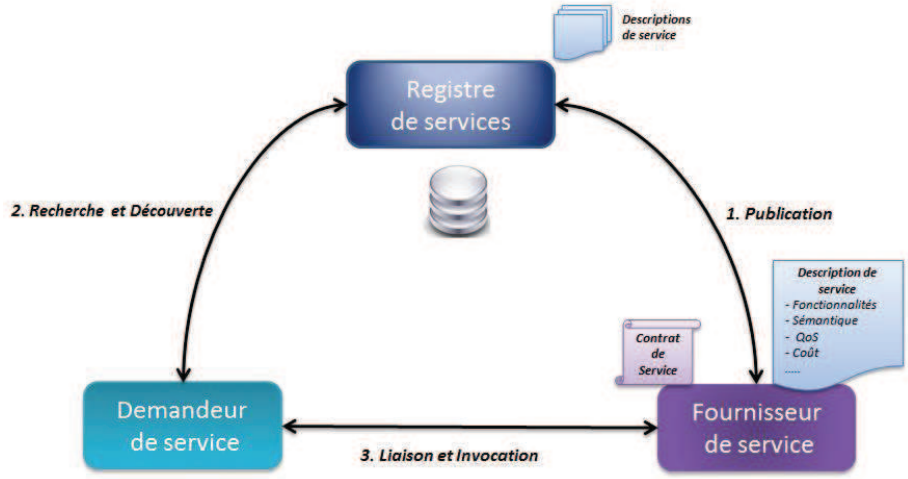



***Figure 3 :*** *Architecture Web Service.*

Cette architecture Web Service est implémentée à l'aide des diverses technologies en s'appuyant sur une infrastructure en couches représenté sur la Figure 4. Chaque couche de la pile web service fait désormais l'objet d'une normalisation. Elle répond à des préoccupations différentes telles que le transport, l'échange de messages, la description, la découverte et la publication, les processus et la sécurité.

…*La couche Transport* a pour rôle le transport des messages XML entre applications communicantes. Il existe plusieurs protocoles dans cette couche qui peuvent supporter les web services, on peut citer HTTP, SMTP et FTP.

…*La couche Message* est basée sur le standard Simple Object Access Protocol (SOAP) [2]. C'est un protocole dont la syntaxe est fondée sur XML et qui a été initialement proposé par Microsoft et IBM. Actuellement, SOAP est devenu une spécification recommandée par W3C. Il a pour objectif d'assurer la normalisation des échanges des données dans des environnements distribués. Il assure l'interopérabilité entre composants applicatifs tout en restant indépendant de la plate-forme et des langages de programmation. Il s'appuie sur les deux standards XML et HTTP respectivement pour la structure des messages et pour le transport mais il peut utiliser d'autres protocoles de communications comme SMTP et POP.

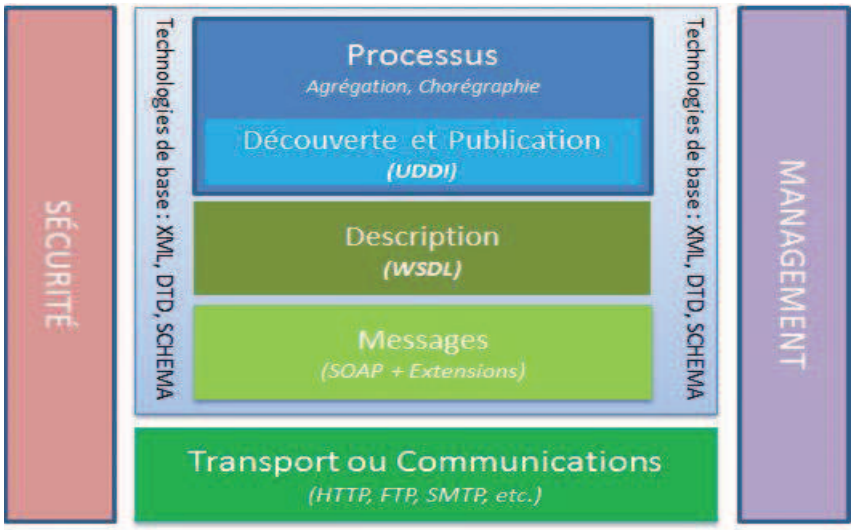



***Figure 4 :*** *Pile technologique de l'architecture Web Service.*

…*Au niveau de la couche de Description,* un service web est décrit par le langage Web Services Description Language (WSDL) [3][4]. Il s'agit d'une spécification qui définit une grammaire XML pour expliciter un service web ainsi que sa méthode d'invocation. Un document WSDL comporte seulement une description des fonctionnalités d'un service sans se préoccuper des détails techniques propres au fournisseur de service qui ne sont pas dévoilés aux demandeurs. WSDL constitue un contrat pour l'invocation d'un web service ; Il repose sur HTTP/SOAP comme mécanismes d'invocation de services distants.

La dernière couche dans la pile technologique des Web Services est celle *des Processus.* Cette couche assure la gestion des processus métier dans le cadre des Web Services qui s'étendent sur plusieurs applications disponibles sur Internet. Or pour créer un processus, il faut tout d'abord commencer par une phase de découverte des web services disponibles. Cette tâche est réalisée par la couche *Découverte et Publication qui* s'appuie généralement sur le standard Universal Description Discovery and Integration (UDDI) [5] qui assure le partage des services dans un registre commun et permet la publication, la découverte et l'enregistrement d'un service selon ses fonctionnalités ou son fournisseur. Du fait que le web service est composable, cette couche des processus gère les opérations concernant la composition telle que la chorégraphie et l'orchestration utilisant le langage WSFL (Web Services Flow Language) pour décrire comment sont effectuées les communications, les collaborations et les flux entre les services au niveau implémentation. La chorégraphie permet de gérer des processus métier complexes qui nécessitent la coordination entre différents services métier « Business Services » afin de répondre à une requête de mise en œuvre d'un Business Service.

L'apport majeur de cette architecture est d'intégrer, automatiser et de gérer les processus métiers intra et interentreprises. L'architecture Web Service s'est imposée (tout comme le langage XML) graçe à sa simplicité et ses fondations normalisées. En effet, les Web Services peuvent être implémentés sur différentes plates-formes et avec des langages variés, ce qui facilite l'accès aux applications entre entités et ainsi simplifier les échanges de données. Par conséquent, un autre apport intéressant de cette architecture apparait celui de l'interopérabilité entre systèmes et plateformes hétérogènes. Cette interopérabilité est due à l'utilisation de normes ouvertes spécifiées par l'OSI et le W3C. En plus, cette architecture

procure un faible couplage, c'est-à-dire, seulement la description de service est partagée entre les différents acteurs ; ainsi elle peut prendre différentes formes et fournir différents degrés de précision ayant comme but la spécification des fonctionnalités offertes par le service. Les Web Services sont considérés comme des composants applicatifs. Nous obtenons ainsi un autre avantage : l'hétérogénéité des implémentations et des plates-formes est masquée au demandeur de service, ainsi que la localisation du service. En fait, un service web peut être remplacé par un autre d'une manière transparente grâce à l'interface de service à condition qu'il respecte le contrat établi entre le fournisseur et le demandeur. Le succès des Web Services n'est pas un hasard. En effet, les Web Services sont caractérisés par leur déploiement facile et rapide. Autrement dit, une entreprise utilisant les Web Services peut mettre à disposition des nouveaux services en combinant d'autres services déjà existants tout en limitant les coûts ainsi que le temps nécessaire à leur déploiement. En outre, les Web Services bénéficient d'un avantage non négligeable en utilisant le protocole HTTP à savoir leur fonctionnement à travers de nombreux firewalls sans avoir à modifier les règles de filtrage.

Les Web Services présentent de nombreux avantages, mais présentent également quelques inconvénients. Les Web Services ont relativement de faibles performances quand ils sont comparés à CORBA et Java-RMI à cause de l'échange des messages qui est particulièrement lent du fait que la quantité des informations est importante. Ceci nous amène à aborder le problème de la QoS. En effet, il n'existe aucun mécanisme permettant de garantir la QoS de service web malgré qu'elle ait été rajoutée à la description du contrat d'un service Web dans la dernière version WSDL (WSDL 2.0).

### 3.2.2 La sécurité : Concepts et Standards

L'émergence des services Web pose plusieurs problématiques dont celle de la sécurité des échanges interentreprises (Business to Business ou B2B) ou entre partenaires. La gestion de la sécurité est un élément fondamental de la technologie des services Web.

Les solutions de sécurité pour les services web existent mais elles ne sont pas toutes récentes. En fait, la plupart de ces solutions reposent essentiellement sur des technologies traditionnelles et des mécanismes de base utilisés pour la sécurisation des systèmes d'information et qui n'intègrent pas nécessairement les spécificités des web services.

En effet, les web services peuvent exposer des processus métier sensibles qui nécessitent un traitement particulier en termes de sécurité aux deux bouts de canal de communication. En plus, la prolifération des sources de données, des applications et des réseaux ainsi que les différentes méthodes d'accès à ces divers éléments suscitent de nouvelles vulnérabilités, menaces et attaques. Par conséquent, tout service web doit reposer sur une stratégie de sécurité qui minimise les risques de sécurité de son infrastructure informatique et prescrit un modèle où chaque niveau sera responsable de la gestion de sa sécurité.

Pour cela, les Web Services doivent primordialement intégrer les méthodes de base de la sécurité afin de permettre une meilleure sécurité et une bonne fiabilité

De point de vue pratique, il faut, tout d'abord, commencer par sécuriser l'infrastructure informatique avant d'intégrer les aspects de sécurité spécifiques aux web services ; ensuite, rendre les communications sûres, sécuriser les données et les messages échangés et finalement assurer le contrôle d'accès aux web services en utilisant les standards existants.

Il est primordial de penser à la sécurité de l'infrastructure informatique sur laquelle repose un service web. Diverses technologies peuvent être intégrées dans un plan de sécurité global permettant à une organisation de garantir la sécurité de son infrastructure. Pour ce faire, il faut identifier d'une façon approfondie les risques potentiels liés au contexte (par exemple, virus, pirates, incendies imprévisibles, etc.). Puis, il faut analyser les conséquences des violations de la sécurité et envisager des mesures préventives. Enfin, il est nécessaire d'intégrer des mesures de sécurité sur tous les niveaux du réseau de l'entreprise en fonction de l'audit de sécurité réalisé.

Un niveau important à sécuriser dans une architecture Web Service est celui des communications ou de la couche transport. Pour cela, plusieurs moyens peuvent être envisagés. Une des solutions les plus faciles à mettre en œuvre est d'assurer la fiabilité des communications entre le demandeur (le client) et le fournisseur (le serveur). Pour atteindre cet objectif, plusieurs mécanismes peuvent être mise en place selon la portée du réseau et des interactions. On peut citer, parmi les plus répandus et les plus accessibles, les mécanismes suivants: des règles basées sur l'existence d'un pare-feu permettent de limiter l'accès à une liste d'adresses IP connues ; le protocole SSL (Secure Sockets Layer) permettant d'établir des communications sécurisées sur un réseau non sécurisé (Internet) ; et le réseau virtuel privé (VPN, Virtual Private Network) qui est une extension d'un réseau privé assurant des communications sécurisés sur des réseaux partagés ou publics comme internet.

Une fois la communication dans un environnement Web Service devient sécurisée, il faut s'assurer de la sécurité des données, plus particulièrement, des messages SOAP. Ainsi, pour assurer la confidentialité et l'intégrité, les messages seront encryptés. Pour ce faire, il existe plusieurs moyens comme le cryptage avec le standard proposé par W3C : XML Encryption Standard.

Un autre aspect essentiel et primordial à traiter pour sécuriser un service web est celui du contrôle d'accès. Un des mécanismes de base à mettre en place est l'authentification. C'est un processus qui consiste à vérifier l'identité d'une personne, d'un service ou d'une application émettant un message. L'authentification nécessite des preuves d'identité (par exemple, mot de passe). Le système vérifie ces informations et si elles sont valides, l'entité sera authentifiée.

Le jeton (ou token) peut être utilisé comme un moyen assurant le transfert des données d'authentification et qui est imbriqué dans le header des messages SOAP. Parmi les types de jeton, on peut citer: les certificats digitaux (X509 Token) où l'échange des données d'authentification se fait par certificat ; SAML Token où l'échange des données d'authentification est au format XML ; et User Name Token où l'échange de couples login/ mot de passe se fait en WS-Security. Ils existent d'autres jetons permettant l'authentification via des systèmes tiers comme le Ticket Kerberos imbriqué dans les headers SOAP.

Une autre solution parmi les plus simples pour authentifier les accès à un web service est d'utiliser les fonctionnalités d'authentification du protocole utilisé pour l'échange des messages. La plupart des Web Services exploitent les fonctions d'authentification du protocole HTTP. Pour assurer l'interopérabilité entre les Web Services, il est nécessaire que les méthodes de sécurité utilisées étendent la norme SOAP intégrant ainsi la sécurité directement dans les Web Services pour proposer des fonctionnalités de sécurité optimale.

Le deuxième mécanisme de base à mettre en place est celui de l'autorisation ou bien la gestion des droits d'accès. Il s'agit d'assurer que l'entité a le droit d'accéder qu'à la ressource à laquelle elle est autorisée.

Pour assurer tous les aspects de sécurité susmentionnés, plusieurs organisations telles que OASIS [6], W3C et Liberty Alliance, et différents acteurs majeurs de l'industrie informatique ont défini de nombreux standards et techniques pour sécuriser les Web services.

La Figure 5 illustre un modèle de référence pour les standards de sécurité des Web services. Ce modèle fait correspondre les différents standards aux différentes couches fonctionnelles d'une implémentation typique de Web service. Ces couches sont modélisées d'après le modèle OSI mais elles ne sont pas destinées à être interprétées comme étant strictement hiérarchiques.

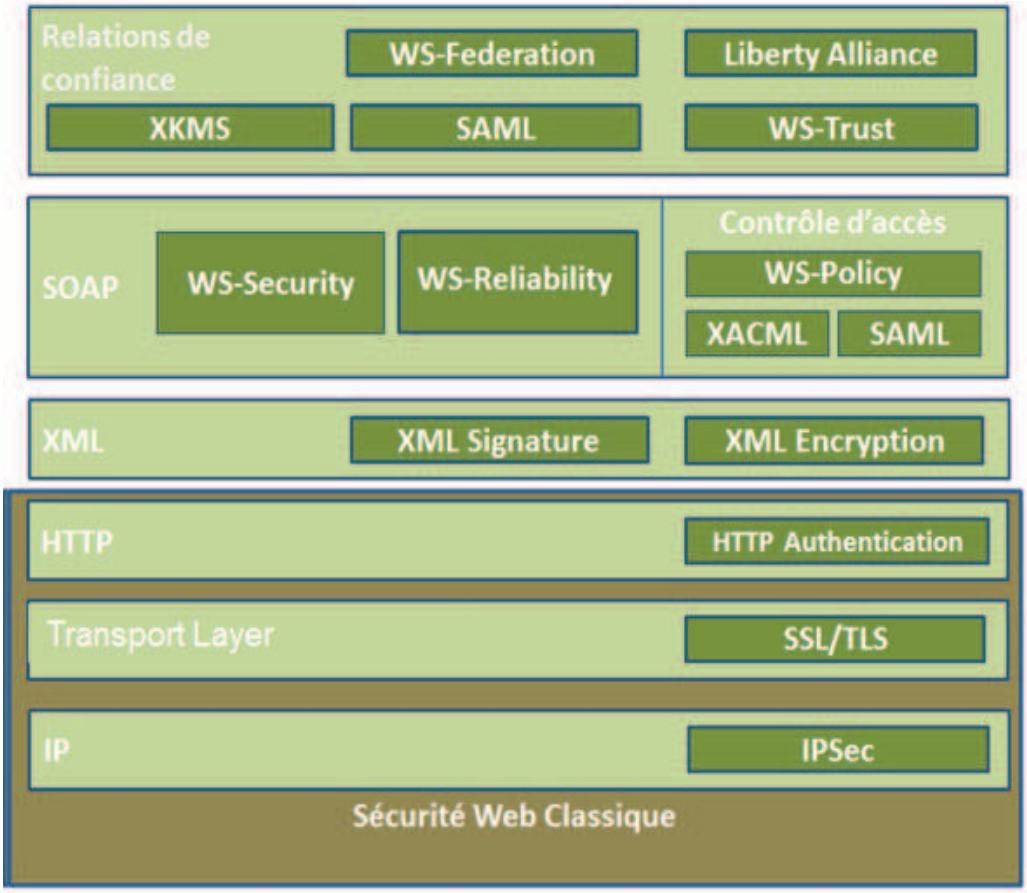



***Figure 5 :*** *Standards de sécurité Web Service.*

La sécurité des couches réseau et transport n'est pas propre aux Web Services. Les technologies disponibles et leurs propres implémentations visent plus généralement à garantir la confidentialité et l'intégrité point à point des données en transit. Dans une certaine mesure, certains de ces protocoles permettent d'effectuer l'authentification des acteurs. C'est le cas en particulier de TLS et SSL v3 parfois utilisés dans ce sens dans le cadre de Web Service.

Au niveau de la couche transport applicatif, la seule norme de sécurité apporté à ce niveau est l'authentification HTTP qui peut être implémentée de deux manières :

l'authentification *Basic* et l'authentification par *Digest*. Quelle que soit la technique utilisée, le niveau de sécurité est considéré relativement faible (voire très faible pour l'authentification). Par conséquent, lorsque ces techniques sont implémentées, elles sont généralement au-dessus d'une couche réseau ou transport sécurisée.

Pour la couche de présentation, deux mécanismes de sécurité ont été définis : XML Signature [7] et XML Encryption [8] sont deux recommandations de W3C qui fournissent respectivement la capacité de signer et de chiffrer des données XML. La particularité de ces deux normes réside dans le fait qu'elles permettent de signer et/ou de chiffrer uniquement certaines parties d'un document XML.

Pour la couche SOAP, la plupart des implémentations de sécurité actuelles se limitent à la protection des messages SOAP. En effet, le niveau de sécurité fourni par les normes à ce niveau est un compromis acceptable entre la maturité des technologies, l'impact structurel sur les infrastructures existantes et la protection contre les principales menaces. Deux normes majeures ont été définies à ce niveau : WS-Security [9] et WS-Reliability [10]. Ce dernier a essentiellement pour objectif de garantir la qualité et la disponibilité des canaux de communications. WS-Security quant à elle, intègre XML Signature et XML Encryption et leur rajoute un certain nombre de fonctions essentiellement orientées vers la gestion des jetons. Le contrôle d'accès est assuré à travers les trois standards SAML [11], XACML [12] et WS-Policy [13]. En effet, XACML autorise ou interdit l'accès aux ressources. L'assertion SAML est ensuite créée et traitée en fonction des critères établis par le Web Service. Enfin, WS-Policy est une spécification cadre qui permet de définir les spécifications nécessaires à la communication avec un Web Service. Cette spécification couvre trois domaines à savoir la sécurité, l'adressage et la fiabilité des communications (reliable messaging). Il existe une autre norme dénommée WS-Security Policy [14] qui est une déclinaison de WS-Policy qui définit les politiques de sécurité. Ainsi, elle établit le format de plusieurs types d'assertions concernant la confidentialité, l'intégrité et les jetons de sécurité.

Pour finir, la nature des web service impose la définition et la mise en place de mécanismes permettant d'établir une chaine de confiance entre les différents acteurs.

Dans le cas des Web Services impliquant un nombre réduit d'acteurs dépendant de la même autorité, il est possible de se reposer sur des mécanismes basiques tels que des relations « un pour un ». Ce schéma induit que tout acteur dispose d'une clé publique de tous les autres. Dans ce cadre, la norme XKMS est conçue pour la gestion des clés. Il existe d'autres normes qui sont définies également pour permettre la mise en œuvre de telles relations entre les acteurs : Liberty Alliance d'un côté et WS-Trust [15] et WS-Federation [16] de l'autre côté. WS-Trust se repose sur WS-Security Policy pour la définition des jetons nécessaires à la communication entre les Web Service. WS-Federation quant à elle, définit les domaines de confiance et la manière dont les demandeurs et les fournisseurs interagissent ensemble.

### 3.2.3 Synthèse

La sécurité dans les architectures Web Services est fournie selon un modèle en silo où chaque couche est responsable de sa sécurité tout en se basant sur des standards ouverts d'Internet. En termes de sécurité, les normes ne sont pas complètement établies. Les standards qui existent peuvent ne pas être utilisés ou assurer seulement une partie des exigences de sécurité. En plus, le fait que les Web Services utilisent le protocole HTTP pourrait également être considéré comme un inconvénient puisque les firewalls peuvent être

contournés. Donc, HTTP permet de passer au travers des règles de sécurité des Web Services.

La sécurité, telle qu'elle est proposée par les Web services, pourrait-elle apporter des réponses aux nouveaux besoins de notre contexte mobile, hétérogène et trans-organisationnel ?

Les Services Web permettent la création de services demandés par l'utilisateur grâce à des composants proposés à travers le Web d'une manière statique et monolithique en se basant sur le Register Service qui est également une application client-serveur peu flexible. Par conséquent, il est difficile d'ajouter ou de modifier dynamiquement un composant de service tout en gardant le même niveau de sécurité requis. Le Web Service contribue à la simplification et à la réutilisation de service mais il ne peut pas s'adapter au contexte ambiant de l'utilisateur tout en tenant compte de ses préférences et puis sa mobilité. Il nous faut des composants qui ont des capacités de réutilisation et de mutualisation à la fois pour répondre aux besoins de l'utilisateur en termes de sécurité et de continuité de services malgré la mobilité et l'hétérogénéité.

## 3.3. La sécurité et SOA

### 3.3.1 Concepts de base : Définition, Avantages, Critères d'un service

L'évolution de l'informatique depuis de nombreuses années a amené les organisations à développer des systèmes d'information de plus en plus hétérogènes. Ces organisations, bien qu'elles communiquent entre elles, n'utilisent pas forcément les mêmes technologies. Les exigences de ces organisations se reflètent essentiellement au travers des termes de réutilisation, facilité d'échange de messages, flexibilité, adaptation dynamique et rapide, disponibilité et sécurité.

Les architectures orientées services semblent être une des solutions possibles envisagées pour répondre à ces exigences, bien qu'il soit important de prévoir si cette mouvance se traduira par une évolution de l'existant, ou elle engendrera plutôt une refonte majeure des architectures déjà mises en place. Pour cela, on doit tout d'abord comprendre ce que signifie ce concept d'orienté service.

SOA est une nouvelle architecture orientée service qui a pour but de répondre au mieux aux nouvelles exigences des entreprises tel que le ROI (Return On Investment) et le TTM (Time To Market) s'appuyant sur un ensemble de services simples.

L'idée principale est de mettre en œuvre les services ou les applications sous forme de composants logiciels. Ces composants doivent absolument être autonomes, interopérables et réutilisables. La composition de service basée sur la réutilisation des mêmes blocks fonctionnels (les composants logiciels), lors de la création des différentes applications, permet une création plus rapide et plus facile de ces applications et répond ainsi aux exigences du marché actuel.

Dans la littérature, il existe plusieurs définitions de l'architecture orientée service, on peut citer quelques-unes :

Gartner Group définit SOA comme suit: «A service-oriented architecture is a style of multitier computing that helps organizations share logic and data among multiple applications and usage modes »;

Celle publiée par le Consortium OASIS: « Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations»;

Celle donnée par Microsoft: «The policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface».

En s'appuyant sur ces définitions, on peut dire que SOA décrit une façon de bâtir des solutions à base de services. Ce style d'architecture est une approche évolutive qui contribue à mettre en place un système d'information flexible, assurant ainsi une interopérabilité intrinsèque et une meilleure agilité pour l'entreprise.

L'architecture Orientée Service est bien évidemment une approche architecturale permettant l'intégration des fonctions qui incluent des applications d'entreprise comme des services interopérables impliquant ainsi la création d'une nouvelle génération d'applications. C'est une méthode de conception basée sur des standards qui fournit des directives et des principes permettant de transformer un réseau existant de ressources informatiques hétérogènes, distribuées, complexes et rigides en ressources intégrées, simplifiées et particulièrement souples pouvant être modifiées et combinées afin de mieux répondre aux besoins de l'entreprise. Par conséquent, les processus métiers des entreprises sont structurés selon une approche basée sur le principe de services distribués qui fonctionnent indépendamment les uns des autres afin de réaliser une fonctionnalité globale.

D'un point de vue architectural, SOA propose une organisation capable de gérer les interactions entre divers acteurs. Cette organisation comprend, comme illustré dans la Figure 6, deux catégories de services:

La première catégorie est celle des services (mécanismes) de base qui assurent la publication, la découverte, la composition, la contractualisation, la négociation et l'invocation des services.

La deuxième catégorie décrit les services additionnels assurant la prise en charge des mécanismes non-fonctionnels tels que la sécurité, la gestion des transactions ou encore la qualité de services.

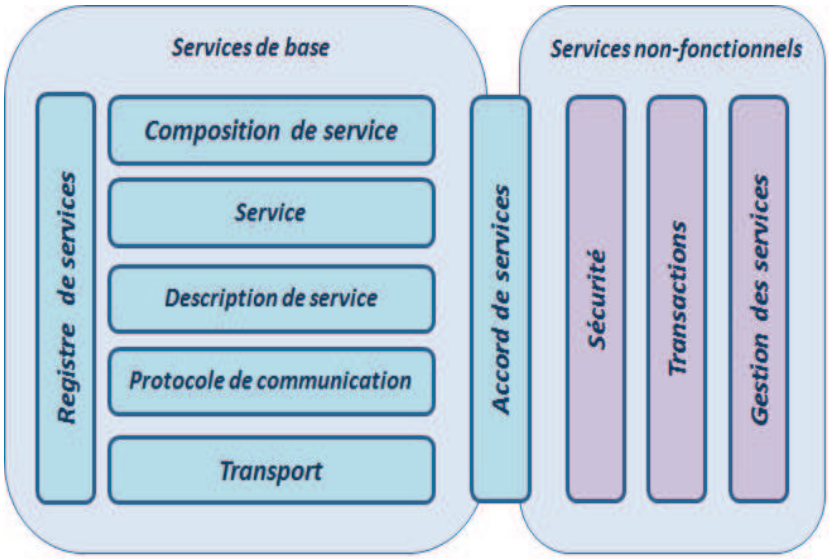



***Figure 6 :*** *Modèle architecturale SOA.*

Les moyens de transport et le protocole de communication sont la base d'un environnement orienté service. Ils assurent les interactions et les communications entre les différentes entités de l'architecture. Il faut aussi avoir une description bien précise du service décrivant ses fonctionnalités, ses éléments non fonctionnels ainsi que la façon dont il doit être invoqué. Pour faciliter la recherche d'un service qui répond aux besoins du demandeur et pour permettre au fournisseur l'enregistrement de ses services, un mécanisme à mettre en place semble être nécessaire : c'est le registre des services.

L'accord de service fait partie des mécanismes de base mais aussi de la partie non fonctionnelle. Il représente le contrat entre le fournisseur et le demandeur. Ce contrat spécifie non seulement les fonctionnalités de service mais aussi les aspects non fonctionnels ou contractuels comme par exemple le temps de réponse ou le degré de fiabilité que le service doit respecter.

Les autres éléments non-fonctionnels sont nécessaires dans un contexte SOA ; ils servent de base à l'établissement d'un contrat d'utilisation du service.

– la Sécurité permet, par exemple, au fournisseur de service de sécuriser l'accès à son service.

– la Transaction permet, au cas où divers services sont utilisés en collaboration, d'assurer la cohérence des données.

– la Gestion de services se charge de l'administration des ressources et de leur utilisation au sein de la plate-forme afin de fournir un bon fonctionnement des applications et garantir la qualité de service.

D'un point de vue technologique, le paradigme SOA n'est pas étroitement lié à une technologie ou à une mouvance particulière. Il est bien possible d'implémenter des projets respectant l'approche SOA aussi bien sous une technologie Microsoft qu'avec des solutions open source J2EE ou autres.

L'adoption de SOA a été grandement facilitée par l'émergence opportune de la technologie des Web Service et leurs standards bien définis. En fait, les acteurs qui se sont

positionnés autour des Web service et qui proposent des solutions matures à ce niveau sont ceux les plus impliqués à la contribution et l'investissement dans cette approche. Toutefois, il est possible d'envisager la mise en place d'une architecture orientée services sans utiliser les services web malgré que ces derniers sont particulièrement adaptés à ce type d'architecture et ils ont fortement contribué à son avènement.

Comme son nom l'indique, l'approche orientée service considère le concept de service comme élément clé et fondamental pour définir et développer des applications. Un service est basé sur des opérations qui offrent un ensemble d'actions spécifiques utilisables au sein dans un environnement SOA. En effet, ces opérations fournissent une ou plusieurs réponses à une ou plusieurs requêtes. Chaque service doit être autonome indépendant du contexte ou d'autres services externes afin de garantir sa réutilisabilité et son interopérabilité dans le système d'information.

Les applications, dans un contexte SOA, sont construites à partir des services basés sur des standards ouvert, et qui peuvent simplement être réutilisés pour développer des nouvelles applications ou répondre à des nouvelles exigences client.

En fait, un environnement SOA intègre des services, des composants et des processus business. Un composant fonctionne comme un service et chaque processus business est composé par un ensemble de services, et le processus lui-même est aussi considéré comme un service.

Ainsi, une architecture orientée service consiste essentiellement en une collection de services qui interagissent et communiquent entre eux. Cette communication peut consister en une simple réponse, en une interrogation ou en une activité plus complexe.

Une des techniques de base sur laquelle repose SOA pour assurer les communications entre les services est la transmission des messages. Ce n'est pas une technique nouvelle, elle est déjà utilisée par plusieurs middlewares depuis longtemps. SOA à adopté cette technique et l'a rendu plus spécifique à ses besoins de sorte qu'après l'émission d'un message, le service n'a plus le droit de le contrôler ; il ne sait pas à quel service ce message va être envoyé ou via quelle façon à savoir synchrone ou asynchrone. De cette manière, SOA réalise le couplage faible de service, car toutes les communications sont indépendantes.

En effet, SOA vise à transformer le système d'information en une plate-forme gigantesque de services faiblement couplés et automatiquement intégrables pour gérer la communication au sein de l'entreprise.

Afin de construire une solution logicielle basée sur SOA, nous avons besoin de respecter les principes du paradigme orienté service [17]:

- *Description formelle et contrat* : La fonctionnalité d'un service est décrite formellement dans une description de service. Un contrat de service décrit et régit la relation entre le fournisseur de service et chaque demandeur de service. Il facilite l'interaction entre les services en indiquant : l'interface service, ses caractéristiques, la liste des opérations et chaque message d'entrée/sortie supporté par chaque opération. Ce contrat se réfère à une description de service et précise notamment les règles et les exigences non fonctionnels de l'utilisation du service.

- *Couplage lâche* : Ceci signifie que les services sont faiblement liés. Ce qui implique qu'un service peut changer de localisation ou d'implémentation ou de fournisseur sans générer d'impact sur les autres entités de l'architecture. En outre, le service est conçu de façon que la dépendance vis-à-vis le demandeur ou le fournisseur de service soit minimale. Un service se connait seulement par sa propre description partagée entre le demandeur et le fournisseur qui contient seulement un ensemble d'informations nécessaires et utile à l'utilisation du service comme le nom de service et les paramètres demandés par ce service. Le couplage faible offre une meilleure souplesse en termes de réutilisation et de recombinaison de services pour créer de nouvelles fonctions entre les applications. Cette propriété permet aux compositions de services d'avoir une plus grande flexibilité dans le choix des services ainsi que dans l'ordonnancement de leur composition.

- *Réutilisabilité*: Un service peut donc être utilisé parallèlement par divers consommateur de service dans un contexte de différents processus métier. Ceci revient à distribuer la logique de l'application à travers des services de manière que chaque service puisse potentiellement être utilisé par plus qu'un demandeur. La réutilisabilité répond à l'évolution du logique métier et permet de réduire les charges de développement.

- *Stateless* (sans état) : Un service *stateless* ne doit rien savoir du processus métier dans lequel il est intégré. Cela facilite le couplage lâche ainsi que sa réutilisation. Il ne conserve d'informations ni d'états ni sur les communications avec ses demandeurs. En plus, la gestion d'information sur les états est une tâche complexe et demande beaucoup de ressources ce qui complique massivement la modularité ainsi que la disponibilité de service. Pour cette raison, le service doit déléguer la gestion des informations d'état à un tiers.

- *Abstraction* : Elle met l'accent sur la nécessité de cacher les détails d'un service. Un consommateur n'a besoin que de connaitre la description du service et de la définition de ses interfaces pour pouvoir l'utiliser. En effet, l'interface de service fait l'abstraction et apparait comme une « black box » qui cache la logique sous-jacente. L'abstraction est une propriété nécessaire pour établir et préserver le couplage lâche.

- *Interopérabilité* **:** Elle est fondamentale pour toutes les autres propriétés. Elle est facilement réalisée vu que les services interagissent et collaborent entre eux à travers les interfaces indépendamment des plateformes et de langages d'exécution.

- *Découverte* : Un service est conçu, décrit et classé de façon à ce que l'on puisse le retrouver. La découverte repose sur un mécanisme qui permet à un utilisateur de trouver les services selon ses exigences et répondant à ses besoins, il s'agit de l'annuaire. Ce dernier contient les descriptions et les contrats de service.

- *Autonomie :* Elle est traduite par le contrôle maximal qu'un service exerce sur sa logique métier. Plus le service a du contrôle sur son environnement, plus son comportement au cours de l'exécution sera prévisible. Pour atteindre un niveau élevé d'autonomie, il faut tout d'abord réduire l'accès partagé aux ressources d'un service et ensuite augmenter le niveau d'isolation physique des services. L'autonomie est une propriété qui exige que la logique sous-jacente au service ainsi que les ressources qu'il utilise soient une propriété privée de service et sous son contrôle exclusif. Cependant, l'autonomie au niveau service

définit les frontières entre les services de façon que ces derniers soient indépendants les uns des autres, tout en ayant la possibilité du partage des ressources sous-jacentes.

- *Composition :* Un service composé est un nouveau concept basé sur l'interaction avec d'autres services élémentaires ou composés afin d'utiliser leurs fonctions pour réaliser des taches plus complexes. Un processus métier peut utiliser un ou plusieurs services pour accomplir une tâche. Des fonctions relativement complexes peuvent être réalisées par la composition de services simples dans divers processus métier et ceci va servir ainsi à la réutilisation.

### 3.3.2 Le défi de SOA : La Sécurité

La sécurité est un besoin crucial dans les architectures SOA qui doit poursuivre leur évolution. Elle représente une exigence transversale dans les environnements SOA qui couvre toute l'architecture et concerne tous les niveaux (Figure 7). Elle englobe tous les aspects du cycle de vie SOA.

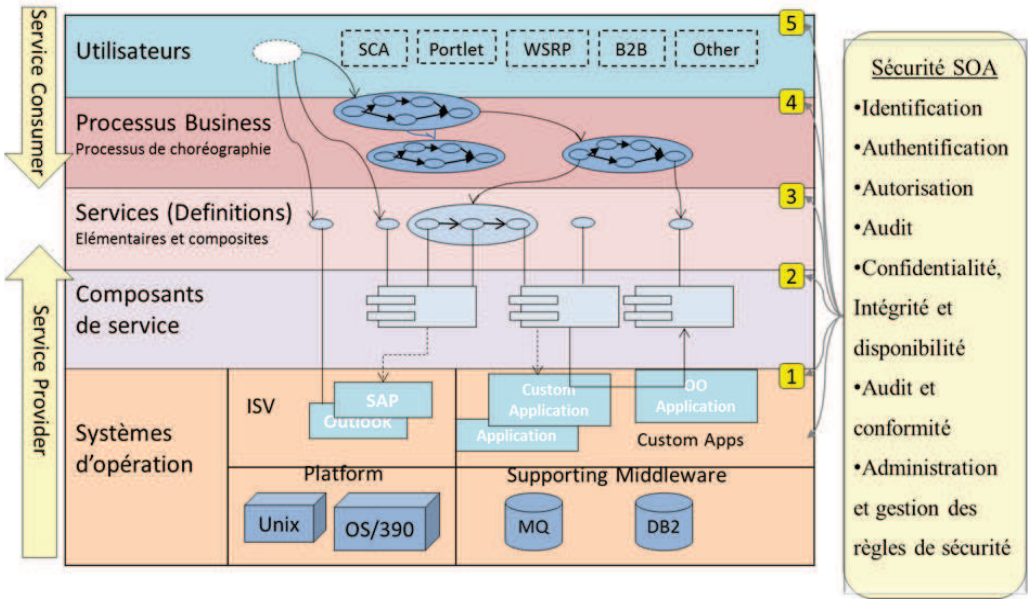



***Figure 7 :*** *La sécurité dans SOA.*

Il est clair que les approches traditionnelles des divers aspects de sécurité ne suffisaient pas. SOA n'altère pas les besoins fondamentaux de sécurité qui doivent être adressés par les applications à l'échelle d'une entreprise ou à une échelle plus ouverte et plus étendue (par exemple le Web). Mais, la manière dont la sécurité est effectuée doit être adaptée à cette architecture SOA. Ceci concerne les aspects architecturaux du système et sa conception ainsi que les technologies de sécurité adoptées. Alors, SOA nous mène à repenser la sécurité et elle permet de proposer quelques nouvelles approches. Nous décrivons, par la suite, trois de ces nouvelles approches de sécurité à savoir « Message-level Security », « Security as a service » et « Policy- driven security » [18].

Tout d'abord, nous allons commencer par la description de la sécurité au niveau messages. En effet, SOA apporte des changements dans les exigences relatives à la confidentialité et l'intégrité des données. Par exemple, lorsqu'un message envoyé à la partie 1 (courtier) contient des éléments destinés à la partie 2 (banque), nous avons besoin de la

capacité de chiffrer et / ou signer la partie destiné à être utilisé uniquement par la partie 2. De toute évidence, les mécanismes existants de la sécurité au niveau de la couche de transport à savoir SSL / TLS ne sont pas efficaces, puisqu' ils ne peuvent pas empêcher la partie 1 de la lecture (l'interception) et / ou l'altération de la partie du message destiné à la partie 2. Ainsi, la sécurité au niveau message (par opposition à la sécurité de niveau transport) est une nouvelle approche pour résoudre ce problème. Avec cette approche, les différentes parties d'un message peuvent être protégés différemment et indépendamment, et donc utilisables uniquement par les parties (récepteurs) auxquelles les messages sont destinés.

La deuxième approche est celle de la « Security as a service ». En fait, la solution de sécurité pour SOA doit être conforme aux principes de base de cette architecture, elle-même devrait être fournie en tant que service. En effet, les applications ne doivent plus être en charge de l'authentification et l'autorisation, car ils risquent de ne pas connaître le contexte dans lequel elles sont utilisées. La question qui se pose est de savoir qui peut alors se charger de l'authentification et de l'autorisation ? La réponse, en restant en accord avec la vision SOA, est d'avoir des services de sécurité.

Un service de sécurité offre aux applications la possibilité d'authentifier, d'autoriser, crypter / décrypter des messages, signer / vérifier les signatures des messages et collecter les logs liés aux messages. Il peut aussi filtrer les messages pour protéger les applications contre les vulnérabilités connues et inconnues.

L'idée d'un service de sécurité est en quelque sorte similaire à celle d'un service applicatif. Au même titre que le service applicatif, le service de sécurité devrait être utilisable par n'importe quelle application ; la technologie ne devrait pas constituer un obstacle. Contrairement à un service applicatif, un service de sécurité est lié à l'infrastructure et peut entrer en jeu même si il n'est pas explicitement invoqué. Par exemple, le service de sécurité peut être implémenté comme étant un service dans le ESB.

Un service de sécurité est indépendant et interopérable avec les autres services, et ne fait pas partie d'aucune application. Ce modèle de sécurité peut évoluer en s'alignant avec les besoins des entreprises, sans affecter les applications. Traiter la sécurité comme un service aide les développeurs de services au sein des entreprises de se concentrer essentiellement sur les services business métier sans faire des efforts pour intégrer la sécurité dans les applications.

La mise en œuvre de la sécurité en tant que service n'était pas possible jusqu'à récemment en raison d'absence de normes et technologies de communication avec un service de sécurité. Tant que LDAP permet aux applications de s'appuyer sur un serveur d'annuaire central pour l'authentification des utilisateurs et la récupération des attributs de l'utilisateur, les applications doivent encore mettre en œuvre leur propre logique d'autorisation et d'autres aspects de sécurité. SOA supporte des normes telles que Security Assertion Markup Language (SAML) et WS-Trust qui peut être utilisé pour mettre en œuvre un tel service.

Pour finir nous allons décrire l'émergence d'une troisième approche capable de rendre la sécurité de SOA facile à développer et à gérer. Cette approche adresse des besoins non fonctionnels de la sécurité pour les architectures SOA comme l'interopérabilité et la facilité de gestion et de développement. Il s'agit du « Policy-driven security » qui est une nouvelle approche axée sur les politiques de sécurité. L'idée derrière cette approche est simple. Les

exigences et les mécanismes de sécurité ne doivent pas être encastrés dans les applications. Plutôt, les exigences de sécurité d'une entreprise doivent être déclarées séparément en tant que «politique de sécurité».

Une déclaration de politique de sécurité devient utile à bien des égards: elle sépare la logique de sécurité du logique métier, laissant la première à des spécialistes de sécurité. Il devient plus facile d'assurer la consistance de la sécurité à travers de multiples applications. De plus, l'interopérabilité est améliorée puisque les politiques des parties impliquées peuvent être comparées afin de savoir comment rendre leurs implémentations de sécurité compatibles.

Après la description de ces majeures approches émergentes de sécurité dans SOA, nous allons maintenant exposer quelques travaux de recherche visant à fournir de nouvelles propositions conceptuelles de la sécurité pour les Architectures Orientées Services. Ces solutions proposées peuvent en quelque sorte recourir aux approches susmentionnées.

Face à la migration des architectures SOA de l'échelle d'une entreprise à une échelle plus étendue et plus globale destinée aux utilisateurs finaux, Shah et Patel [19] étudient les exigences de sécurité pour une telle architecture dénommée « Global SOA » et proposent une architecture ubiquitaire et dynamique de sécurité. Leur stratégie se base sur l'utilisation des intercepteurs (Handlers) qui permettent de modifier les messages (requêtes et réponses) SOAP comme, par exemple, le cryptage et le décryptage des donnés au sein du corps du message SOAP. L'avantage de cette technique est de rendre la sécurité indépendante des fonctionnalités business. Les demandeurs de services configurent alors les intercepteurs (intercepteur de cryptage, intercepteur de signature numérique, intercepteur d'authentification…), d'une manière dynamique et sans aucune pré-configuration, au côté client selon les exigences de sécurité au côté fournisseur de service.

Boehm et al [20] analysent la conception et la réalisation de la sécurité des architectures orientées-service d'une manière adaptée aux principes de SOA. De point de vue architectural, ils approuvent l'externalisation des fonctionnalités de sécurité sous la forme des services de sécurité dédiés. Ceci s'accorde avec le paradigme « orienté-service » et améliore la flexibilité et l'adaptabilité de l'architecture étant donné que la sécurité peut évoluer indépendamment et les services de sécurité peuvent être réutilisés parmi de multiples services business. Ils traitent aussi le couplage rigide entre l'authentification et l'autorisation dont souffrent les architectures de sécurité traditionnelles et qui s'oppose au besoin d'un couplage lâche dans les systèmes SOA. Pour des raisons de flexibilité et facilité d'utilisation, ils soulignent l'utilité de la fédération d'identité et le « single sign-on ».Ils adressent particulièrement l'authentification et l'autorisation fédérées basées sur les technologies de sécurité Web services.

Qi-rui et al. [21] proposent une solution unifiée d'authentification et d'autorisation qui supporte les systèmes distribués basés sur SOA. Cette proposition est basée sur l'architecture « Service-Access-Agent ». Elle utilise un mécanisme d'authentification à deux niveaux afin de séparer l'authentification intra-domaine et inter-domaine pour des raisons d'efficacité et pour une configuration simple. Le framework général de cette solution se compose du GAAC (Global Authentication and Authorization Centre), SAA (Service Access Agent) et les Web services. Le GAAC assure l'authentification inter-domaine en authentifiant les SAAs et procure un contrôle centralisé d'autorisation d'accès aux Web services. Le SAA est responsable du contrôle et de la gestion des Web services dans le domaine local de

sécurité. Cette solution assure l'authentification et l'autorisation au niveau service. Elle présente une contribution intéressante mais l'approche centralisée adoptée est monolithique, verticale et à couplage fort. *Ceci soulève les questions suivantes : Est-il suffisant dans un contexte ubiquitaire et cross-organisationnel d'avoir une telle solution ? Comment peut-on avoir plus de flexibilité, dynamicité et extensibilité dans les fonctions de sécurité ?*

E. Bertino et al. [22] proposent un cadre de référence architectural basé sur les services de sécurité comme les services d'identification, d'authentification et d'autorisation. Ils traitent le fait que chaque composant du pipeline de sécurité est considéré comme un service selon les principes de SOA. Ils traitent aussi les mécanismes de coordination entre différents services de sécurité. Alors, une approche basée sur l'événement est utilisée pour disséminer les événements pertinents de sécurité à tous les services potentiellement affectés. Pour ce faire, un service de notification est intégré dans le framework architectural afin de notifier les services de sécurité des évènements pertinents. Ce travail de recherche traite l'approche de sécurité comme service qui est particulièrement prometteuse pour les architectures orientées service. Cependant, il ne spécifie pas comment les fonctions de sécurité pourront être assurées d'une façon transparente et simplifiée pour l'utilisateur avec une composition de service dynamique et orientée utilisateur. *Ceci soulève la question : Comment peut-on fournir une session sécurisée et sans couture dans un environnement dynamique en respectant les exigences et les préférences de l'utilisateur ?*

L'architecture de sécurité orientée service décrite dans [23] et [24] consiste en trois couches. La première couche contient les interfaces de service bien définies et stables qui sont utilisées par les autres services. L'interface d'authentification fournit des opérations pour authentifier l'utilisateur (saisissant son identifiant et son mot de passe) et génère un jeton de sécurité temporaire. La décision du contrôle d'accès peut être déléguée à l'interface de service de vérification de l'autorisation. Au niveau de la couche fonctionnelle, le service de jeton sécurisé valide l'identité prétendue du consommateur du service. Un « Policy Decision Point » (PDP) évalue les politiques et prennent la décision d'autorisation. Ces composants fonctionnels sauvegardent et récupèrent leurs données à partir des composants placés au niveau de la couche de données en dessous. Ce travail met l'accent seulement sur le critère d'interopérabilité tandis que plusieurs autres critères doivent être spécifiés afin de satisfaire les nouvelles approches comme l'approche orientée-utilisateur qui exige la personnalisation, la dynamicité et la transparence dans une session sécurisée et sans couture.

### 3.3.3 Synthèse

Selon SOA, la couche service est semblable à un grand pool de services avec des composants de différents fonctions. Cette approche apporte plus de flexibilité et permet aux fournisseurs de service de répondre plus rapidement aux demandes des utilisateurs. Pourraient-elles, les solutions actuelles, assurer réellement la flexibilité si elles gardent la contrainte verticale entre le client et le serveur et si nous voulons prendre en considération les effets de mobilité ? Nous pensons que non car il manque quelques critères comme la mutualisation et l'autogestion pour avoir une flexibilité et dynamicité effectives. En fait, la mutualisation consiste à avoir non seulement des composants réutilisables mais aussi partageables dans différents contextes de différents utilisateurs. Alors, les composants de sécurité doivent être mutualisables pour être utilisés dans tous les contextes par plusieurs utilisateurs sans contraintes. L'approche courante de SOA ne spécifie pas aussi le critère d'autogestion des composants de service qui permet d'avoir une session sécurisée sans

couture avec le niveau désiré de sécurité et de QoS en dépit de la mobilité et des changements grâce à une composition dynamique de service.

S'ajoute à ceci que la composition de service seulement au niveau de la couche service n'est pas suffisante. De nos jours, avec la présence des réseaux et des terminaux hétérogènes, qui peuvent affecter considérablement l'expérience utilisateur, nous devons opter pour une nouvelle perception capable d'homogénéiser les différentes ressources ; les composants de la couche réseau ainsi que la couche terminal doivent être perçus comme des services. Ainsi, les composants de sécurité doivent considérer le terminal comme un service.

## 3.4. La sécurité et le Cloud

### 3.4.1 Description de l'architecture du Cloud

Depuis les années 80, nous assistons au développement de la virtualisation, de l'infogérance et de l'externalisation. La recherche de plus de souplesse impacte le marché qui est devenu extrêmement fluctuant. Cela contribue de façon significative à l'alignement entre la demande du client et l'organisation des ressources informatiques.

En plus, l'évolution d'internet, le développement des réseaux haut débit, la location des applications sur demande et le paiement à l'usage ont suscité l'apparition d'un nouveau concept dénommé Cloud. Ce dernier regroupe plusieurs technologies servant à délivrer différents services. Ce concept consiste notamment en une interconnexion et une coopération d'un ensemble de ressources informatiques dans un réseau bien défini, et dont le mode d'accès est basé sur les protocoles d'Internet. L'accès aux services se fait à la demande par une application standard facilement disponible à savoir, un navigateur Web. En effet, le Cloud représente le passage de l'informatique vers Internet. Tous les utilisateurs (entreprise, client final, fournisseur) peuvent accéder et exploiter de nombreux services en ligne sans avoir à gérer l'infrastructure sous-jacente souvent complexe.

En conséquence, le Cloud est devenu le sujet le plus débattu (buzzword) dans le secteur des technologies de l'information. Il semble néanmoins qu'un consensus se dégage sur le fait que le Cloud jouera un rôle de plus en plus important dans les opérations informatiques dans les entreprises au cours des années à venir.

Dans la littérature, il existe de très nombreuses définitions du Cloud. Nous pouvons citer quelques-unes.

Quand on parle de Cloud, Cisco dit que « les ressources informatiques et les services sont abstraits de l'infrastructure sous-jacente et sont fournies à la demande et à l'échelle dans un environnement partagé (multi tenant) ». A l'échelle indique que le service donne l'illusion d'une disponibilité infinie des ressources afin de répondre à toutes les demandes.

Selon Microsoft : « Le Cloud désigne l'ensemble des disciplines, technologies et modèles commerciaux utilisés pour délivrer des capacités informatiques (logiciels, plates-formes, matériel), comme un service à la demande ».

Une autre approche très intéressante est celle du NIST (National Institute of Standards Technology) qui a développé sa propre définition des services de Cloud :

« C'est un modèle d'accès à des ressources informatiques partagées et configurables (par exemple, réseaux, serveurs, stockage, applications et services), depuis un accès

réseau, à la demande, de manière simple, à partir de n'importe quel type d'appareil et depuis n'importe quel endroit. Ces ressources sont disponibles rapidement et opérationnelles avec un minimum d'efforts et d'interactions avec le fournisseur de services ».

Nous avons l'habitude de présenter le Cloud comme un mode de traitement de données d'un client, dont l'exploitation s'effectue par internet, sous la forme de services fournis par un prestataire. Il s'agit d'un mode de consommation où une ressource informatique (logiciel, plateforme de développement, infrastructure, etc.) est proposée en tant que service sans que les clients aient besoin de se soucier de sa gestion ni de sa localisation. Cette ressource peut être physiquement localisée sur une infrastructure informatique interne, externe, voire hybride. Les ressources dans le Cloud sont mises à disposition sur la base d'un abonnement contractuel et fournis depuis Internet, dont le périmètre peut être revu à la hausse ou à la baisse en fonction du besoin. Pour finir, on peut dire que le Cloud est un modèle de délivrance de services qui repose essentiellement sur les technologies d'internet et de virtualisation. Il offre des services à la place des produits technologiques avec une mise à jour continue et automatique. Le Cloud comporte des caractéristiques essentielles et clés classés selon deux points de vue différentes mais complémentaires.

Les caractéristiques d'un point de vue utilisateur sont :

- Self-Service à la demande et le payement à l'usage : l'utilisateur peut faire du provisionning de la capacité du Cloud (Serveur, Storage, Mémoire …) de manière automatique sans aucune interaction avec le fournisseur de service.

- Accès ubiquitaire aux services : l'accès aux services est disponible sur internet de n'importe quelle plate-forme (PDAs, téléphone mobile, portables, …) grâce à un large accès réseau.

- Services mesurés : les services sont tracés dans du Cloud. Ce dernier contrôle l'utilisation des ressources (services) et leur consommation en s'appuyant sur les métriques qui permettent différents modèles de facturation, tout en assurant la transparence pour l'utilisateur et le fournisseur de services.

Les caractéristiques d'un point de vue fournisseur sont :

- Elasticité: Cette propriété permet de libérer et/ou réserver les ressources (physiques ou virtuelles) en fonction de la demande et des besoins. Donc, l'allocation des capacités est dynamique dans le Cloud.

- Mise en commun des ressources: le fournisseur offre des services et des ressources qui peuvent être sollicités par plusieurs utilisateurs grâce au modèle multi-tenant. Les ressources sont mutualisables et partagées entre les utilisateurs grâce à ce modèle.

- L'abstraction des services : nous avons ajouté la virtualisation aux caractéristiques définies par le NIST, car elle joue un rôle important dans le Cloud.

L'élasticité, la virtualisation et le multi-tenant sont des caractéristiques fondamentales pour les fonctionnalités du Cloud. Ces caractéristiques apportent de nombreux avantages en termes d'agilité, d'économie d'échelle (les dépenses d'investissement baissent au profit des dépenses opérationnelle, la rationalisation), de simplicité pour l'architecture du Cloud. En revanche, des nouveaux verrous à lever au niveau de la sécurité émergent, surtout en ce qui

concerne la virtualisation et le modèle multi-tenant. Pour cela nous allons détailler ces deux concepts ainsi que les modèles des services et les modèles de déploiement du Cloud.

- **Le modèle Multi-tenant**

La terminologie multi-tenant est une expression anglaise qui signifie multi-locataire. Ce modèle a pour objectif de trouver le compromis entre la mutualisation et la customisation.

Dans ce modèle, les services sont construits de façon qu'ils soient partagés par plusieurs tenants en même temps. Les services sont mutualisables, et les tenants partagent la même instance de service.

Le modèle Multi-Tenant [25] est adopté essentiellement du côté des fournisseurs ou des entreprises puisqu'il offre plusieurs avantages en termes:

- Scalabilité: désigne la capacité du système (Cloud) de s'adapter au changement du à la montée en charge (d'ordre de grandeur de la demande) tout en maintenant ses propriétés fonctionnelles et ses performances. En modifiant certains de ses paramètres, un serveur extensible au niveau de l'infrastructure peut, par exemple, augmenter le nombre de ses utilisateurs, sans diminuer ses capacités fonctionnelles.

- Performance : Les fournisseurs peuvent facilement détecter et corriger les problèmes qui sont susceptibles d'un impact grâce à la surveillance et le monitoring c'est-à-dire l'automatisation de l'exploitation des applications.

- Amélioration des services : Avec une seule plate-forme à administrer, les fournisseurs peuvent offrir des services plus adaptés à savoir le dépannage et la résolution des problèmes.

- Mise à niveau accélérée : Automatisation des mis à jour applicatives (montée de version), les nouvelles versions de logiciels ou les correctifs peuvent être déployés plus rapidement.

- **La virtualisation**

Le concept de la virtualisation couvre l'ensemble des techniques permettant d'abstraire les caractéristiques physiques d'un système matériel, système logiciel, ou d'un réseau. On entend parler de la virtualisation dans plusieurs domaines et aux différents niveaux (hardware, software/service, réseaux,…). En réalité, la virtualisation telles qu'elle est appliquée et réalisée aujourd'hui se focalise essentiellement au niveau hardware, permettant ainsi faire fonctionner plusieurs machines virtuelles disposant chacune d'un système d'exploitation sur la même infrastructure physique. La virtualisation implique le partage des ressources, l'abstraction et l'isolation. Elle élimine les frontières physiques d'un système cela contribue à l'amélioration du taux de disponibilité des ressources, mais introduit un nouveau niveau d'administration [26].

La virtualisation a des avantages importants en termes de performance, de coût et de management.

- **Les modèles de services**

Toutes les organisations s'appuient sur des services applicatifs au quotidien. Les fournisseurs de services (SP pour Service Providers) rendent leurs prestations accessibles aux utilisateurs à partir d'interfaces Internet. Le Cloud cherche à externaliser les prestations d'infrastructure informatique nécessaire à l'hébergement de ces services. L'industrie des services informatiques a pour habitude de classer les services de Cloud selon deux types de modèles à savoir les modèles de déploiement et les modèles de services.

Dans ce paragraphe, nous allons présenter les modèles de services qui décrivent les différentes catégories de services accessibles depuis une plate-forme de Cloud. Autrement dit, ce sont les types de services que le fournisseur est capable de proposer. Les plus connus sont le SaaS, le PaaS et l'IaaS.

- Infrastructure as a Service (IaaS): Il s'agit d'une mise à disposition, à la demande, des machines virtuelles, du stockage virtuel, une infrastructure virtuelle ainsi que du matériel et des ressources. Le fournisseur de services gère toute l'infrastructure. Quant à l'utilisateur, il a la possibilité de contrôler les systèmes d'exploitation, le stockage, les machines virtuelles, les applications déployées et il a la possibilité de sélectionner des composants réseau (routeur, firewalls,…). Dans ce type de service, seul le matériel est dématérialisé. Parmi les fournisseurs de ce type de services les plus réputés, on peut citer Amazon EC2 (Amazon Elastic Compute Cloud) qui est considéré comme un fournisseur classique de solution d'IaaS qui permet de louer des machines virtuelles de tailles prédéfinies pour exécuter des applications.

- Platform as a Service (PaaS): le client peut déployer sur l'infrastructure du Cloud ses propres applications à condition que le fournisseur du Cloud supporte le langage de programmation. Ce modèle de services dispose d'environnements spécialisés au développement comprenant les langages, les outils et les modules nécessaires. Dans ce type de service le matériel, l'hébergement et le Framework d'application sont dématérialisé. Le fournisseur de services gère l'infrastructure du Cloud, le système d'exploitation et les logiciels associés. Le client est responsable de l'installation et de la gestion des applications qu'il déploie.

- Software as a Service: l'utilisateur a la possibilité d'utiliser les applications offertes par les fournisseurs de services via le réseau. Ces applications sont accessibles indépendamment de l'équipement de l'utilisateur à travers des interfaces légères (navigateur Web, etc.). Le matériel, l'hébergement, le Framework d'application et le logiciel sont dématérialisés. Dans ce type de service, le client ne gère que ses propres donnés. Tout ce qui se passe entre l'application et les infrastructures relève de la responsabilité du fournisseur. Parmi les revendeurs de solutions SaaS figurent Google Apps, Oracle on Demand, Salesforce.com et SQL Azure.

En Cloud, la flexibilité est obtenue grâce à la virtualisation des systèmes d'exploitation. La plateforme est exécutée via des machines virtuelles et les ressources peuvent être allouées et délibérées à la demande. Ainsi, l'IaaS est considéré le service le plus flexible. D'autre part, l'IaaS est le service le plus simple à mettre en place.

Pour accéder aux services de Cloud, deux technologies principales peuvent être actuellement identifiées. Web services sont couramment utilisés pour fournir l'accès aux

IaaS services, et web browsers sont utilisés pour accéder aux SaaS services. Dans l'environnement PaaS les deux approches peuvent être trouvées.

- **Les modèles de déploiement du Cloud**

Les modèles de déploiement se réfèrent à la localisation et à la gestion des infrastructures de Cloud. Ils indiquent où la solution de Cloud se situe et pour quel objectif. Les principaux modèles de déploiement sont public, privé, communautaire et hybride ;

- Le Cloud public: ce type d'infrastructure est accessible à un large public et appartient à un fournisseur de services de Cloud.
- Le Cloud privé: l'infrastructure du Cloud est louée à une organisation unique. Elle peut être gérée par l'organisation elle-même (Cloud privé interne) ou par un tiers (Cloud privé externe).
- Le Cloud communautaire: l'infrastructure est partagée par plusieurs organisations ayant des intérêts communs. Elle peut être également gérée par les organisations elles-mêmes ou par un tiers.
- Le Cloud hybride: l'infrastructure se compose de deux Cloud ou plus (privé, communautaire ou public), qui restent des entités indépendantes, mais qui sont liées par une technologie normalisée ou propriétaire, permettant la portabilité des données et/ou des applications.

### 3.4.2 Les solutions de sécurité adoptées dans le Cloud

Le Cloud permet à ses clients d'éviter les frais de démarrage, de réduire les couts d'exploitation et d'offrir plus d'agilité et de flexibilité par l'acquisition immédiate des services et des ressources d'infrastructure en cas de besoin. Le Cloud permet aussi aux fournisseurs de développer, déployer et exécuter des applications qui peuvent augmenter leurs capacités sans difficulté (scalabilité) et fonctionner avec performance et fiabilité sans aucun souci des propriétés de l'endroit des infrastructures sous-jacentes. Cependant, le modèle du Cloud et ses caractéristiques architecturales soulèvent des problèmes et des défis de sécurité.

D'abord, chaque modèle de service dans le Cloud peut induire des soucis de sécurité [27]. Dans le modèle IaaS qui fournit un ensemble de composants virtualisés d'infrastructure comme les machines virtuelles et le stockage, il existe des points critiques comme le fait de faire confiance à l'image de la machine virtuelle, la protection des hôtes et la sécurisation de la communication entre les hôtes.

Dans le modèle PaaS, les environnements de développement fournis par ce modèle ont un impact visible sur l'architecture de l'application comme les contraintes sur les services qui peuvent être demandés du système d'exploitation. Par exemple, un environnement PaaS pourrait limiter l'accès à des parties bien définies dans le système de fichiers, nécessitant ainsi un service d'autorisation rigoureux.

Finalement, dans le modèle SaaS, les fournisseurs du Cloud offrent des logiciels d'application comme des services sur demande. Parce que les clients acquièrent et utilisent des composants logiciels de différents fournisseurs, ceci a engendré des problèmes crucials concernant une composition sécurisée de ces composants et la protection des informations traitées par ces services composés.

Pour les modèles de déploiement du Cloud, la sécurité pour le Cloud privé et le Cloud communautaire est sous le contrôle des fournisseurs et des partenaires du Cloud. Par ailleurs, la sécurité du Cloud public n'est pas garantie par les fournisseurs. Dans ce cas, un autre acteur entre dans le jeu, c'est l'utilisateur. Il est responsable aussi de la sécurité du Cloud.

Les caractéristiques architecturales du Cloud, bien qu'elles apportent aux organisations une alternative efficace, flexible et rentable d'hébergement des ressources informatiques, suscitent aussi des inquiétudes en termes de sécurité [27].

En effet, le Cloud permet l'externalisation des données et des applications, mais le défi est d'assurer que seulement les entités autorisées peuvent y accéder. En utilisant les environnements du Cloud, nous accordons à des tiers la prise des décisions sur nos données et nos plates-formes d'une façon jamais vue auparavant. Il est essentiel de définir des mécanismes appropriés afin d'empêcher les fournisseurs du Cloud d'utiliser les données de leurs clients d'une manière qui n'a pas été convenue dans le contrat préétabli entre les deux parties. Les clients ont besoin d'avoir confiance envers leurs fournisseurs, leur compétence technique et leur stabilité économique.

La mise en commun des ressources est un autre critère unique au Cloud qui permet essentiellement aux fournisseurs du Cloud de gérer l'utilisation des ressources plus efficacement en partitionnant une infrastructure virtualisée et partagée parmi plusieurs clients. De point de vue client, le niveau de partage des ressources, l'isolation des données appartenant à plusieurs utilisateurs et la présence des mécanismes de protection présentent des éléments clés et critiques dans la notion d'usage d'une infrastructure partagée. Ainsi, les fournisseurs doivent tenir compte des aspects de sécurité liés, par exemple, à la politique d'accès, au déploiement des applications et à l'accès aux données et leur protection afin de fournir un environnement mutualisé sécurisé.

Quant à la virtualisation, elle désigne une technologie importante qui contribue à fournir de l'infrastructure et des ressources abstraites aux clients sous la forme des machines virtuelles isolées. Un hyper-viseur ou un moniteur de machine virtuelle est élément du logiciel de la plate-forme de virtualisation qui permet à des systèmes d'exploitation multiples de fonctionner sur un ordinateur hôte concurremment. Malgré que ceci fournisse un moyen de générer des ressources virtualisées à partager, la présence d'une telle technologie augmente la surface d'attaque. Il faut alors des mécanismes pour assurer une forte isolation et une communication sécurisée entre les machines virtuelles. Ceci peut être assuré en utilisant un mécanisme de contrôle d'accès flexible qui gouverne le contrôle et le partage des capacités des machines virtuelles au sein d'un hôte du Cloud.

L'hétérogénéité est aussi un autre critère du Cloud qui pourrait poser des défis de sécurité. L'hétérogénéité dans le Cloud se décline sous des formes différentes. D'abord, les fournisseurs du Cloud utilisent différents ressources matérielles et logicielles pour construire les environnements du Cloud. Dans une certaine mesure, la virtualisation des ressources assure une homogénéité de haut niveau du système, mais l'utilisation de la même

infrastructure pour supporter différents tenants avec des exigences différentes de protection et de système pourrait générer des difficultés. Il y a également un problème potentiel avec l'hétérogénéité verticale des services du Cloud. En fait, un client pourrait s'abonner à un service IaaS fourni par un certain fournisseur, l'associer à un service PaaS d'un autre fournisseur, et acquérir plusieurs services SaaS fournis par un troisième fournisseur. Les hypothèses que peuvent faire ces fournisseurs en composant les services peuvent affecter gravement la confiance émergente et les propriétés de sécurité. En outre, l'hétérogénéité existe aussi dans le niveau de traitement de sécurité fourni par chaque composant, générant ainsi des défis d'intégration. Dans un environnement multi-tenant, les besoins de protection de chaque tenant peuvent être différents. De plus, chaque tenant peut avoir des relations de confiance différentes avec le fournisseur et certains tenants peuvent être en réalité des attaquants malveillants, résultant par conséquent des problèmes complexes de confiance.

Toutes ces caractéristiques pourraient exposer le modèle du Cloud à des menaces et des attaques éventuelles. Dans le Cloud, la sécurité est partagée entre le fournisseur et l'utilisateur du Cloud. Les deux entités ont besoin de faire confiance l'un à l'autre et de se coopérer pour améliorer la sécurité. Il existe de nombreuses menaces de sécurité qui émergent à l'intérieur ou à l'extérieur de l'environnement du Cloud du fournisseur ou de l'utilisateur. Elles peuvent être classées généralement comme des menaces internes, des attaques externes malveillantes, perte de données, problèmes liés au partage des ressources, perte du contrôle, et perturbation ou interruption du service [28].

Pour faire face à ces menaces et atteindre les objectifs de disponibilité, confidentialité et intégrité, différentes approches et solutions de sécurité ont été déployées. Nous citons ci-dessous quelques solutions existantes relatives à des aspects de sécurité nécessaires pour fiabiliser l'environnement du Cloud comme l'authentification et la gestion d'identité, le contrôle d'accès, le provisionnement et la composition de service sécurisés, la gestion de confiance, la protection des données, et la gestion d'une hétérogénéité sémantique.

- **Authentification et gestion d'identité :**

Un mécanisme de gestion d'identité (IDM) peut aider à authentifier les utilisateurs et les services en se basant sur des accréditations. Un système IDM doit être capable de protéger les informations privées et sensibles des utilisateurs.

La gestion d'identité basée sur l'approche « user-centric » a récemment bien retenu l'attention [29]. Dans cette approche, les identifiants ou les attributs permettent d'identifier et définir le profil utilisateur. Cette approche permet aux utilisateurs de mieux contrôler leurs identités numériques. Tant que les utilisateurs peuvent accéder au Cloud de différents endroits comme la maison, le bureau, l'école, ou autres lieux publiques, ils doivent être capables d'exporter leurs identités numériques et les transférer en toute sécurité à plusieurs ordinateurs. Un IDM « user-centric » implique aussi que le système maintienne le contexte d'information pour chaque utilisateur afin de répondre au mieux à la requête d'un utilisateur dans une situation donnée.

Parmi les problèmes récurrents concernant la gestion d'identité dans le Cloud [30], est le fait que le framework d'identification et d'authentification ne pourrait pas naturellement s'étendre au Cloud publique et le changement du framework existant afin de supporter les services Cloud peut s'avérer difficile. L'alternative d'employer deux systèmes d'authentification différents, un pour le système interne de l'organisation et l'autre pour les systèmes externes basés sur le Cloud, est une approche complexe. La fédération d'identité,

popularisé avec l'introduction des architectures orientées service, semble être une solution. Elle permet l'organisation et le fournisseur du Cloud de faire confiance l'un à l'autre et de partager les identités numériques et les attributs dans les deux domaines et de fournir un moyen pour le « single sign-on ». SAML [11] et OpenId [31] sont deux possibles moyens pour assurer la fédération d'identité.

- **Contrôle d'accès :**

L'hétérogénéité et la diversité des services, ainsi que les multiples conditions d'accès aux domaines dans le Cloud, demandent une politique de contrôle d'accès rigoureuse. En particulier, les services de contrôle d'accès doivent être suffisamment flexibles pour saisir les exigences dynamiques d'accès liées au contexte, aux attributs ou aux informations d'identifications (accreditations), et pour mettre en œuvre le principe de sécurité du moindre privilège. Ces services de contrôle d'accès pourraient avoir besoin d'intégrer des mécanismes de protection des informations personnelles à travers des règles complexes. Il est important aussi que le système de contrôle d'accès utilisé en Cloud soit facilement géré et sa distribution de privilèges soit administrée d'une manière efficace. En outre, les modèles de services du Cloud doivent fournir des interfaces de contrôle d'accès génériques dans le but d'interopérabilité, ce qui demande une spécification de contrôle d'accès neutre vis à vis la politique et un cadre d'application capable d'adresser les problèmes d'accès inter-domaine.

Parmi les nombreuses méthodes proposées jusqu'à présent, le contrôle d'accès basé sur les rôles (en anglais, role-based access control : RBAC) a été largement reconnu comme le modèle le plus prometteur grâce à sa simplicité, sa flexibilité à saisir les exigences dynamiques, son support du principe du moindre privilège, et sa gestion efficace des privilèges [32]. De plus, RBAC est neutre par rapport à la politique, il peut capturer plusieurs exigences de la politique, et il est le mieux adapté aux besoins d'intégration de la politique.

Des extensions récentes du RBAC, comme les modèles "credential- based RBAC", "generalized temporal RBAC", et "location- based RBAC", fournissent les structures de modélisation nécessaires et les capacités pour capturer les conditions d'accès basées sur le contexte. Dans le Cloud, les fournisseurs de services ne connaissent pas en général leurs utilisateurs en avance. Par conséquent, il est difficile d'assigner les utilisateurs directement aux rôles dans la politique de contrôle d'accès. Ainsi, recourir à une politique basée sur des informations d'identification (credentials) ou des attributs pourrait améliorer cette capacité. Cependant, il existe peu de travaux employant RBAC et ses extensions au sein des environnements orientés service comme le Cloud.

- **Framework de gestion de confiance :**

Afin de faciliter l'intégration d'une politique entre plusieurs domaines dans les environnements du Cloud, un framework basé sur la confiance qui permet une intégration automatisée d'une politique fondée sur la confiance s'avère essentiel. Pour ce faire, il faut répondre à plusieurs questions : Comment peut-on établir la confiance et déterminer le mappage des accès afin de satisfaire les exigences d'accès inter-domaine, et comment peut-on gérer et maintenir les valeurs de confiance dynamiques et adapter les conditions d'accès au moment où la confiance évolue ?

Des mécanismes existants de négociation de confiance s'intéressent principalement à l'échange des informations d'identification [33][34] et ne traitent pas le besoin d'intégrer des techniques de négociation de confiance basée sur les exigences avec les mécanismes de contrôle d'accès [35]. Une approche possible est de développer un framework d'intégration de politique basé sur la confiance qui facilite l'intégration de politique et l'évolution selon les exigences de l'accès inter-domaine et aux services.

Puisque la dynamique de la composition des services dans le Cloud peut être complexe, le framework de confiance et de contrôle d'accès doit inclure les primitives de délégation [35]. Des travaux existants liés à la délégation, incluant la délégation basée sur les rôles, se sont intéressé aux problèmes liés à la délégation des privilèges. Les mécanismes cryptographiques efficaces pour la délégation de confiance impliquent des problèmes de vérification et de révocation complexes des chaines de confiance, générant des problèmes de gestion significatifs. Ces approches doivent être incorporées dans les frameworks de composition de service.

### 3.4.3 Synthèse

Le Cloud a été adopté par plusieurs entreprises et organisations dans le monde. Il apporte beaucoup d'avantages que ce soit pour les clients ou les fournisseurs. Il permet d'utiliser des ressources matérielles distantes pour créer des services de type SaaS, gérables depuis une interface web et accessible depuis n'importe quel terminal connecté à l'internet. Il facilite les tâches pour l'entreprise qui n'aura pas besoin de monter leur infrastructure réseaux/services, ce qui demande un savoir-faire (maintenance, sécurité…) et un financement. Il permet à l'utilisateur de gérer ses données tout en partageant les réseaux, serveurs, données, services et applications qui peuvent être facilement et rapidement approvisionnés et libérés avec un effort minimum.

Les apports bénéfiques qu'offre une solution de Cloud concernent la baisses des coûts de la production des services informatiques grâce à la disponibilité et l'élasticité des ressources ainsi qu'à des systèmes de facturation portant sur la consommation réelle de services par opposition aux systèmes de forfaits (pour lequel le client paie même s'il consomme rien).

Un autre avantage qui est considéré comme un argument phare des fournisseurs de Cloud est la réduction des dépenses d'investissement, à savoir la réduction des couts opérationnels, des logiciels, du matériel et l'optimisation de la puissance de calcul.

Malgré le frein de la sécurité, les services de Cloud offrent notamment la possibilité d'adapter en temps réel les ressources informatiques à la demande du métier, de transformer les dépenses d'investissement en dépenses opérationnelles. En effet, l'utilisation croissante de services de Cloud, qui ne présentent pas les mêmes niveaux de risques en fonction de leur nature (privé, public, hybride, etc.). C'est justement cette variété d'exploitation et d'implémentation qui délimite les exigences et les contraintes en termes de sécurité et de confidentialité.

En effet au niveau de SaaS/ PaaS, il faut solliciter un fournisseur compétent en sécurité car tout sera sous sa responsabilité. Au niveau IaaS/PaaS, il n'y a généralement aucun engagement des fournisseurs en matière de sécurité, car la responsabilité est du côté client dans les machines virtuelles (à sécuriser) qu'ils vont faire héberger dans le Cloud. S'ajoutent à cela les vulnérabilités propres aux infrastructures de virtualisation amplifiées par l'hétérogénéité des plates-formes (pas de garantie possible dans ce cas).

En conclusion, la sécurité est fournie comme service et hérite les principes SOA. Le Cloud apporte de nombreux avantages en termes d'externalisation, de prestation de services, d'élasticité, la fiabilité et l'évolutivité. Cependant, il n'y a pas un modèle de référence de la sécurité, chaque fournisseur offre les services de sécurité selon les besoins client et son savoir-faire.

## 3.5. Conclusion

| | Valeurs ajoutés | Limites |
|---|---|---|
| **Web Service** | • Automatisation et gestion du processus métier au sein de l'entreprise.<br>• Gestion de la sécurité en couches sous forme de services réutilisables --> éviter les problèmes de silo.<br>• Déploiement de la sécurité de bout en bout. | • Approche client/serveur basée sur des Web Services APIs.<br>• Au niveau protocolaire: HTTP favorise l'approche client/serveur avec un niveau de sécurité faible.<br>• Absence du contrôle de QoS au niveau des Web Services. |
| **SOA** | • Nouvelle génération d'applications ouvertes et accessibles.<br>• Sécurité fournie en tant que service.<br>• Services réutilisables, interopérables, autonomes et à liens lâches. | • Modèle de service: approche orientée opération.<br>• Pas de maintien du niveau de sécurité requis lors de changement du contexte.<br>• Pas de prise en compte des préférences de l'utilisateur et de la mobilité. |
| **Cloud** | • Scalabilité, Pay-as-you-go, délivrance de service sur demande<br>• Réduction des couts d'investissement et opérationnels.<br>• Tout est fourni comme service --> Security as a service. | • Augmentation des risques et des menaces de sécurité.<br>• Pas d'architecture de sécurité bien adaptée.<br>• Pas de gestion de QoS : autogestion et mutualisation. |

# 4

# Proposition

Plus les besoins de l'utilisateur NGN exigent la continuité du service dans un contexte hétérogène et mobile, plus le besoin de sécurité comme un service devient indispensable. Afin de garantir l'unicité et l'accès sécurisé de la session centrée utilisateur dans ce contexte, nous proposons alors une nouvelle architecture de sécurité basée sur le concept « Security as a Service » (§4.1). Cette architecture fournit les services de base de sécurité (§4.2) à savoir le service d'identification, le service d'authentification et le service d'autorisation qui permettent de contrôler et sécuriser l'accès aux différents services de la session de l'utilisateur. Ces services sont composés selon des niveaux de visibilité, où tout est considéré comme service, formant ainsi la session de service qui doit être sécurisée. Pour ce faire, nous introduisons des services de support de sécurité (§4.3) au niveau du terminal (VPDN) ainsi qu'au niveau des services (VPSN) permettant de sécuriser la composition des services dans la session et d'assurer la continuité de la sécurité. A la fin, nous proposons un service d'Audit (§4.4) de sécurité basé sur la QoS permettant de contrôler et évaluer la sécurité des services.

## 4.1. Security as a service

Suite à l'émergence des réseaux et services de nouvelle génération (NGN et NGS) et à l'évolution des technologies, l'environnement devient de plus en plus hétérogène et mobile. En termes de sécurité, le bouleversement est important. En effet, les frontières du système auparavant bien délimitées deviennent beaucoup plus ouvertes : des services multiples inconnus à l'avance et des communications multiples entre les services et avec les utilisateurs. De plus, l'hétérogénéité et la variété des ressources (terminaux, réseaux et services) impliquées dans la session de l'utilisateur augmentent la complexité des tâches de sécurité. En outre, la mobilité avec ses différentes déclinaisons (mobilité du terminal, mobilité de l'utilisateur, mobilité du réseau et mobilité de service), se reflète principalement sur la session qui doit être unique, sécurisée, « sans couture » et « sans coupure » en assurant la continuité des services. La mobilité impacte fortement la délivrance de service de bout en bout sécurisée.

Cet environnement ouvert, hétérogène et mobile, qui expose des risques significatifs en termes de sécurité, est devenu de plus en plus vulnérable. Pour cette raison, la sécurité ne peut plus être assurée par l'application d'une manière statique et centralisée. Il nous faut de la souplesse et de l'adaptabilité à des situations qui sont de plus en plus complexes. Une nouvelle approche s'avère nécessaire : "Security as a service", la sécurité est fournie comme un service en se déclinant sous forme de composants de services qui doivent être composables selon les besoins (spatiaux et temporels) et les préférences de l'utilisateur.

Cette approche est capable d'offrir à l'utilisateur un accès simplifié et sécurisé à ses services tout en assurant la continuité et l'unicité de la session et respectant les exigences de mobilité.

Par conséquent, nous adoptons pour la conception de notre architecture de sécurité l'architecture orientée service (SOA). Cette approche SOA fournit plusieurs avantages en termes de réutilisation, dynamicité, flexibilité, interopérabilité et évolution de services. Cependant, suivant notre contexte NGN/ NGS, notre proposition des services de sécurité n'est pas limitée aux caractéristiques des services SOA susmentionnées. En fait, notre approche consiste à fournir d'autres aspects architecturaux supplémentaires comme la mutualisation et l'autogestion.

Nous décrivons en détail, ci-dessous, la spécification d'un service de sécurité selon le modèle UBIS (§4.1.1), et notre architecture de sécurité orientée service (§4.1.2).

### 4.1.1 Spécification d'un composant de sécurité UBIS

L'utilisateur d'aujourd'hui change tout le temps de localisation, possède plusieurs terminaux pour se connecter et souhaite avoir une session avec un accès sécurisé sans couture et sans coupure quels que soient ses déplacements, son terminal et à n'importe quel moment où il sollicite son service. Pour répondre à ces besoins, il a fallu évoluer vers une nouvelle génération de services qui soit capable de s'adapter aux changements de contexte et de préférences de l'utilisateur. Une réingénierie d'architecture accompagnée par une réingénierie de service semble nécessaire. Pour ce faire, nous avons adopté la notion service comme l'élément clé de notre modèle architecturale de sécurité. En s'appuyant sur la définition d'un service proposé par l'ISO 20000 [36]: « Un service est une prestation composable qui doit être source de valeur pour le consommateur et le fournisseur », on peut considérer que toute architecture qui veut structurer des services de futur (NGS) doit obligatoirement supporter une composition dynamique de services. Pour cette raison, nous avons conçu notre architecture de sécurité en se basant sur un ensemble de composants fonctionnels modélisés sous forme de services.

Ces services sont conçus selon un modèle de service qui s'appuie sur un ensemble de caractéristiques (§4.1.1.1), un agent de QoS (§4.1.1.2) et un composant de Monitoring (§4.1.1.3). Ces éléments sont décrits dans ce qui suit.

#### 4.1.1.1 Caractéristiques d'un composant de service

Dans cette partie nous allons spécifier un composant de service, qu'il soit ou non un composant de sécurité, en décrivant ses différents critères lui permettant de participer à cette composition dynamique de service dans un tel contexte « user-centric ».

Un composant de sécurité est un élément du système qui exécute un service prédéfini et est capable de communiquer avec d'autres composants. Les composants de service de sécurité doivent respecter fondamentalement les mêmes critères de l'approche service spécifiés dans SOA et UBIS. En effet, la composition de services doit être horizontale pour faire sauter les problèmes de l'approche monolithique« client/serveur », et personnalisée selon l'approche « user centric » de notre contexte, et doit faire converger tous les services appartenant à différents domaines (Telco, Web et IT). Ceci nous ramène à redéfinir un nouveau modèle de l'approche service qui est applicable pour n'importe quel contexte technologique et dans n'importe quel contexte d'usage.

Par conséquent, nous proposons un modèle de service basé sur un ensemble de caractéristiques qui doivent être respectées par tous les composants de service, qui s'appuie non seulement sur les propriétés spécifiées dans le cas de SOA mais aussi étoffée par des nouvelles caractéristiques permettant une composition de service sécurisée, dynamique et « sans couture ».

- ***Exigences fonctionnelles du service***

Un composant de service de sécurité doit tout d'abord respecter les propriétés de base citées ci-dessous. Il doit être :

- *Interopérable* : La composition de service global repose essentiellement sur l'interopérabilité. Cette propriété permet la coopération entre plusieurs ESs tout en évitant les conflits de traitement et de deadlocks. En effet, l'interopérabilité est assurée via des interfaces génériques d'entrée et de sortie permettant d'avoir un couplage lâche entre les ESs qui composent un service global.

- *Réutilisable* : Une phase d'adaptation peut être nécessaire selon le contexte d'exécution. Un élément de service est conçu de façon qu'il soit générique afin de favoriser son réutilisation dans différents processus, ce qui permet de réduire les efforts de développement nécessaires pour répondre à des nouveaux besoins.

- *Autonome* : Ce critère signifie qu'un composant de service est fonctionnellement indépendant. Il doit être autosuffisant ce qui veut dire qu'il doit être capable d'assurer ses fonctionnalités sans avoir besoin d'un autre composant de service. Ceci permet d'éviter qu'il devient incapable de remplir sa fonction si un autre devient non disponible par cause de panne ou autres. L'autonomie permet aussi de remplacer un élément de service dégradé par un autre sans perturber le fonctionnement des autres composants (ESs).

Afin d'assurer plus de flexibilité et de dynamicité à la composition de service ainsi qu'une personnalisation de service quant au contexte « user-centric » tout en garantissant une qualité de service optimale, et pour que les composants de services peuvent être partagés entre différents utilisateurs et sollicités dans différentes localisations, différents contextes et par différents terminaux, nos composants de services doivent être conformes aussi aux caractéristiques suivantes; ils doivent être :

- *Mutualisables* : Ce critère reflète la capacité d'un composant de service à partager, simultanément ses ressources entre plusieurs utilisateurs. Ceci permettra d'optimiser l'utilisation de ces ressources et de répondre à plusieurs requêtes de service. Pour garantir cet aspect, les composants de service ne devront pas exiger un état ou des renseignements particuliers par rapport aux clients quand ils sont invoqués.

- *Stateless* : Afin de pouvoir mutualiser un composant de service pour plusieurs utilisateurs, il doit être « stateless ». Donc un composant de service doit exécuter des tâches (opérations) génériques pour tous les utilisateurs sans considérer leurs contextes ou maintenir leurs données spécifiques. Ce critère aide à garantir le remplacement dynamique d'un composant par un autre dans une composition de service à couplage lâche.

- *Interconnectés* : Ce critère est ajouté à l'interopérabilité pour permettre ainsi l'interfonctionnement de plusieurs composants de service qui peuvent coopérer entre eux afin de fournir un service global. L'interconnexion entre les services amène à définir des interfaces et des liens génériques.

- *Auto-gérables*: Lors de la conception, un élément de service doit être capable de surveiller et de contrôler ses propres paramètres de QoS courante. Il doit ainsi vérifier en temps réel si son comportement est conforme à celui négocié dans son contrat de QoS préétabli.

Pour la spécification de nos composants de sécurité, nous suivons le modèle du « black box » ayant trois interfaces sur lesquelles ils reçoivent les requêtes (Figure 8). Pour permettre un interfonctionnement entre les composants de service, ces interfaces doivent être les plus génériques possibles. Et ils doivent aussi permettre d'accéder aux fonctionnalités réalisées par les composants de service. Elles sont :

- *Usage Interface*: elle permet de faire appel aux fonctions principales réalisées par le composant de service pour lancer le traitement des données par les opérations afin de rendre le service demandé.

- *Control Interface*: elle inclut la signalisation nécessaire pour le provisionnement des composants de service, l'activation et la désactivation.

- *Management Interface* : Elle permettra d'identifier les échanges référant l'autogestion concernant la QoS du composant, et ainsi gérer les événements internes qui peuvent se produire.

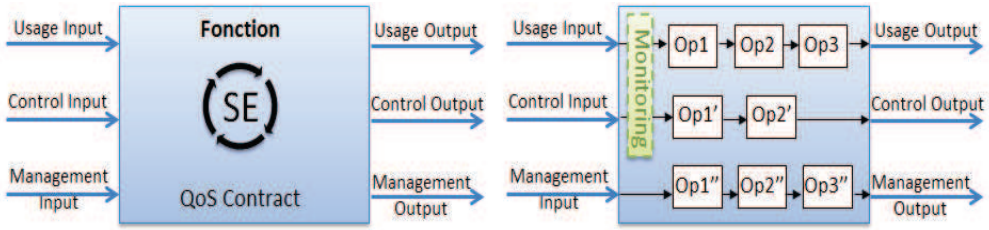



***Figure 8 :*** *Le modèle « Black-box » d'un composant de service avec les 3 interfaces.*

### ➢ *Exigences Non-fonctionnelles du service : QoS*

***Une exigence non-fonctionnelle*** est une exigence qui caractérise une propriété (qualité) désirée du système liée à l'usage tel que la performance, la disponibilité, la sécurité, la robustesse, la maintenabilité, l'efficacité, le temps de réponse, etc. Une exigence non-fonctionnelle doit être ***mesurable*** sinon elle n'est qu'une intention ou un objectif général à savoir la convivialité d'une application ou d'un service.

Dans l'objectif d'améliorer la délivrance de service pour satisfaire les besoins de l'utilisateur qui désire avoir un support d'échange le plus *transparent* possible lorsqu'il demande un service, il semble nécessaire de définir les exigences non-fonctionnelles qui décrivent les aspects comportementaux d'un service qui peuvent impacter la qualité de l'exécution de la fonction de transfert de l'information.

Basé sur les travaux de notre groupe de recherche, chaque composant de service doit tenir en compte, dès sa conception, non seulement les aspects fonctionnels mais aussi les aspects non-fonctionnels.

Dans le cas présent, la QoS caractérise la partie non-fonctionnelle d'un composant de service UBIS, c'est-à-dire, elle représente son comportement et le contrat établi entre un utilisateur et un fournisseur. Son évaluation (Modèle de QoS UBIS) repose sur quatre critères : ***Disponibilité, Fiabilité, Délai et Capacité*** (Figure 9).

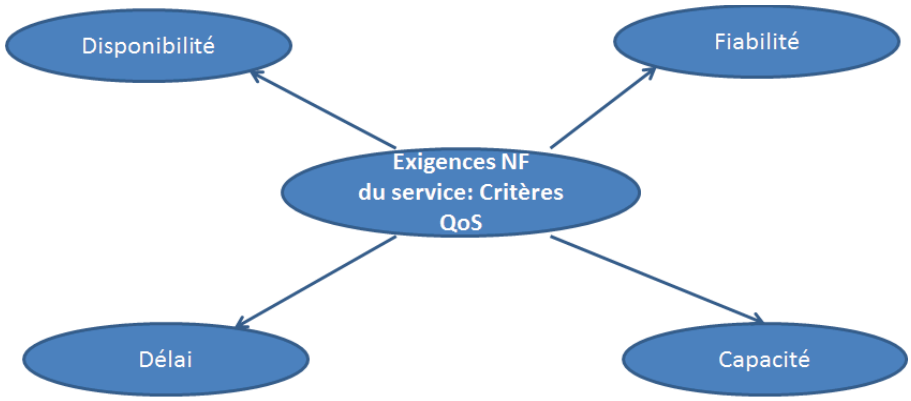



***Figure 9 :*** *Modèle de QoS.*

Ces critères découlent des quatre types de la transparence du transfert d'information:

La *« transparence temporelle et spatiale* » signifie qu'il est possible de transférer le contenu à chaque fois que l'utilisateur en produit et aussi longtemps que dure sa génération, là où il se trouve ***(Disponibilité) ;***

La *« transparence sémantique*», se reflète par le transport d'informations sans altération pour qu'elles restent interprétables par le destinataire ***(Fiabilité) ;***

La *« transparence à la distance* » traduit le fait qu'il est possible de faire n'importe quelle mise en relation sans mettre de temps ni changer la relation temporelle intrinsèque aux informations générées ***(Délai) ;***

La *« transparence au comportement de la source* » consiste à pouvoir transférer le volume d'information instantanément généré ***(Capacité) ;***

C'est ainsi que nous avons retenus ces quatre critères de QoS qui semblent être nécessaires et suffisants pour caractériser les aspects non-fonctionnels de n'importe quel service dans notre contexte. Ces critères se définissent alors comme suit:

- **La Disponibilité**: représente l'aptitude d'un service à être présent et prêt à remplir sa fonction à un instant donné. Elle exprime l'accessibilité d'une ressource en tenant compte des conditions contractuelles temporelles et spatiales. Elle peut être vue comme le taux de réussite suite à des sollicitations pour l'accès à l'entité (service).
- **La Fiabilité**: représente l'aptitude d'une entité à accomplir sa fonction sans détérioration de l'information traitée tout en respectant les demandes et les conditions contractuelles.

- **Le Délai :** traduit l'aptitude d'un service à s'exécuter en respectant le délai spécifié dans les demandes et les conditions contractuelles. Il exprime la durée de traitement dans un nœud et la durée d'attente et de traversée sur les liens du réseau.
- **La Capacité**: représente l'aptitude d'une entité à avoir les moyens nécessaires pour réaliser les taches requises.

#### 4.1.1.2 Agent de QoS

Pour assurer l'autogestion du composant de service, un agent de QoS est intégré à chaque composant de service permettant d'évaluer ou superviser son comportement et de vérifier s'il respecte le contrat de QoS auquel il est contraint. Cet agent compare en temps réel sa QoS courante à une valeur seuil à ne pas dépasser, et selon le résultat, il envoie comme notification soit « IN-Contract » soit « OUT-Contract ».

En effet, l'agent QoS évaluent la qualité de service d'un composant de service à travers trois types de valeurs à savoir *les valeurs de conception, les valeurs courantes et les valeurs seuils :*

- *Les valeurs de conception* traduisent les capacités maximales du traitement d'un composant de service qui sont définies au moment de conception. Elles interviennent lors de la planification et le dimensionnement des services.
- *Les valeurs courantes* renseignent sur le comportement des entités. Ces valeurs permettent d'avoir durant l'exploitation une image du comportement des services en *temps réel.*
- *Les valeurs seuils* indiquent la limite à ne pas dépasser pour assurer un fonctionnement normal du service. Ces valeurs, une fois atteintes, permettent de déclencher les réactions adéquates et les processus d'autogestion

Afin de pouvoir contrôler et évaluer la QoS d'un composant de service par l'agent QoS, une phase de mesure s'avère nécessaire. Une description quantitative n'est pas suffisante, les quatre critères de QoS (Fiabilité, Capacité, Disponibilité, Délai) doivent être évaluables et associés à des métriques (voir tableau) pour pouvoir donner des garanties quantitatives à l'utilisateur final. Ces métriques personnalisent la vision fournisseur de service et celle de l'utilisateur. Prenons par exemple le critère de disponibilité, un fournisseur de service va rejeter l'utilisateur β (taux de rejet), si son réseau étant dimensionné pour un certain nombre d'utilisateurs, ne peut plus l'accepter, donc pour cet utilisateur, le réseau est inaccessible (taux d'inaccessibilité). Notons que le réseau est néanmoins en activité. Le taux de rejet relève du vocabulaire du fournisseur (producteur de service), alors que l'accessibilité se place du point de vue de l'utilisateur.

Les métriques de QoS doivent être spécifiques au niveau de service considéré et donc à l'endroit où sont prises les mesures, le tableau ci-dessous décrit le cas d'un service applicatif.

| ***Critères QoS*** | ***Métriques QoS Vision Fournisseur de service*** | ***Métriques QoS Vision Utilisateur Final*** |
|---|---|---|

| | | |
|---|---|---|
| ***Disponibilité*** | Taux de rejet | Taux d'inaccessibilité |
| ***Fiabilité*** | Taux d'erreur<br>Taux de pertes | Taux d'altération des messages |
| ***Délai*** | Temps de traitement | Temps de réponse |
| ***Capacité*** | Requêtes/seconde | Messages/seconde |

***Tableau 1 :*** *Les métriques de QoS : vision fournisseur de service et utilisateur final.*

Pour un service réseau, les métriques de QoS ne sont pas les mêmes, ils sont spécifiques à ce service, par exemple le critère délai est associé à la latence et la gigue.

#### 4.1.1.3 Monitoring

Le composant de Monitoring se met en coupure sur le plan d'usage. Il a pour tâche d'observer et collecter les données relatives aux exigences fonctionnelles d'un composant de service qui sont préétablies dans son contrat.

Il dispose d'un ensemble de compteurs permettant de **mesurer** son comportement en temps réel. Il compte systématiquement tout type de requêtes reçues et envoyées par le composant de service (Figure 10), par exemple, le nombre de requêtes reçues erronées (IN Error) et le nombre de requêtes reçues rejetées pour cause de manque de place (OUT Discard). Il peut calculer également différent délai comme le temps de traitement. C'est une entité qui peut être centralisée, mais si on prend la logique où tout est service, il peut être intégré facilement dans chaque composant service. Des travaux sont en cours pour ce composant de monitoring.

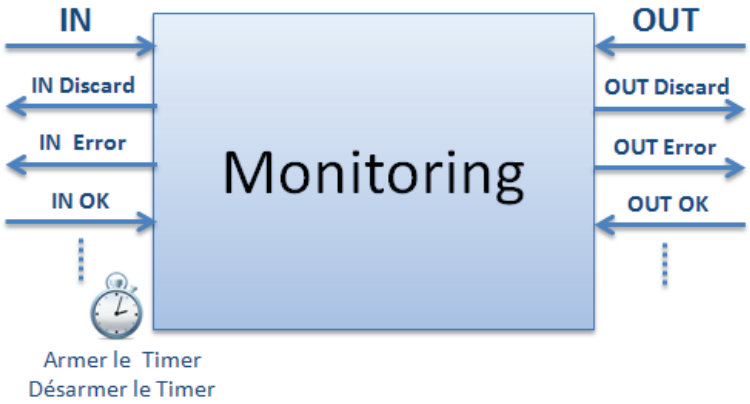



***Figure 10 :*** *Le service Monitoring.*

Le terme « **Mesure** » peut revêtir différents sens dans le domaine de l'évaluation. Il se traduit dans la langue anglaise (l'anglais technique) par plusieurs mots «Measure», « Measurement ». C'est pourquoi, il est convenu de préciser la définition retenue et les correspondances décrites ci-dessous pour chacun de ses vocables afin d'éviter les ambiguïtés:

- « Measure » = valeur mesurée : désigne « la valeur » de la mesure ; il s'agit donc d'une valeur quantitative, résultat de la « mesure de la mesure de sécurité» ;
- « Measurement » = mesurage : il est défini comme étant le «processus de mesurage». Il s'agit de la «mise en œuvre de métriques» d'une manière élémentaire, ou globale. Le terme « Measurement » couvre donc la définition, le choix, le déploiement et l'évaluation d'indicateurs.

Dans le domaine de la sécurité, le mesurage, selon ISO/IEC 27004 [37], représente le «*processus d'obtention d'information relative à l'efficacité du système de management de sécurité de l'information (SMSI) et des mesures de sécurité, à l'aide d'une méthode d'évaluation, d'une fonction d'évaluation, d'un modèle analytique et de critères de décision*».

On utilise également un autre terme dénommé « Metrics » ou « Métriques » qui est à la base utilisé dans la métrologie et utilisé notamment dans le domaine de l'évaluation. Les «Métriques» représentent l'ensemble des éléments permettant de fournir une évaluation qualitative ou quantitative représentative d'une situation donnée.

Rappelons que la métrologie est la science qui s'intéresse aux côtés théoriques et pratiques de la mesure, dans tous les domaines de la science et de la technologie.

Nous retenons le terme «Mesure» qui désigne seulement le résultat de l'action mesurer.

Le résultat de la mesure effectuée par le composant monitoring permet d'avoir les valeurs mesurables de la QoS associés aux critères (Disponibilité, Fiabilité, Délai, Capacité) qui vont servir à l'évaluation du comportement (QoS) d'un composant de service par l'agent QoS (Voir Tableau 2).

| ***Critères QoS*** | ***Métriques QoS : Mesures (Service Monitoring)*** |
|---|---|
| ***Disponibilité*** | **Taux de rejet** : Nombre de requêtes rejetées / nombre de requêtes totales<br>qui traduit le taux d'accessibilité au composant de service |
| ***Fiabilité*** | **Taux d'erreur :** Nombre de requêtes erronées / nombre de requêtes totales<br>qui traduit le taux d'altération de messages |
| ***Délai*** | **Temps de traitement** |
| ***Capacité*** | **Requêtes/seconde** |

***Tableau 2 :*** *Mesures effectuées par le service Monitoring.*

### 4.1.2 L'architecture de sécurité

Pour tirer profit des nouvelles architectures orientées services et pour pouvoir répondre aux besoins de l'utilisateur moderne, il nous faudra harmoniser entre la mobilité,

l'hétérogénéité de l'environnement, le contexte ambiant, les préférences de l'utilisateur ainsi que la sécurité.

Face à l'image des architectures SOA, nous ne pouvons plus avoir une application de sécurité monolithique. Elle doit être fournie comme un service. En plus, la mobilité nous impose une répartition des tâches. La coopération devrait rester sécurisée.

Afin de dépasser les défis relatifs à notre contexte que nous avons identifiés, nous proposons une nouvelle organisation d'architecture de sécurité basée sur l'approche orientée service. Cette architecture de sécurité (Figure 11) a pour fonction de gérer et contrôler la sécurité de la session tout en garantissant la continuité de service et à la fois assurer l'accès sécurisé aux services. Elle s'appuie essentiellement sur trois acteurs (composants) majeurs: le fournisseur des services de sécurité (Securityware), l'agent de sécurité (Security Agent) et la base de données (Security Datastore).

Une des briques essentielles dans notre architecture est le *Securityware* qui joue un rôle principal dans le management et le contrôle de la sécurité. Il est conçu pour fournir l'ensemble des services de sécurité. Il comporte, comme il est illustré dans la figure, les composants de sécurité suivants : *Le service d'identification* : Il permet de reconnaitre l'identité de l'utilisateur par le système.

- *Le service d'authentification* : Il permet de déterminer si une identité est bien celle qu'elle prétend être. Il a pour but de garantir une authentification unique par session pour l'ensemble de services sollicités.

- *Le service d'autorisation* : Lors de l'ouverture de sa session, un utilisateur demande un ensemble de services et donc des composants de services. Le service d'autorisation intervient à ce niveau pour autoriser (ou interdire) une personne à utiliser chaque composant de service. En fait, il évalue les droits de l'utilisateur et il accorde les permissions selon les droits associés à son rôle.

- *Le service de session* : Il assure la création et l'activation de la session et la génération de l'identifiant de la session Session-ID (après une authentification réussie). Ce service assure la gestion et le maintien de la session qui se veut sécurisée de bout en bout.

- *Le service Token* : Il permet de générer et de mettre à jour le jeton d'authentification.

- *Le service VPSN et VPDN* : Le service VPSN gère le réseau des services interagissant ensemble pour offrir le service global demandé par l'utilisateur. Le service VPDN gère l'ensemble des terminaux de l'utilisateur en considérant chaque terminal comme un service. Ils génèrent respectivement les identifiants VPSN-ID et le VPDN-ID. Ces identifiants sont maintenus uniques durant la session, malgré la régénération du jeton, pour garder une session unique.

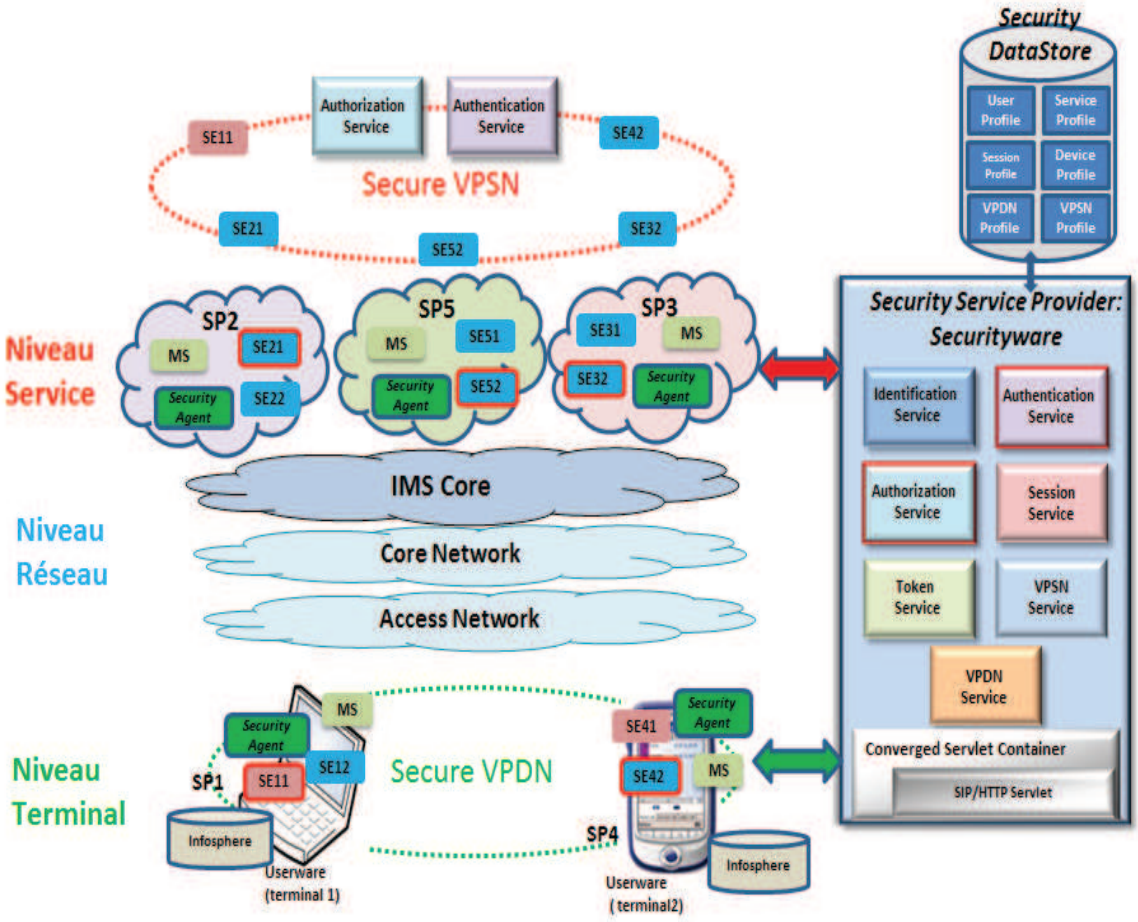



***Figure 11*** **:** Architecture de la sécurité.

Afin de faciliter et simplifier la gestion de sécurité et le contrôle d'accès, nous proposons aussi le *Security Agent* qui est déployé sur chaque plateforme de service ou terminal qui est considéré simultanément comme une plateforme de service et comme un service. Le rôle du *Security Agent* est d'intercepter les demandes de l'utilisateur dirigées vers la ressource protégée (service ou terminal) et de dialoguer avec le *Securityware* pour vérifier si l'utilisateur a le droit d'y accéder ou non. Donc, il protège l'accès aux plateformes de service en vérifiant les autorisations auprès du *Securityware.*

Enfin, le *Security Datastore* contient toutes les informations requises par le *Securityware*, notamment, il représente la base informationnelle contenant le profil de l'utilisateur, le profil du terminal, le profil du VPDN, le profil du VPSN, le profil de la session et le profil du service.

D'une part, notre modèle architectural de sécurité facilitera une agrégation verticale sécurisée entre des composants « terminal » et des composants « service-applicatif ». En effet, la vue verticale représente la session de l'utilisateur devant respecter l'approche « User Centric ». Cette approche vise principalement à mettre tout le système « au service » de l'utilisateur et elle s'oppose à celle ou l'utilisateur qui doit se plier aux différentes contraintes de connexions (Network Centric) ou de traitement (Application Centric). Donc, la session doit être établie dynamiquement suivant les préférences de l'utilisateur et les possibilités de son environnement durant ses déplacements. C'est cette session qu'il nous faut sécuriser, sans compliquer la tâche à l'utilisateur à chaque fois qu'il rajoute un service à sa session. En fait, notre socle de sécurité exposé par le *Securityware* correspond à une plateforme spécialisée

regroupant un ensemble de composants de sécurité qui traitent l'identification, l'authentification, l'autorisation, etc. Ainsi, le *Securityware* permet de gérer la sécurité non seulement au niveau des composants des services applicatifs, mais aussi au niveau de l'ensemble des terminaux qui forment le réseau ambiant personnel de l'utilisateur, tout en déployant des *Security Agents* partout dans les terminaux considérés comme des plates-formes de services. Ceci permet ainsi d'assurer une agrégation verticale des niveaux service et terminal intervenant dans la session.

D'autre part, notre modèle architectural de sécurité fournira aussi une composition horizontale sécurisée entre les composants de service sollicités par l'utilisateur au niveau service ainsi qu'au niveau terminal. En effet, d'une vue horizontale, le «*Securityware* » ainsi que le « Security Agent » interviennent lors de la composition des services pour construire le VPSN à partir des services « terminal » ou des services applicatifs en assurant un accès sécurisé à l'ensemble de ces services.

## 4.2. Services de base de sécurité

Nous décrivons les fonctionnalités des trois composants de sécurité de base (identification, authentification et autorisation). Chaque composant est représenté par le modèle préconisé dans UBIS « Black Box ». Nous nous intéressons essentiellement au plan d'usage qui décrit les opérations exécutées par le composant suite aux requêtes des utilisateurs.

### 4.2.1 Le service d'identification

Le service d'identification a comme tache d'attribuer un identifiant à chaque utilisateur pour qu'il puisse accéder à ses services et soit reconnu dans le système. Comme tout service, il est mutualisable : il peut être utilisé dans n'importe quel contexte et par plusieurs utilisateurs. C'est un service générique qui peut offrir à l'utilisateur la possibilité de disposer de plusieurs modes d'identification selon les besoins et les préférences de l'utilisateur.

- ***Aspect fonctionnel***

Le composant de service d'identification (Figure 12) s'appuie sur les éléments fonctionnels suivants :

- ***Recherche ()***: il vérifie si le pseudo existe dans la base de données. S'il ne trouve pas ce pseudo, les opérations mentionnées ci-dessous sont exécutées successivement. Dans le cas contraire, nous recevons une réponse négative (NOK), ce qui veut dire que l'utilisateur possède déjà un identifiant.

- ***Créer_login ()*** : le composant crée un login pour l'utilisateur.

- ***Créer_Password ():*** le composant crée un mot de passe.

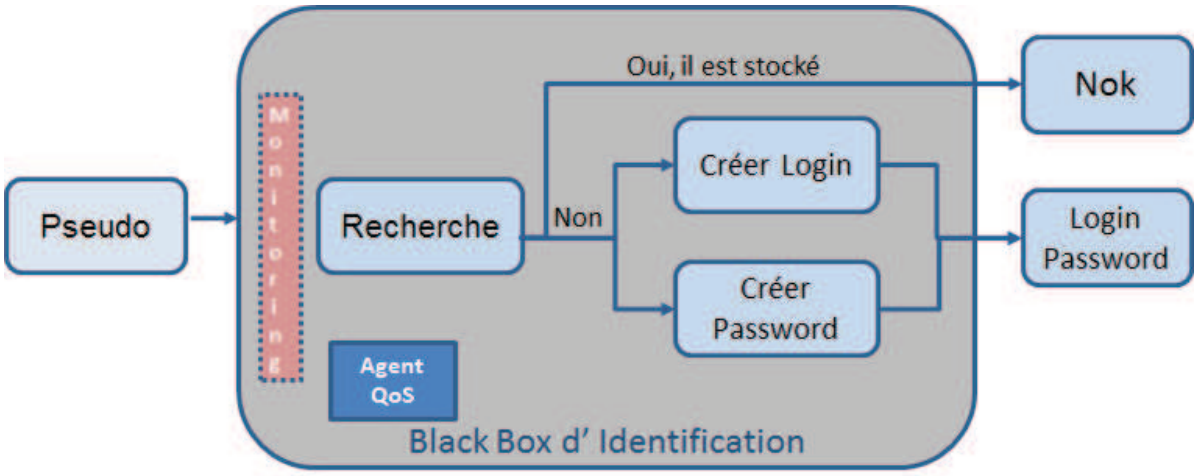



***Figure 12 :*** *Black Box du service d'identification.*

Selon l'enchainement des opérations précédentes, le résultat en sortie peut être un login, un couple (login / password), ou un login et plusieurs mots de passe.

- ***Aspect non-fonctionnel***

Le composant d'identification est autonome et auto-gérable puisqu'il intègre le service de Monitoring qui permet de collecter et mesurer les paramètres liées au comportement du service au niveau du plan d'usage, par exemple le taux de rejet des requêtes d'identification et le temps de traitement. Au niveau du plan de contrôle, l'agent QoS permet d'évaluer les critères à partir des mesures effectuées par le service de Monitoring et de signifier si le contrat est respecté ou pas.

### 4.2.2 Le service d'authentification

Le service d'authentification garantit l'authentification unique à l'utilisateur durant sa session. Il vérifie les accréditations de l'utilisateur et une fois l'authentification a réussi, le service Token va générer un jeton contenant l'identité de l'utilisateur et par la suite ce dernier n'aura plus besoin de se ré-authentifier à chaque demande d'un service.

- ***Aspect fonctionnel***

Le composant de service d'authentification (Figure 13) repose sur les éléments fonctionnels suivants:

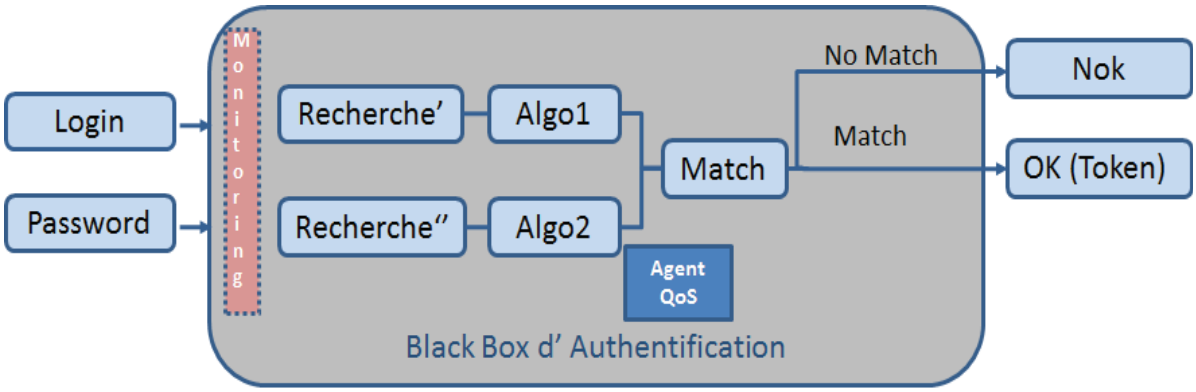



***Figure 13 :*** *Black Box du service d'authentification.*

- ***Recherche' ()*** : permet de retrouver le password enregistré dans la base données qui correspond au login de l'utilisateur.

- ***Recherche'' ()*** : permet de retrouver les éléments cryptographiques nécessaires de l'authentification comme la clé de déchiffrement pour les algorithmes exécutés

- ***Algo1 ()***: exécute un algorithme appliqué aux accréditations de l'utilisateur qui permet, par exemple, de déchiffrer le password retrouvé par l'opération *Recherche'()*.

- ***Algo2 ()***: exécute un algorithme pour décrypter le password en appliquant la fonction de hachage par exemple en utilisant la clé de déchiffrement retrouvée par l'opération *Recherche'' ()*.

- ***Match ():*** permet de comparer les paramètres d'identification fournis par l'utilisateur avec ceux enregistrés dans la base après. Si le résultat de la comparaison est positif, le service Token génère le jeton d'authentification, sinon la réponse sera un message (NOK).

  - ***Aspect non fonctionnel***

Ce service d'authentification est mutualisable suivant le niveau de sécurité demandé vu qu'il est capable de servir indifféremment plusieurs sessions actives de différents utilisateurs. Il est interopérable, il peut coopérer avec d'autres composants de service comme le service d'identification, le service Token et le service d'autorisation. Il est autonome et auto-gérable puisqu'il s'auto-surveille par son service de Monitoring et il contrôle son comportement durant l'exécution par son agent de QoS.

### 4.2.3 Le service d'autorisation

Pour avoir une session sans couture, un utilisateur doit pouvoir accéder à n'importe quel moment, de n'importe quel lieu et de n'importe quel terminal à ses services. Pour ce faire, nous faisons appel à la notion du VPSN. Avant d'ouvrir la session d'une partie, on lui construit un VPSN qui regroupe l'ensemble des composants de services utilisés ou qui peuvent être utilisés par cette partie durant sa session.

Les composants de services faisant partie du VPSN doivent impérativement être activables et peuvent être activés et utilisés à n'importe quel moment par la partie. Ainsi, le VPSN ne doit contenir que des composants de service auxquels la partie est autorisée. Ceci veut dire que la demande d'autorisation de la partie auprès de tous les composants de services auxquels elle est abonnée et auxquels elle désire accéder durant sa session, doit être faite ***une et une seule*** fois au moment de la construction de son VPSN. Ceci justifie l'appellation « service d'autorisation unique ».

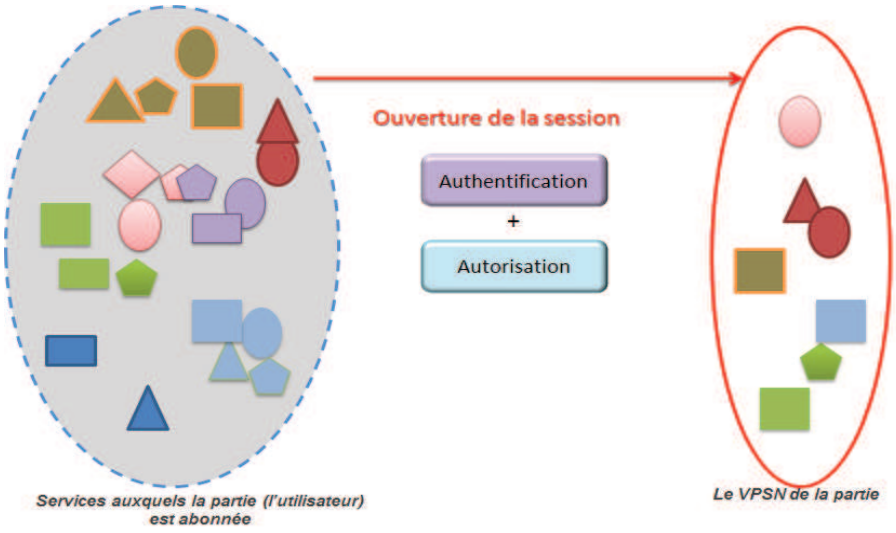



***Figure 14*** *: Le service d''autorisation unique.*

Lors de l'initialisation de sa session, l'utilisateur sollicite des services selon ses besoins. Pour pouvoir accéder aux différents composants de services demandés, il doit avoir les droits nécessaires. Alors, le service d'autorisation permet de lui attribuer les droits d'accès à chaque composant de service ou de lui interdire (Figure 14).

L'exécution de ce service nécessite deux éléments à savoir l'identité de l'utilisateur demandant l'autorisation d'accès et l'identité du composant de service. Si la réponse est positive(OK), le composant de service est inséré dans le VPSN de l'utilisateur. Sinon, la demande d'accès au service est refusée (NOK).

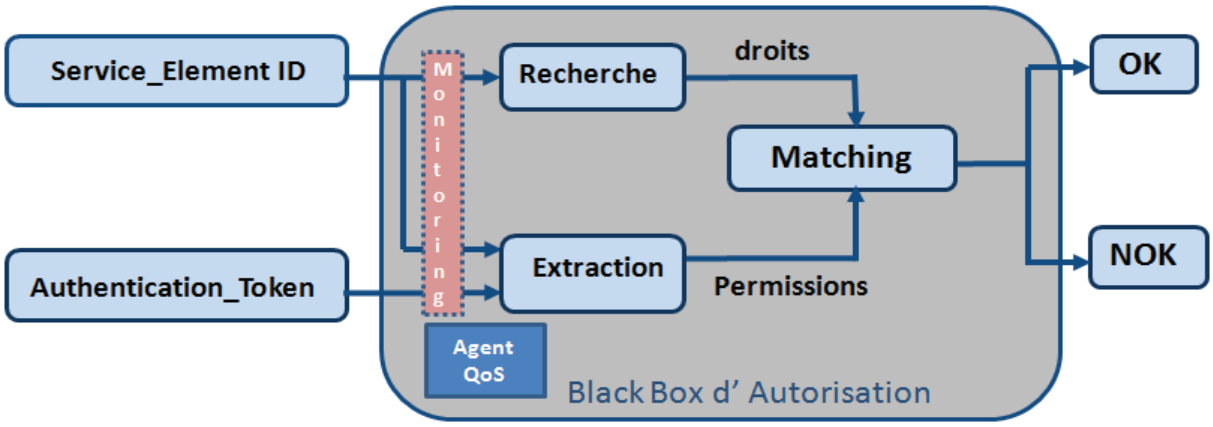



***Figure 15 :*** *Black Box du service d'autorisation.*

- ***Aspect fonctionnel***

La fonction du composant autorisation est réalisée par trois opérations (Figure 15):

- L'opération « Search » : recherche la liste des droits offerts par le composant de service (par rapport auquel l'autorisation est faite). Cette opération a besoin juste de l'identifiant de ce dernier en entrée pour qu'elle puisse fonctionner. En sortie, cette opération donne la liste des droits offerts par le composant de service.

- L'opération « Extract » : extrait les permissions donnés à un utilisateur (dont l'identité est stockée sur le jeton d'authentification) sur le service en question par l'un (ou plusieurs) des rôles joués par cet utilisateur. L'opération commence par l'extraction de l'identité de l'utilisateur à partir du jeton. Ensuite, elle recherche le profil de l'utilisateur pour trouver lequel parmi ses rôles celui qui lui donne accès au composant de service en question. Une fois elle trouve le rôle, l'opération « Extract » donne en sortie la liste de permissions définies par rapport à ce rôle sur le composant de service.

- L'opération « Matching » : compare entre les droits offerts par le composant de service en question et les permissions définies par le rôle de l'utilisateur sur ce composant. Deux cas de figure se présentent :

  - *Si la liste de droits offerts par la ressource (composant de service) > liste des permissions données par le rôle:* dans ce cas de figure, si on autorise l'utilisateur, la ressource va lui permettre de faire des actions auxquelles elle n'est pas autorisée selon la stratégie de son organisation. C'est pourquoi l'opération « Matching » donne une réponse négative en sortie.

  - *Si la liste de droits offerts par la ressource < liste des permissions données par le rôle :* la ressource va imposer son mode d'utilisation et les droits en plus décris par le rôle ne vont pas être pris en compte. Du coup, l'utilisateur est autorisé à utiliser la ressource. L'opération « Matching » donne ainsi une réponse positive.

- ***Aspect non-fonctionnel***

Comme tout service dans notre architecture, le service d'autorisation est mutualisable, interopérable avec les autres services et auto-gérable. Il contient un service de Monitoring qui permet de surveiller, par exemple, les tentatives et les échecs d'accès non autorisés, et un agent QoS qui évalue les critères à partir des mesures pour surveiller le comportement de ce service.

## 4.3. Services de support de sécurité

Nous traitons les aspects de sécurité à partir des deux niveaux de visibilité : le niveau « terminal » et le niveau « service ». D'une part, nous montrons comment assurer un usage sécurisé des différents terminaux de l'utilisateur dans une session continue en introduisant le concept du « device as a service » et du VPDN (§4.3.1). D'autre part, nous décrivons la

manière d'offrir une composition de service dynamique sécurisée au sein du VPSN dans un contexte « user-centric » (§4.3.2).

### 4.3.1 Du côté du terminal : VPDN

Actuellement, l'utilisateur NGN/NGS possède plusieurs terminaux à partir desquels il peut accéder et utiliser ses services. Il les considère comme une seule et même ressource. Il peut les avoir à sa disposition là où il le désire et passer de l'un à l'autre en fonction de ses besoins. Nous devons gérer ce cas de figure avec la gestion de la mobilité de session. Le VPDN (Virtual Private Device Network) représente le réseau des terminaux qui sont présents dans le PAN (Personal Area Network) de l'utilisateur et utilisés durant sa session de service. Il est conçu pour être responsable de la gestion et de l'agrégation de ces terminaux afin de préserver la continuité de service en cas de changement de terminal. Il permet aussi d'assurer la liaison et la reconnaissance entre les différents terminaux d'un même utilisateur. Ces terminaux peuvent se communiquer et échanger des données ou des services.

Dans le VPDN, chaque terminal est, d'une part, considéré comme un composant de service (« device as a service ») rendant un service prédéfini et capable de communiquer avec d'autres composants (terminaux). Il définit des fonctionnalités de personnalisation et d'adaptation et utilise le profil de l'utilisateur en temps réel pour gérer ses sessions en se basant sur ses préférences et sa mobilité ainsi que le contexte ambiant.

D'autre part, chaque terminal (*Userware*) héberge un ensemble de service, il peut donc être considéré comme une plate-forme de service ou un fournisseur de service. En fin de compte, il agira comme un marché ouvert de service où les utilisateurs pourront vendre (offre) et/ou acheter (demande) des services les uns des autres. De plus, le *Userware* rassemble l'ensemble de l'outillage de l'utilisateur pour gérer ses sessions en fonction de ses préférences et de sa mobilité. Il contribue à construire des applications par composition de services. En fait, il offre un ensemble de services « terminal » (par exemple : service d'affichage, service clavier et service écran tactile) et il permet de déployer d'autres services applicatifs (par exemple : service de localisation). Par conséquent, le terminal joue un double rôle : il est un composant de service géré par le VPDN et il représente une plateforme de service.

Pour garantir la sécurité au niveau terminal de l'architecture, nous allons appliquer notre vision orientée service basée sur des composants de service génériques, mutualisables, stateless, et autogérables exposé dans notre fournisseur de services de sécurité dénommé *Securityware*.

En fait, la sécurité a un spectre très large, mais dans notre contexte, il s'agit de s'assurer de la cohérence et de la fiabilité de trois fonctions importantes, à savoir : L'identification, l'authentification et l'autorisation. Nous décrivons ci-dessous un scénario basé sur une utilisation d'un seul terminal (aussi considéré comme plate-forme de service) qui va construire le VPDN. Ainsi, nous mettons l'accent sur la sécurité et nous montrons comment les composants de service de sécurité interviennent dans ce cas (Figure 16). Dans le cas où nous aurions plusieurs terminaux, la création du VPDN est partagée.

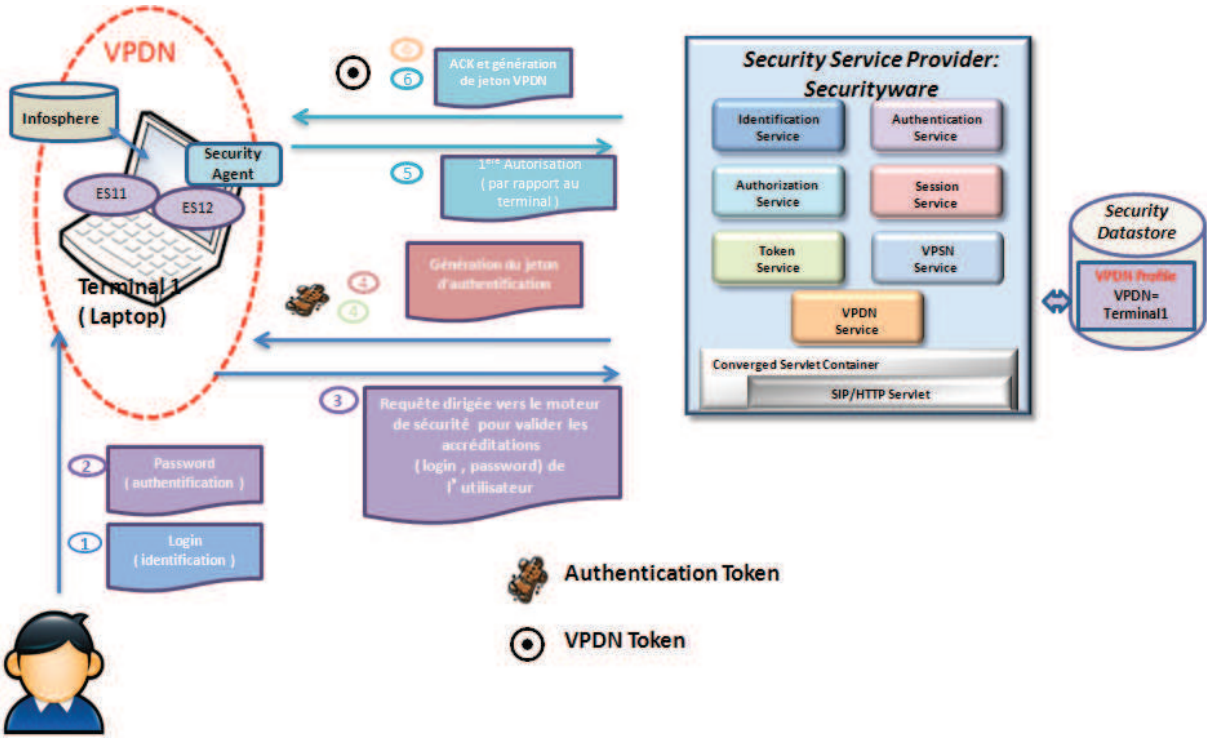



***Figure 16 :*** *Le VPDN et la sécurité.*

Lors de l'utilisation du terminal, le *Security Agent* sollicite le *Securityware* pour activer le service d'identification. En effet, l'utilisateur doit éventuellement s'identifier par une simple déclaration "*login*" qui est couplé avec un mot de passe pour des raisons de sécurité dans notre système. Si le login existe déjà dans la base de données *Security Datastore*, le *Security Agent* du terminal reçoit de la part du *Securityware* une réponse qui veut dire que l'utilisateur possède déjà un identifiant. Dans le cas contraire, le service d'identification crée un login et un password après l'inscription de l'utilisateur.

Après la phase d'identification, le système demande à l'utilisateur de s'authentifier, l'utilisateur tape son password. La demande est dirigée auprès du *Securityware*. Par la suite, le service d'authentification vérifie les accréditations (login/password) fournis par l'utilisateur par rapport aux informations enregistrées dans la base des données *Security Datastore*.

Une fois l'utilisateur authentifié, l'initialisation de la session sera effectuée par le Service de Session qui prend en charge la génération des attributs de la session courante. Ce service assure la gestion et le maintien de la session qui se veut sécurisée de bout en bout. Ce composant interroge le Service Token qui permet de générer le jeton de session contenant les informations nécessaire liées à la session et à l'utilisateur.

Par la suite, le jeton est transmis par le *Securityware* au *Security Agent* du terminal. Sa transmission a lieu dans un canal sécurisé. Grâce à ce jeton, l'utilisateur pourra utiliser, selon ses droits, n'importe quel service de différentes plateformes de service pour composer sa session sans avoir besoin d'une réauthentification de sa part.

Pour finir, l'utilisateur, est-il autorisé à utiliser le terminal et de créer son VPDN ? A-t-il le droit ?

Donc, une demande d'autorisation par rapport à l'utilisation du terminal est envoyée par le *Security Agent* auprès du *Securityware*. Le Service d'autorisation évalue les droits effectifs

sur la base des informations (identité et preuves d'authenticité) fournies par le Service d'authentification. Une réponse positive ou négative est envoyée auprès du *Security Agent* en fonction des droits associés au rôle de l'utilisateur et en fonction des privilèges liés à la ressource. Une fois, l'utilisateur est autorisé, le Service VPDN génère le "VPDN-ID" qui est l'identifiant de l'ensemble des terminaux sollicités par l'utilisateur durant sa session et qui construisent le VPDN. Le Service Token récupère l'identifiant du VPDN et met à jour le jeton de session. Le terminal est enregistré dans le profil du VPDN

### 4.3.2 Du côté des services : VPSN

Dans notre contexte, l'utilisateur est le point central du dispositif, l'objectif principal est de lui offrir une session unique personnalisée et sécurisée, tout en lui permettant d'accéder à ses services d'une manière continue, malgré l'hétérogénéité de son environnement et la mobilité. Pour atteindre ce but, la gestion de de la session de service ainsi sa sécurité devra être assurée autrement. Ainsi, notre architecture proposée offre une composition de service sécurisée aux utilisateurs finaux avec des changements dynamiques en tenant compte de la mobilité, des exigences de QoS et de sécurité et des offres de services disponibles.

Une composition de service sécurisée au niveau du plan service représente un VPSN. Il inclut l'ensemble des composants de service utilisés par un utilisateur durant sa session basée sur l'approche « user-centric ». Alors, les services sont composés dynamiquement en fonction des préférences de l'utilisateur. Le VPSN gère les interactions entre ces composants via des liens en respectant une logique de service. Nos services de sécurité sont parmi les services de gestion dans le VPSN et qui sont transparents à l'utilisateur et visent à sécuriser sa session.

Durant une session « user-centric », l'utilisateur est invité à composer dynamiquement les services dont il a besoin pendant une période du temps. Par exemple, il peut ajouter un service vidéophone à son appel vocal ou transférer un programme TV de sa télé à son PDA en se déplaçant. Tous ces changements sont assurés d'une manière transparente « sans couture » au sein d'une session unique et sécurisée.

Pour créer le VPSN de l'utilisateur à partir de son terminal, le *Security Agent* envoie une demande d'autorisation auprès du *Securityware* pour la création du VPSN. Si la réponse est positive, le jeton est mis à jour par le Service Token et complété par le VPSN_ID généré par le Service VPSN. Par conséquent, la session de service est initialisée par le terminal après l'autorisation et la génération du VPSN-ID.

Par la suite, l'utilisateur est capable de composer les services exposables qu'il désire avoir dans sa session selon ses préférences et ses besoins. Il peut choisir ces services de diverses plates-formes selon ses droits préétablis dans son contrat. Ainsi, le jeton est transféré à chaque *Security Agent* protégeant une plateforme de service visitée afin de préserver l'authentification unique de l'utilisateur durant sa session.

Pour chaque service sollicité par l'utilisateur, une autorisation est donnée cas par cas. Le Service d'Autorisation est inclus dans le VPSN comme un service de gestion pour accomplir l'autorisation d'accès aux services. Il interagit avec les *Security Agents* des différentes plateformes de services requises pour vérifier la correspondance des droits de l'utilisateur et des privilèges liés aux services désirés en s'appuyant sur l'identité de l'utilisateur qui demande l'accès au composant de service en question (par rapport auquel cette autorisation est faite). A chaque passage d'une plateforme à une autre, le *Security Agent* examine la

validité du jeton. L'autorisation donne une réponse positive (OK) ou négative (NOK). Selon cette réponse le composant en question sera inséré ou non dans le VPSN de l'utilisateur.

Le Service d'Authentification est aussi inclus dans le VPSN en cas où une authentification plus forte est requise. Par exemple, si un des services choisis dans le VPSN est un service bancaire (payement), alors il faut une authentification forte.

Les composants d'Authentification et d'Autorisation sont des services de base de gestion qui participent au pré-provisionnement et le provisionnement des services qui composent le VPSN. Ces composants accomplissent leurs rôles d'une façon transparente sans l'intervention de l'utilisateur. Finalement, le VPSN crée est enregistré dans le profil VPSN.

Pour atteindre une gestion et un contrôle automatisés et distribués de la sécurité au niveau service de l'architecture, nous allons appliquer notre vision proposée de la sécurité fournie comme service basée sur des composants de service génériques, mutualisables, stateless et auto-gérables. Afin de visualiser la flexibilité et la dynamicité de la composition de service sécurisée pour construire la session de service (VPSN), nous allons proposer un scénario, comme le montre la Figure 17, qui se base sur une configuration où nous avons plusieurs plates-formes de service (le terminal est également considéré comme une plate-forme de service). Par conséquent, la création du VPSN est partagée, chacune des plates-formes le complétant avec les services dont elle a la responsabilité.

Dans ce scénario Bob est à la maison, il utilise son ordinateur portable. Selon ses besoins et ses préférences spécifiés dans son contrat, il veut afficher une vidéo en haute définition résolution (ES11 : Elément de service 11), consulter ses e-mails (ES21) et recevoir tous les texto (SMS) sous forme de message vocaux (ES31)

- ***Composition de service sécurisée***

Après la vérification des autorisations pour chaque service sollicité par Bob, une composition de service sécurisée est effectuée et sa session de service (VPSN) est provisionnée. Ainsi, Bob dispose d'un VPSN qui contient tous les services qui répondent à sa demande en fonction de ses préférences et ayant le niveau de sécurité requis. Nous rappelons que SE11 est fourni par le fournisseur de service SP1 (Userware: terminal), SE21 est fourni par le fournisseur de services SP2, SE31 est fourni par le fournisseur de services SP3 et les services de sécurité sont fournis par le *Securityware*.

Ces cinq services construisent le VPSN suivant une logique de service indiquant l'ordre dans lequel ils doivent être exécutés.

VPSN (Bob) = Authentification Service +Autorisation Service + SE11 + SE21 + SE31

Les services de sécurité sont impérativement interconnectés et interopérables avec les autres services afin de fournir un service global sécurisé. Ces services de sécurité interagissent également avec les *Security Agents* qui vérifient la validité de jeton lors de chaque passage d'une plate-forme de service à une autre pour assurer un accès sécurisé et préserver l'unicité de la session.

La composition de service permet l'interopérabilité, l'interaction et la coopération entre les différents composants de service.

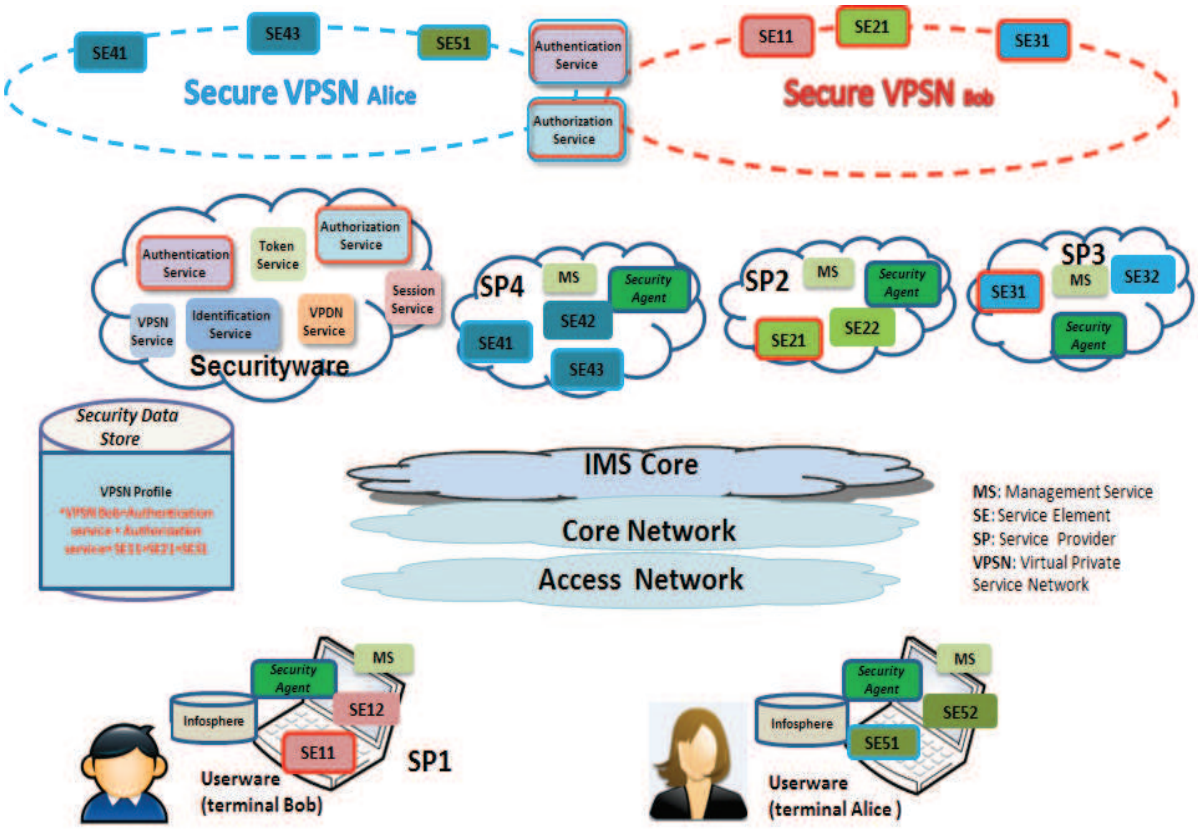



***Figure 17 :*** *Mutualisation et composition de services de sécurité*

- ***Les composants de service de sécurité sont Mutualisables, génériques et stateless***

Un composant de service de sécurité est mutualisable. Cette propriété fournit un moyen d'optimiser les ressources en les partageant de manière efficace entre les différents utilisateurs. En effet, le composant de service peut être mutualisé parmi plusieurs VPSN et utilisé dans différents contexte grâce aussi à sa généricité. En outre, plusieurs tranches du même ES peuvent être trouvées dans le même VPSN, chacun d'eux servant à remplir la même fonction, mais avec des cibles différentes, avec ses propres exigences et contrat.

Prenons par exemple le cas du Service d'Authentification qui peut être partagé entre deux utilisateurs dans deux contextes différents. Le premier utilisateur Alice a besoin d'une authentification forte pour faire des transactions bancaires ; le deuxième utilisateur a besoin uniquement d'une authentification simple pour regarder un film. Pour ce faire, le composant de service de sécurité, en particulier le Service d'Authentification, doit être stateless. Ceci signifie que le composant de service ne doit pas enregistrer l'historique et conserver les informations d'état relatives aux clients qui l'ont invoqué. Cette propriété nous amène à avoir des services ubiquitaires qui peuvent être déployés dans n'importe quel environnement. Ceci rend également possible l'utilisation des différents services fournis par différents plates-formes (fournisseur) de services. Par conséquent, nous pouvons passer d'un service à l'autre, même s'ils n'appartiennent pas au même SP ou ne sont pas déployés sur la même plateforme. Par conséquent, la composition de services devient plus efficace et plus flexible.

- ***La continuité de la session de service et de la sécurité : les services de sécurité sont auto-gérables***

Tout changement dans le contexte de l'utilisateur (mobilité, préférences), ainsi que les changements liés à un composant de service (disponibilité, les attaques malveillantes, les failles de sécurité) peut provoquer l'ajout, la suppression ou le remplacement d'un ou plusieurs composants faisant partie du VPSN. Durant la session « user-centric », chaque composant de service participant à cette session s'autogère. Cette solution permet une décentralisation et une automatisation du contrôle et de la gestion des services. L'apport majeur de cette solution est de détecter la défaillance au bon moment durant les changements et également de permettre d'approvisionner et de réapprovisionner les ressources services sans causer une interruption durant l'usage des services grâce à un service de découverte périodique qui maintient un ensemble de services ubiquitaires qui sont fonctionnellement équivalents aux services construisant le VPSN actif. Cette découverte est lancée à toutes les plates-formes de services fédérées qui sont dans le même cercle de confiance. En fait, la recherche est basée sur les trois critères suivants:

i) Les services qui appartiennent au cercle de confiance,

ii) Les services qui répondent à la fonctionnalité requise, et

iii) les services auxquels l'utilisateur a les droits d'accès selon son rôle.

Ainsi, nous allons préserver la sécurité du VPSN et la continuité de la session de l'utilisateur.

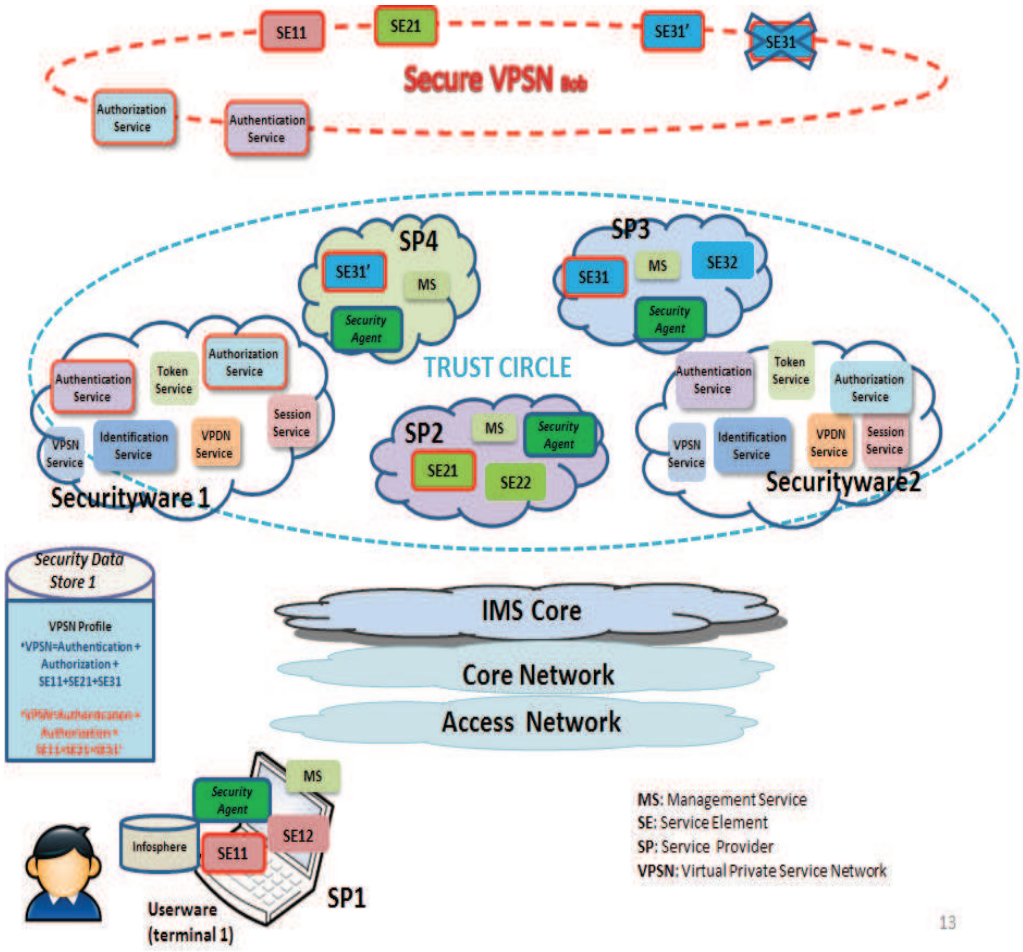



***Figure 18 :*** *Auto-gestion de services de sécurité*

Pour mieux clarifier ces aspects, prenons l'exemple suivant (Figure 18) : nous supposons que SE31 devient indisponible suite à une faille de sécurité (attaque malveillante). Ainsi, une nouvelle composition de services est nécessaire pour remplacer ce service. Le composant indisponible, qui a été attaqué, doit être remplacé par un autre qui lui est fonctionnellement équivalent et ayant capable de fournir le niveau de sécurité requis. Par conséquent, le composant de service SE31 est remplacé par « SE31' », qui a été déjà découvert. Il appartient au fournisseur de services de confiance SP4.

Par conséquence, une nouvelle composition est mise en place et la logique de service devient comme suit:

VPSN (Bob) = Service d'Authentication + Service d'Autorisation + SE11 +SE21 + SE31 '

Ceci s'applique notamment aux services de sécurité qui sont aussi auto-gérables. Si le service ne remplit pas ses fonctions correctement, il va être remplacé par un autre service ubiquitaire fonctionnellement équivalent et qui a le même niveau de QoS préétabli dans son contrat.

## 4.4. Services de gestion de sécurité : le service d'audit

### 4.4.1 Audit : Définition

L'audit est un processus méthodique, indépendant et documenté permettant de recueillir des informations objectives pour déterminer dans quelle mesure les exigences satisfont les référentiels du domaine concerné. Il s'attache notamment à détecter les anomalies et les risques dans les organismes et secteurs d'activité qu'il examine.

En effet, l'audit désigne l'observation, l'enregistrement, l'analyse et la compréhension des évènements importants ou critiques qui vont concourir à constituer le fil de son histoire, après la constatation d'une panne ou d'une attaque. Dans la pratique, les évènements sont enregistrés dans des journaux qui sont des témoins de confiance chargés d'interpréter la trame des opérations et d'imputer la responsabilité d'une erreur ou d'un acte malveillant à son initiateur.

Il convient de différencier l'audit de la notion d'*auditabilité* qui fait référence à la traçabilité. Elle se définit par la capacité d'un système à garantir la présence des informations nécessaires à une analyse ultérieure d'un événement courant ou exceptionnel dans le but de déterminer s'il y a effectivement eu violation de la sécurité, et dans ce cas, quelles informations ou autres ressources ont été compromises. C'est également la fonction destinée à déceler et à examiner les événements susceptibles de constituer une menace pour la sécurité. Cependant, l'audit ne se limite pas à la traçabilité.

### 4.4.2 De l'évaluation du comportement de service (QoS)…

Pour pouvoir offrir un meilleur service personnalisé selon les préférences fonctionnelles et non-fonctionnelles de l'utilisateur, il faut évaluer le comportement des composants de services tout en s'appuyant sur les conditions contractuelles préétablies avec le fournisseur de service.

En effet, un contrôle est défini selon l'ISO 9000 comme étant : « *l'**évaluation** de la conformité par observation et jugement accompagné, si nécessaire, de mesurages, d'essais ou de calibrage* ».

D'après De Ketele, *« Evaluer consiste à **recueillir** un ensemble d'informations reconnues comme suffisamment **pertinentes**, **valides et fiables**, et à examiner le degré d'adéquation entre cet ensemble d'informations et un ensemble de **critères** jugés suffisamment adéquats aux **objectifs fixés** au départ ou ajustés en cours de route, en vue de fonder une prise de **décision*** ».

Evaluer le comportement d'un composant de service commence par la gestion de la QoS en prévoyant les paramètres mesurables et se poursuit lors de l'exploitation en gérant les dysfonctionnements et les variations de QoS. La qualité de service de bout en bout d'un composant de service reflète la qualité globale perçue par l'utilisateur final. Cependant, la QoS au niveau des composants garde une importance pour la conception et la gestion des services, étant donné qu'elle participe à fournir et maintenir une QoS de bout en bout.

L'agent de QoS se charge de contrôler la conformité et d'évaluer le comportement (QoS) d'un composant de service à travers les trois types de valeurs mesurables telles que les valeurs seuils, les valeurs de conception et les valeurs courantes en se basant sur les mesures effectuées par le service Monitoring.

➢ **Exemple : Evaluation du comportement d'un service d'authentification**

Rappelons que la sécurité est fournie comme un service (dans cette thèse). Cela veut dire que chaque composant de service de sécurité comporte une partie fonctionnelle et autre non-fonctionnelle dès sa conception. Nous allons donc évaluer le comportement la conformité d'un composant de service de sécurité. Nous prenons comme exemple le service d'authentification. Nous présentons, dans le tableau ci-dessous, les paramètres mesurables de QoS liés au composant de service d'authentification pour pouvoir évaluer les quatre critères spécifiés par le modèle « DFDC »: disponibilité, fiabilité, délai et capacité. Ce qui nous donne après la possibilité d'évaluer sa conformité selon les conditions contractuelles et les règles préétablies.

| ***Critères QoS*** | ***Métriques QoS : Mesures (Service Monitoring)*** |
|---|---|
| ***Disponibilité*** | **Taux d'échec d'authentification** : Nombre de requêtes d'authentification rejetées / nombre de requêtes totales |
| ***Fiabilité*** | **Taux d'authentification erronées :** Nombre de requêtes erronées (tentatives d'authentification) / nombre de requêtes totales |
| ***Délai*** | **Temps de traitement des requêtes d'authentification (**validation des accréditations) |
| ***Capacité*** | **Requêtes d'authentification /seconde** |

***Tableau 3 :*** *Mesures de QoS appliquées au service d'authentification.*

En résumé l'évaluation du comportement du service d'authentification repose sur la mesure effectuée par le service Monitoring intégré dans ce service. Le résultat est transféré à l'Agent QoS qui se charge d'évaluer la QoS en temps réel pour détecter la défaillance au bon moment durant les changements (mobilité, préférences, dégradation). L'Agent QoS compare la valeur courante de QoS du composant de service à sa valeur seuil (à ne pas dépasser). Dans le cas où cette valeur est inférieure à la valeur seuil il notifie par un « IN Contrat ». Dans le cas où cette valeur est supérieure à la valeur seuil, il notifie par un « OUT Contrat » et donc le service n'est plus conforme. Cette évaluation de comportement est une étape importante et essentielle dont il sera tenue compte pour la prise de décision vis-à-vis la conformité de service d'authentification selon le contrat préétabli et peut être utilisée pour la sécurité.

Cette phase d'évaluation du comportement (QoS) de composant d'authentification est nécessaire pour contrôler la conformité de service. Mais est-elle suffisante pour contrôler la sécurité de ce même composant et prendre la décision par rapport à un acte malveillant ou une attaque ?

### 4.4.3 ...vers un « Audit de sécurité »

L'évaluation du comportement de service est considérée comme un premier diagnostic d'un point de vue conformité (QoS) qui nous aide à prendre les bonnes décisions durant tout

le cycle de vie d'un service de la conception à l'exploitation ainsi que lors de la session utilisateur.

Cependant, cette évaluation se fait seulement par rapport au comportement d'un service mais non par rapport à la sécurité des services et des données générées et échangées qui doivent être protégés contre les attaques.

C'est pourquoi, nous proposons un audit de sécurité, complémentaire à l'évaluation du comportement, qui est basé sur la qualité de service permettant un meilleur diagnostique pour prendre la bonne décision lors d'un dysfonctionnement en général et d'une attaque de sécurité en particulier. Il permet de suivre la sécurité ainsi que la QoS durant l'exécution et au final préciser les actions nécessaires.

Il a pour vocation d'identifier les vulnérabilités existantes sur un système et d'évaluer l'efficacité des contrôles ou des services de sécurité mis en place. En effet, l'audit est le seul moyen pour valider les aspects sécuritaires d'un système dans le but de vérifier, tout d'abord, son niveau global de sécurité, et par la suite, de l'améliorer.

ITIL (la version 3) [38] définit trois règles de base dans la partie consacrée au processus d'amélioration continue des services pour réaliser un management efficace quel que soit l'activité :

- *« Vous ne pouvez pas gérer ce que vous ne pouvez pas **contrôler** »*
- *« Vous ne pouvez pas contrôler ce que vous ne pouvez pas **mesurer** »*
- *« Vous ne pouvez pas mesurer ce que vous n'avez pas **défini** au départ »*

Gérer la sécurité d'une manière efficace commence par préciser au préalable ses ***exigences.*** Autrement dit, sans objectifs claires et bien définis, les résultats ne seront pas au rendez-vous.

Il n'y a pas non plus de gestion efficace sans mesurage, nous allons voir ce que cela signifie.

Contrôler les services de sécurité a pour but de **prendre une décision** face à une situation donnée. La figure ci-dessous présente le schéma fonctionnel de l'audit de sécurité ainsi que les actions mises en place pour chaque objectif. Ceci est décrit en détail dans ce qui suit.

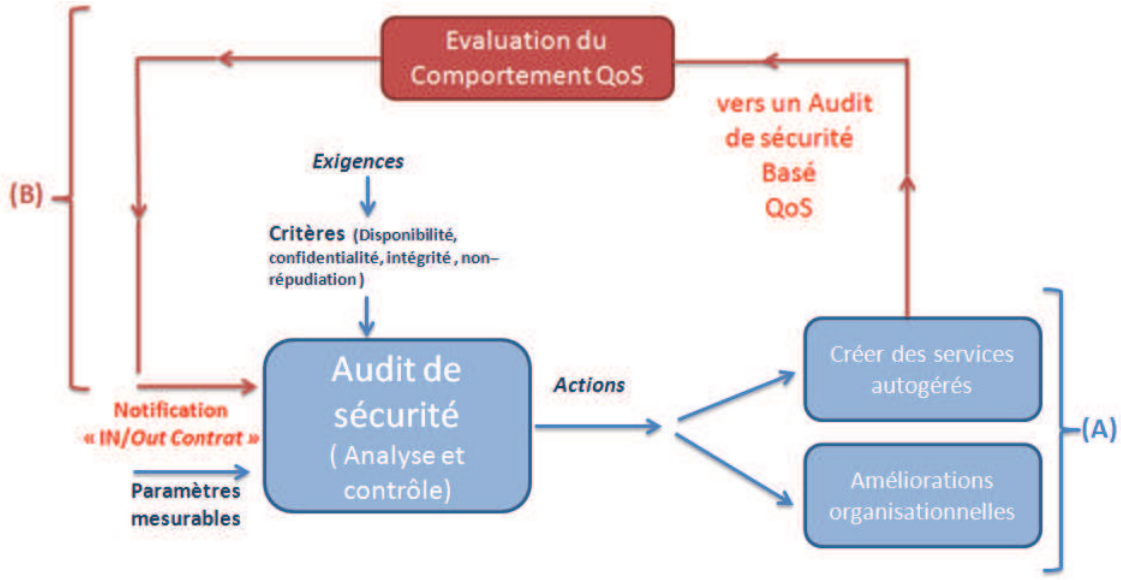



***Figure 19 :*** *Schéma fonctionnel de l'audit de sécurité.*

#### 4.4.3.1 Les exigences de sécurité : Critères génériques

➢ **Les exigences de sécurité : Des critères évaluables**

Fondamentalement, les besoins de sécurité d'une architecture ou d'un système sont traduits par les quatre **critères** de base à savoir la confidentialité, l'intégrité, la disponibilité et la non-répudiation (Figure 20).

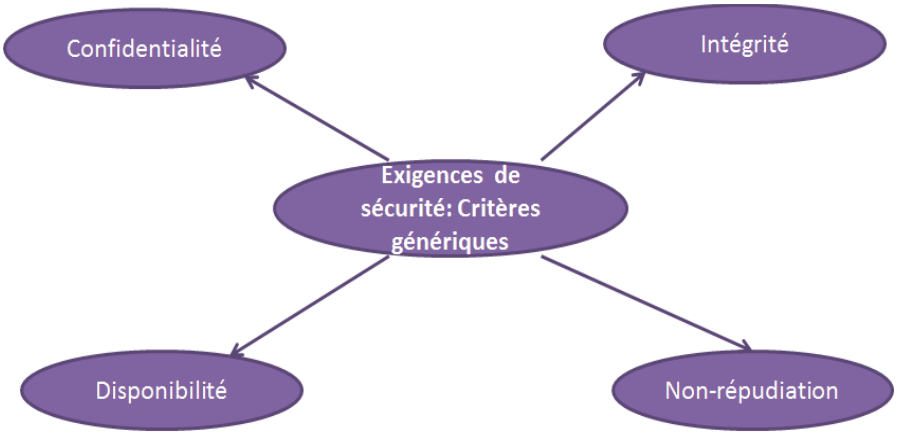



***Figure 20 :*** *Exigences de sécurité.*

- **Confidentialité**

La confidentialité est la protection des données transmises contre les attaques, comme la divulgation non autorisée, en assurant que seuls les utilisateurs habilités dans des contextes prédéfinis ont accès aux informations.

Afin de garantir la confidentialité des données durant l'échange, plusieurs niveaux de protections sont envisageables. En général, cette propriété protège toutes les données transmises entre deux utilisateurs durant une période de temps donnée. Un autre aspect que

la confidentialité peut garantir est la protection du flot de trafic contre l'analyse ou l'écoute clandestine. Par conséquent, un attaquant ne peut pas observer les sources et les destinations, ou autres caractéristiques du flux de données acheminées.

- **Intégrité**

L'intégrité assure qu'une information n'est modifiée ou détruite que dans des conditions prédéfinies selon des contraintes précises. Ces contraintes d'intégrité forment un ensemble d'assertions définissant la cohérence du système afin de garantir que les informations ou les données traitées ne sont modifiées que par une action volontaire et légitime. La meilleure approche est une protection totale du flux de donnés. Il existe deux types de service d'intégrité : orienté connexion et non orienté connexion. Le premier assure la protection contre la duplication, l'insertion, la modification, le rejeu et la destruction. Le deuxième fournit une protection uniquement contre la modification. Ainsi, l'intégrité consiste à garantir un maintien correct de l'échange des données entre les utilisateurs ou les processus sans avoir subi d'altération.

- **Disponibilité**

La disponibilité présente l'aptitude du système à remplir une fonction dans des conditions définies d'horaires, de délais et de performances. Elle doit pouvoir garantir que les tâches ayant des contraintes temporelles soient exécutées au bon moment. Il s'agit également de garantir que l'accès aux ressources est possible quand on en a besoin et que les ressources ne sont pas sollicitées ou conservées inutilement.

- **La non-répudiation**

Elle désigne le fait de ne pas pouvoir nier ou rejeter un événement (action, transaction) qui a eu lieu. A ce critère de sécurité est associée la notion ***d'imputabilité*** qui se définit par l'attribution d'une action (un événement) à une entité déterminée (ressource, personne). Elle est liée à la notion de responsabilité.

### ➢ Les métriques de sécurité

*« To measure is to know. If you cannot measure it, you cannot improve it. »*

*[Lord Kelvin (1824-1907)]*

*En s'appuyant sur la citation du physicien Lord Kelvin,* mesurer commence par savoir, c'est-à-dire, il faut tout d'abord définir les objectifs et les exigences. Dans notre cas, les exigences de sécurité sont définies par les critères énumérés dans la section précédente à savoir la confidentialité, l'intégrité, la disponibilité et la non-répudiation.

Pour pouvoir renforcer la sécurité, avoir une gestion de sécurité efficace et robuste du côté de fournisseur de service, et par la suite offrir le meilleur service de sécurité aux utilisateurs, il est judicieux de pouvoir « mesurer » le comportement des services mis à disposition selon les exigences de sécurité. C'est pourquoi, les critères de sécurité doivent être traduits en métriques et par la suite en paramètres mesurables qui vont nous servir après pour l'évaluation.

Dans ce qui suit, on va déterminer les différents paramètres mesurables à partir de la définition des critères de sécurité.

❖ *La confidentialité*

Plusieurs questions peuvent se poser du côté de l'utilisateur ou du fournisseur de service : L'information échangée n'est-elle connue que par l'émetteur et le récepteur ? L'information stockée est-elle accessible uniquement par les entités autorisées? Les informations d'authentification (mot de passe, certificat, etc.), circulant entre les utilisateurs et les fournisseurs de service sont-elles protégées contre la divulgation ?

Pour garantir la confidentialité, il faut s'assurer que tous les canaux de transport de données entre les plates-formes de service sont sécurisés, c'est-à-dire que tous les chemins que l'information peut prendre durant la session de service sont contrôlés. Pour des raisons d'optimisation, seulement un certain nombre de ces canaux sont généralement contrôlés.

On définit alors les **métriques** suivantes relatives à la **confidentialité** :

- Taux de divulgation d'information
- Probabilité de résistance aux attaques (analyse et écoute clandestine, divulgation d'information)
- Niveau de chiffrement

❖ *L'intégrité*

L'intégrité signifie l'exhaustivité, l'exactitude, la validité d'information. Donc, la question qui peut se poser de la part d'un fournisseur de service ou d'un utilisateur est : L'information reçue est-elle identique à celle émise? Le service est-il fiable? Les données sont-elles altérées?

On peut constater que **les métriques de la fiabilité** dans l'évaluation du comportement (QoS) de service peuvent nous servir pour **mesurer le critère d'intégrité**. Par exemple, le taux de perte ou d'erreur et le taux d'altération de messages peuvent nous renseigner sur l'intégrité des données.

❖ *La Disponibilité*

La disponibilité désigne la garantie de la continuité de service, des objectifs de performances (temps de réponse) et le respect des dates et heures limites du traitement. La question qui se pose par rapport à la disponibilité : Le service fonctionne-t-il correctement ? Est-il accessible au moment où il est sollicité que par des entités autorisées ?

On peut constater que **les métriques de disponibilité et du délai** dans l'évaluation du comportement (QoS) de service peuvent nous servir pour **mesurer le critère disponibilité** dans l'audit de sécurité**.** Par exemple, le taux d'accessibilité peut nous renseigner sur la disponibilité de service et s'il s'agit d'une attaque déni de service (DoS) ou juste d'une faute accidentelle (panne, incident, etc.).

❖ *La non-répudiation*

La non-répudiation est la protection contre la négation d'une action accomplie. Le fournisseur de services ou l'utilisateur peuvent-ils prétendre qu'ils n'ont pas reçu ou effectué une transaction?

Il faut alors garantir la possibilité de reconstituer le déroulement d'un traitement pour des fins de contrôle et de preuve.

La non-répudiation peut être réalisée par un ensemble de mesures garantissant l'enregistrement fiable d'informations pertinentes par rapport à une entité ou un événement.

On définit alors **les métriques** suivantes relatives à la **non-répudiation :**

- Probabilité de trouver la signature ou la clé de chiffrement
- Probabilité de résistance à la répudiation

| *Critères Sécurité* | *Métriques ?* |
|---|---|
| ***Confidentialité*** | - Taux de divulgation d'information<br>- Probabilité de résistance aux attaques (analyse et écoute clandestine, divulgation d'information)<br>- Niveau de chiffrement |
| ***Intégrité*** | - Taux de perte<br>- Taux d'erreur<br>- Taux d'altération de messages |
| ***Disponibilité*** | - Taux d'accessibilité<br>- Temps de réponse |
| ***Non-répudiation*** | - Probabilité de trouver la signature ou la clé de chiffrement<br>- Probabilité de résistance à la répudiation |

***Tableau 4 :*** *Métriques de sécurité.*

*Quels sont les paramètres à mesurer effectivement en se basant sur les métriques déterminées relatives à chaque objectif de sécurité ?*

On constate que les valeurs mesurables prises par le service monitoring pour la fiabilité peuvent être interprétés pour l'intégrité. Pareil, pour la disponibilité et le délai, leurs valeurs mesurables peuvent être interprétées pour la disponibilité d'un point de vue de sécurité, Par exemple, le temps de réponse (délai) qui dépasse un certain seuil peut être dû à une attaque de déni de service.

Cependant, les métriques que l'on a déterminées pour la confidentialité et la non-répudiation doivent être traduites en valeurs mesurables tangibles. Par exemple, pour la confidentialité, comment peut-on-mesurer réellement le taux de divulgation d'information qui une métrique relative à ce critère ?

Dans le cadre de sécurité, garantir généralement les exigences de sécurité (disponibilité, confidentialité, intégrité) commence par vérifier le bon fonctionnement des services de sécurité mis en place.

Donc, les valeurs mesurables relatives au comportement et au bon fonctionnement des services déployés assurant la confidentialité peuvent être utilisées pour mesurer par exemple le taux de divulgation.

Le tableau ci-dessous présente les services relatifs aux deux critères de sécurité : la confidentialité et la non- répudiation, en précisant ceux que nous avons traité de point de vue QoS dans notre thèse.

| Objectifs de sécurité | Les Services | Evaluation du comportement (QoS) |
|---|---|---|
| **Confidentialité** | Identification | ✓ |
| | Authentification | ✓ |
| | Firewall Applicatif | A définir |
| | Autorisation | ✓ |
| | Chiffrement (Protection du contenu, Privacy) | A définir |
| **Non-répudiation** | Signature | A définir |
| | Certificat | A définir |

***Tableau 5 :*** *Les services relatifs à la confidentialité et à la non-répudiation.*

En conclusion, mesurer la confidentialité et la non-répudiation, revient à évaluer la conformité de ces services associés à chaque objectif.

- **Contrôle de la sécurité**

Le composant de l'audit de sécurité se base sur les paramètres mesurables collectés par le service de monitoring pour effectuer une analyse et ***un contrôle*** de la sécurité des services afin d'atteindre les objectifs définis au préalable.

La tâche principale du composant d'audit de sécurité comme le montre (la Figure 19 : A) est le contrôle de la sécurité afin de donner des réponses pertinentes soit par rapport à une situation donnée (faille de sécurité, acte malveillant, mobilité…) ou soit par rapport à un objectif fixé (évaluation d'une distance par rapport à un objectif). Le résultat va engendrer des prises de décisions et des actions permettant de résoudre les problèmes. Ces actions peuvent être préventives ou correctives. Les actions préventives proposent généralement des améliorations organisationnelles à travers les analyses et les évaluations de risques ou en suivant des bonnes pratiques en se basant sur les normes de sécurité les plus répandues (par exemple ISO27001).

Les actions correctives envisagent de mettre en place des nouveaux dispositifs et de créer des services de sécurité auto-gérables permettant de déceler les failles de sécurité ainsi renforcer le niveau de protection contre les menaces potentielles liés au contexte.

L'avantage de cette solution (composant d'audit de sécurité) consiste à contrôler et analyser la sécurité en se basant non seulement sur les paramètres mesurables relatives des composants service en temps réel mais aussi en s'appuyant sur les notifications de non-conformité (out contrat) susceptibles d'être dues à une attaque ou une faille de sécurité. Ceci veut dire que notre proposition d'audit se présente comme étant un composant de service d'audit de sécurité basé sur la QoS (Figure 19: B). En effet, il contrôle et analyse, tout au long des différentes phases de la session de l'utilisateur, la sécurité de bout en bout de ses transactions et de ses services selon ses attentes.

Le service d'audit de sécurité se sert des mesures effectuées pour l'évaluation de comportement (QoS) des services en particulier ceux de sécurité pour les interpréter d'un point de vue de sécurité. Il réagit dynamiquement et de manière autonome dans le cas d'un changement au cours de la session de l'utilisateur.

Prenons l'exemple suivant : Ayant l'intégrité comme objectif fixé de sécurité, l'Audit de sécurité reçoit la notification «Out Contrat», générée par l'évaluation de comportement de service. Si c'est le taux d'erreur des messages qui dépasse le seuil et qui a donc provoqué le « Out Contrat », l'audit de sécurité déduit une attaque potentielle (par exemple, altération de messages) et prend l'action adéquate.

Dans le cas de la disponibilité, si l'Audit de sécurité reçoit, par exemple, une autre notification « Out Contrat » induit par un dépassement des valeurs seuils du taux de rejet, il doit vérifier s'il s'agit d'une attaque de déni de service.

Ce composant d'audit de sécurité est un composant de service intégré dans chaque plate-forme de service pour détecter les failles de sécurité au bon moment. Il comporte deux entrées : les notifications de non-conformité générées par l'évaluation de comportement (l'agent QoS) et puis les paramètres mesurables définis selon les objectifs de sécurité. Le résultat va engendrer des prises de décisions et des actions qui peuvent être soit correctives (création des services-auto gérables) ou préventives (améliorations organisationnelles).

En résumé, la figure ci-dessous montre comment notre audit de sécurité est basé sur l'évaluation du comportement (QoS) d'un service et comment certaines métriques associées au critère de QoS sont utilisées pour l'évaluation de la sécurité.

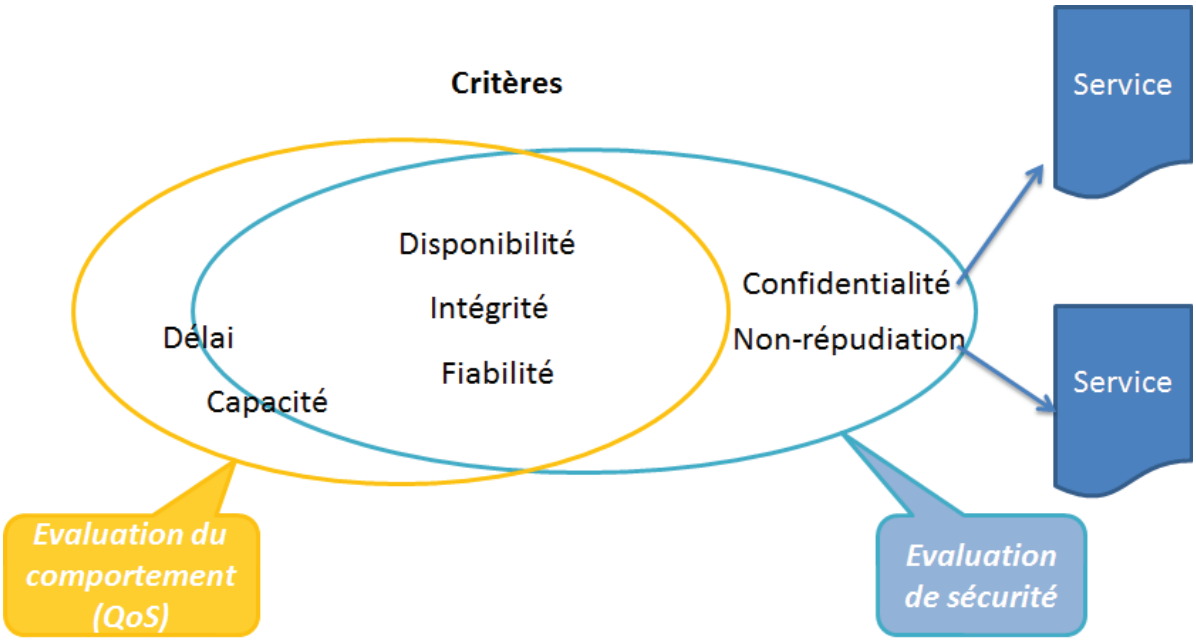



***Figure 21 :*** *Audit de sécurité basé sur la QoS.*

## 4.5. Conclusion

Dans cette partie, nous avons proposé une approche architecturale de la sécurité basée sur le concept «security as a service ». Elle surmonte les défis des architectures web services qui favorise la sécurité en silo et les limitations des architectures SOA au niveau de la conception d'un composant de service, en particulier celui de sécurité, qui est réutilisable et interopérable avec des liens à couplage lâche mais qui ne permet pas d'avoir une gestion efficace et dynamique des besoins des utilisateurs, de leurs préférences, de leurs mobilités et de la sécurité de leur session. Cette architecture peut être une réponse au besoin de sécurité dans le Cloud.

Dans le but d'avoir une meilleure gestion et contrôle de sécurité de bout en bout, notre architecture de sécurité s'appuie sur un ensemble de services exposés dans le « Securityware ». Ces services suivent le modèle d'un service UBIS, ils sont mutualisables, autonomes, stateless et auto-gérables.

La sécurité fournie à travers une composition de service nous a permis d'assurer l'unicité et l'accès sécurisé de la session centrée utilisateur. Cette composition est répartie sur des niveaux de visibilité. Nous nous sommes intéressés à la sécurité de la composition au niveau terminal (VPDN) et au niveau service (VPSN).

Finalement, pour assurer un contrôle efficace et automatisé de la sécurité des services, nous avons proposé un composant d'Audit de sécurité basé sur la QoS.

*Partie 2*

# Aspect Protocolaire: « Token-based SIP+ »

# 5

# Introduction

Dans notre environnement UBIS ouvert et hétérogène où la mobilité est omniprésente avec ses différentes déclinaisons (utilisateur, terminal, réseau, service), la sécurité s'avère un point crucial durant les différentes phases (établissement, modification, etc.) d'une session *User-Centric*. Cette session se veut mobile, unique, sans couture, sans coupure et sécurisée pour répondre aux besoins de l'utilisateur tout en tenant en compte des changements liés à la mobilité et aux défaillances de sécurité (failles, attaques).

En effet, cette session est basée sur une composition de services convergents (web, IT, Telco), tout en s'appuyant sur une agrégation de plusieurs services appartenant à des environnements et des plates-formes hétérogènes. Ce qui rend cette session vulnérable et exposée à des risques significatifs en termes de sécurité liée au contexte. Pour cette raison, nous devons recenser les diverses attaques possibles durant les différentes phases de l'échange.

Donc, la maitrise de la sécurité de ***bout en bout*** de cette session d'une manière transparente passe essentiellement par la capacité d'assurer la sécurité non seulement au niveau des accès aux composants de services quel que soit leur emplacement, qu'ils soient hébergés sur différentes plates-formes ou offerts par différents fournisseurs de services mais aussi au niveau des échanges des messages et des données.

Pour atteindre cet objectif, nous devons lever les verrous évoqués précédemment, à savoir : Assurer l'unicité de la session, garantir des accès sécurisés avec un maintien de la continuité de service sans avoir besoin de se ré-authentifier à chaque demande de service, traiter le manque de confiance vis-à-vis les fournisseurs de services vu que les services sont fournies par différentes plates-formes distantes auxquelles on ne doit pas se fier et enfin sécuriser le protocole utilisé pour accomplir la composition de services et acheminer les données entre les différents composants.

Face à ces verrous, nous commençons par identifier les besoins (§6), et analyser les réponses de l'existant (§7). Puis, pour positionner notre proposition (§8), nous décrivons le contexte de notre étude s'agissant de la session UBIS qui englobe les besoins « user-centric », de mobilité et de sécurité. Le protocole proposé SIP+ est alors décrit. Il répond aux besoins de mobilité et permet une composition de service dynamique et sécurisée. C'est alors que nous proposons et décrivons le jeton que nous associons à SIP+ pour permettre une authentification unique. Nous définissons les mécanismes nécessaires à la sécurisation de ce protocole afin d'assurer la sécurité des échanges.

# 6

# Les besoins

UBIS veut offrir un service d'authentification unique permettant à l'utilisateur d'accéder à tous les services auxquels il est autorisé lors de sa session. Cela signifie que, lors de la demande d'accès aux composants de service, l'authentification s'effectue auprès d'un fournisseur de service de sécurité qui propage l'identité de l'utilisateur, aux différentes plates-formes de service hébergeant les services demandés.

Par conséquent, il est nécessaire de faire face à la complexité de la gestion des accès aux composants dans un environnement hétérogène (multi-plates-formes, multifournisseurs et multiservices). UBIS veut assurer un certain niveau de transparence par rapport aux technologies utilisées et masquer la complexité des architectures vis-à-vis à l'utilisateur.

C'est pourquoi, il faut réduire les nombres de secrets à gérer par utilisateur afin d'optimiser le temps d'accès.

UBIS prévoit également de trouver une réponse à la problématique des changements de mots de passe afin d'offrir une simplicité accrue et un confort à l'utilisateur. En effet, les authentifications successives pour accéder aux différents services doivent laisser place à une seule authentification transparente limitant ainsi les risques d'erreurs de saisies multiples. De la même façon, la déconnexion peut être globale et instantanée.

Cependant, l'authentification unique ne doit pas dégrader les niveaux de sécurité mis en place pour chaque service tout en tenant en compte de l'évolution des usages comme la mobilité qui amène des nouvelles contraintes de sécurité.

Avec moins de mots de passe circulant sur le réseau et moins de procédures d'authentification, UBIS vise à avoir une plus grande facilité pour maîtriser et appliquer une politique de gestion des mots de passe. Ceci permet de limiter les risques engendrés par le transport de ces derniers sur le réseau, afin de mieux contrôler et renforcer la sécurité.

Un autre objectif important à atteindre est d'assurer une meilleure intégration des composants de service dans le système à travers l'utilisation d'un service commun d'authentification axé autour d'un référentiel unique et cohérent, en fournissant les mêmes garanties de sécurité et en définissant une politique cohérente de gestion des comptes pour les différents services.

UBIS veut non seulement permettre à l'utilisateur d'avoir la possibilité de s'authentifier une seule fois pour l'ensemble des services sollicités, mais cela soit possible, quel que soit la localisation de l'utilisateur, qu'il puisse faire le choix de ses composants chez n'importe quel fournisseur de service. Autrement dit, la délivrance de service doit être assurée par

plusieurs fournisseurs de services. Cela implique un partage d'informations qui soit soumis à des contraintes de sécurité.

Pour cette raison, il est nécessaire d'établir une relation de confiance d'une part entre les différents fournisseurs de services et d'autre part entre les différents fournisseurs de sécurité. Les entités en collaboration (fournisseurs de service, fournisseurs de sécurité) doivent être en mesure d'assurer la propagation des identités (jeton) au-delà des frontières de l'organisation à laquelle l'utilisateur appartient. Ceci permettra l'accès contrôlé et sécurisé de l'utilisateur aux services offerts par son fournisseur de services et de ses partenaires. Ce qui définit un deuxième axe de besoin qui est « la fédération ».

En effet, la fédération a pour objectif principal de faciliter les échanges entre les partenaires sans avoir à dupliquer les référentiels (annuaires) des identités en faisant la délégation d'authentification des utilisateurs sur la base de confiance.

C'est un concept qui permet le partage et la gestion des données d'identité ainsi que l'établissement d'une connexion unique à travers multiples domaines de sécurité et différentes organisations. Elle permet à une entité d'offrir une variété de services à des partenaires de confiance.

Fédérer des fournisseurs de services ou de sécurité (d'identité) commence par établir un cercle de confiance regroupant tous les acteurs impliqués qui s'engagent contractuellement à échanger des informations d'authentification. Chaque cercle de confiance doit inclure au moins un fournisseur de sécurité (d'identité) qui maintient et gère les données d'identité, et fournit des services d'authentification.
Cette confiance est généralement définie par des accords d'entreprises ou de contrats qui décrivent les techniques, les responsabilités opérationnelles et juridiques de chaque partie. Une relation de confiance permet à une organisation de faire confiance à l'authentification des utilisateurs et des décisions d'autorisation d'une autre organisation.

Pour mieux illustrer le principe de fédération, nous citons l'exemple illustré dans la Figure 22 suivante :

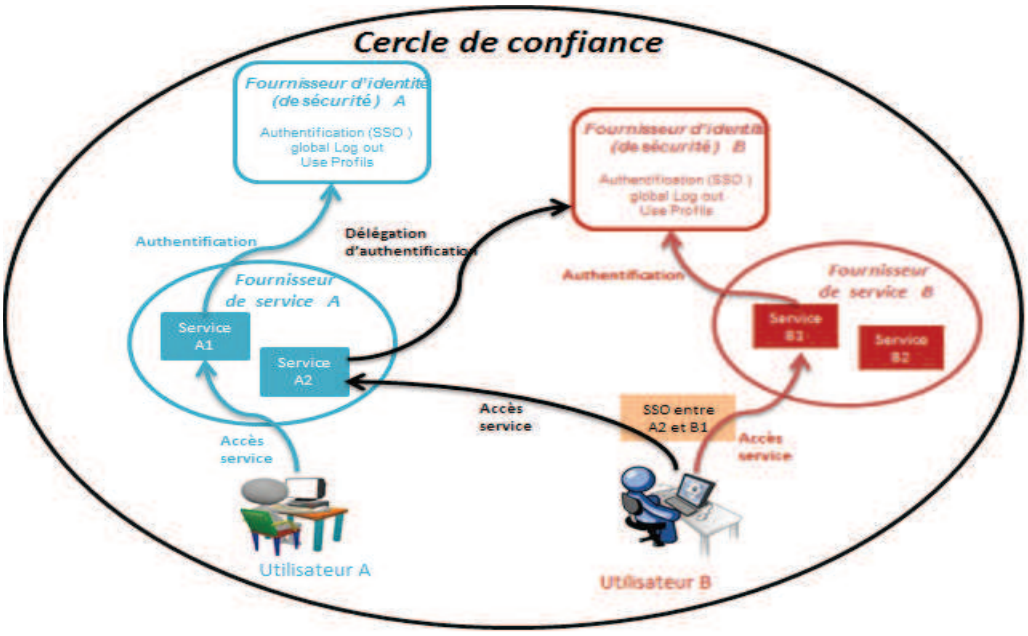



***Figure 22:*** *Principe de la fédération.*

Lorsqu'un utilisateur A accède à un service A1 offert par son fournisseur de service A, l'authentification se fait auprès du fournisseur d'identité A interne. Lorsqu' un utilisateur B externe à l'organisation A veut accéder à un service A2 externe à son fournisseur de service, l'authentification sera faite auprès de son fournisseur d'identité B puisqu' il existe une relation de confiance entre les deux organisations. Lorsqu'un utilisateur B accède à un service A2 externe à son entreprise et un service B1 interne à son entreprise, les deux services s'appuient sur le fournisseur d'identité de son entreprise. L'utilisateur B pourra bénéficier ainsi de Single Sign On entre les deux services émanant de deux organisations différentes.

Nos besoins identifiés ci-dessus s'adressent essentiellement aux aspects de sécurité relatives à l'accès aux services à savoir l'authentification unique et la fédération.

D'un point de vue protocolaire, UBIS a fait le choix du protocole SIP (avec une extension) pour l'établissement et le contrôle de la session, des protocoles HTTP et SOAP respectivement pour le transport des données et pour les web services. Il nous faudra étudier les attaques pour avoir les besoins en contre-mesures. Dans cette thèse, on n'abordera que celles concernant SIP.

# 7

# Analyse de l'existant

## 7.1. Introduction

Après l'identification des nouveaux besoins imposés par notre contexte, nous allons consigner dans cette partie les réponses de l'existant concernant l'authentification unique avec le mécanisme du Single Sign On (§7.2) et la fédération (§7.3). Dans cette thèse nos contributions s'attachent essentiellement à deux axes principaux: la dimension architecturale évoquée dans le chapitre précèdent et la dimension protocolaire de la sécurité que nous allons traiter dans ce chapitre.

Pour cette raison, une étude approfondie, qui s'intéresse particulièrement aux aspects de sécurité, a été faite sur les protocoles SIP (§7.4), HTTP (§7.5) et SOAP (§7.6) qui sont utilisés dans notre contexte durant les différentes phases de la session (usage, contrôle, gestion). Nous avons choisi ces protocoles du fait que les services qui vont participer à la composition de la session de l'utilisateur sont convergents de nature différente (Web, Telco, IT).

## 7.2. Single Sign On

Le *Single Sign On* est une technique qui consiste à soumettre l'utilisateur à un service d'authentification unique par rapport aux différents services accessibles, applications ou fonctions protégées du système. Il repose sur un moteur SSO représentant l'élément central qui assure l'authentification de l'utilisateur et la persistance de sa session en propageant son identité vers les applications. En général, il existe plusieurs architectures des systèmes SSO. Les plus répandues sont celles avec agent et reverse proxy.

### 7.2.1 SSO avec agent

Cette architecture repose sur un agent d'authentification permettant de protéger l'accès aux applications. Il intercepte les demandes de l'utilisateur et les redirige vers le moteur SSO pour vérifier s'il est authentifié. Si l'utilisateur est déjà authentifié c'est-à-dire qu'un jeton a été généré, l'agent demande au moteur SSO de valider ce jeton. En retour, il reçoit les accréditations de l'utilisateur et les transmet à l'application (Figure 23).

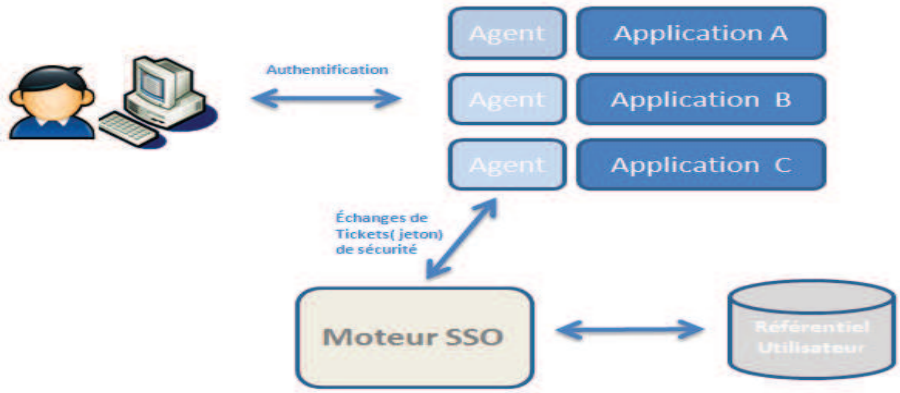



***Figure 23 :*** *SSO avec Agent.*

### 7.2.2 SSO reverse proxy

Dans cette architecture (Figure 24), le moteur SSO joue le rôle d'une passerelle de sécurité qui permet de protéger les applications afin qu'elles ne soient pas accédées directement par les utilisateurs en cachant leurs emplacements réels grâce à des méthodes d'URL rewriting. Le SSO reverse proxy s'appuie sur une mise en cache des identifiants/ mots de passe secondaires dans une base de correspondance. En fait, à chaque identifiant primaire, il associe une série d'identifiants/mots de passe secondaires. Après avoir mené à bien l'authentification primaire, l'utilisateur choisit l'application à laquelle il veut accéder. Par la suite, les données d'authentification secondaire sont envoyées à l'application cible en HTTPS. Enfin, l'utilisateur peut accéder à l'application à laquelle il est autorisé.

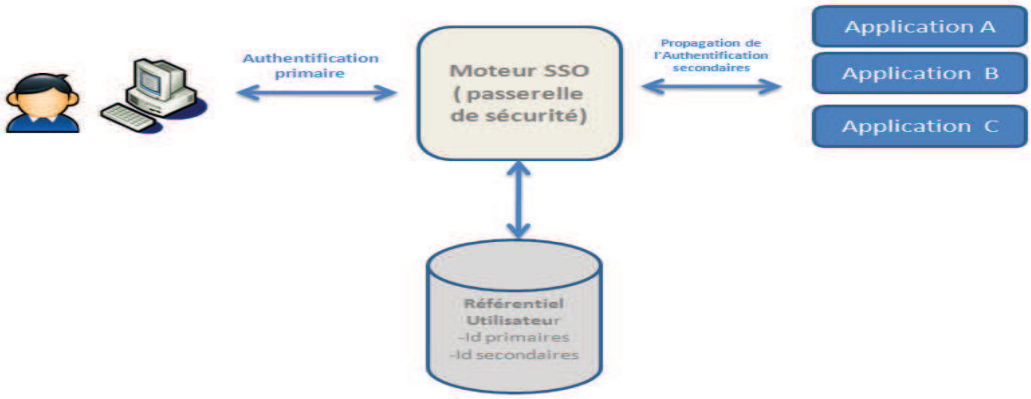



***Figure 24 :*** *SSO Reverse Proxy.*

## 7.3. La fédération

Les solutions de fédération sont nombreuses, nous citons notamment : Liberty Alliance, Shibboleth, WS-Federation, OpenID, etc. Dans ce qui suit nous allons juste détailler le système OpenID et Liberty Alliance à titre d'exemple.

### 7.3.1 Liberty Alliance

Les spécifications de Liberty Alliance [39] définissent trois types d'acteurs : l'utilisateur, le fournisseur d'identité et le fournisseur de services. Pour que ce système puisse fonctionner,

il faudra définir un cercle de confiance entre les fournisseurs d'identité et de services qui se sont mis d'accord pour fédérer l'identité de leurs utilisateurs. L'authentification unique dans Liberty Alliance donne la possibilité à l'utilisateur d'accéder, après s'être identifié et authentifié à l'aide d'un compte unique, à des services proposés par différents fournisseurs appartenant à un même "cercle de confiance".

Liberty Alliance est une approche intéressante. En effet, la séparation entre le fournisseur d'identité et le fournisseur de services procure à ce dernier plus d'autonomie pour administrer ses ressources. Néanmoins, l'authentification avec Liberty Alliance est une approche fédérée qui présente un problème dû au fait qu'un nouveau compte doit être mis en place afin de regrouper tous les comptes existants.

### 7.3.2 OpenID

OpenID [40] est un mécanisme fédératif d'authentification unique et de partage d'attributs. Il permet à un utilisateur de s'authentifier auprès de plusieurs sites sans avoir à retenir un identifiant différent pour chacun d'entre eux mais en utilisant un identifiant unique OpenID sous forme d'URI (ex : "openidfrance.fr/votrenom").

Il se base sur des liens de confiance préalablement établis entre les fournisseurs de services (sites web, forum, email utilisant OpenID) et le fournisseur d'identité (OpenID provider). Ce dernier est responsable de la protection et de surveillance de l'identité des utilisateurs. Il permet à chaque membre de visualiser la liste des fournisseurs de services auxquels il est abonné, ainsi que les informations qui ont été partagées avec ces derniers. Les données personnelles ne sont jamais diffusées sans l'accord du propriétaire et sont stockées sur un serveur dédié et sécurisé.

L'adoption d'OpenID impose à l'usager d'abandonner tous ses comptes existants en créant un nouveau compte géré par son nouveau fournisseur d'identité. En outre, la sécurité d'OpenID dépend largement de la chaine de confiance des différents partenaires (qui ne se connaissent pas nécessairement depuis longtemps).

## 7.4. Le protocole SIP

### 7.4.1 SIP : les fonctionnalités

SIP (Session Initiation Protocol) [41] est un protocole de signalisation défini par l'IETF (Internet Engineering Task Force) permettant l'établissement, la libération et la modification d'une session multimédia (RFC 3261). Il assure la négociation des types médias entre les différents participants en encapsulant des messages SDP (Session Description Protocol) [42]. SIP a été étendu afin de supporter de nombreux services tels que la présence, la messagerie instantanée (similaire au service SMS dans les réseaux mobiles), le transfert d'appel, la conférence, les services complémentaires de téléphonie, etc. Il a été retenu par le 3GPP pour l'architecture IMS (IP Multimedia Subsystem) [43][44] comme le protocole responsable du contrôle de session et du contrôle de service.
Le protocole SIP se charge uniquement de la signalisation. Une fois la session établie, les participants de la session s'échangent directement leur trafic audio/vidéo à travers le protocole RTP (Real-Time Transport Protocol). Par ailleurs, SIP ne fait pas de réservation de ressource, alors il ne peut pas assurer la QoS. Il s'agit d'un protocole de contrôle d'appel et non de contrôle du média.

SIP repose sur un modèle transactionnel client/serveur comme HTTP (Hyper Text Transport Protocol). Il hérite certaines fonctionnalités du protocole HTTP pour naviguer sur le Web et du protocole SMTP (Simple Mail Transport Protocol) pour transmettre des courriers (E-mail). Les messages échangés sont donc soit des requêtes soit des réponses.
En effet, le RFC 3261 définit initialement six requêtes ou méthodes de base de SIP :

- **REGISTER** qui permet l'enregistrement d'un client auprès d'un serveur.
- **INVITE** qui a pour but l'initiation de session de toute nature: session audio, session vidéo, session chat, session fax, session IPTV, etc.
- **ACK** qui sert à confirmer la réponse finale à une méthode INVITE.
- **OPTIONS** qui permet d'obtenir les capacités de l'entité interrogée.
- **BYE** qui termine une session établie.
- **CANCEL** qui permet d'annuler une session qui n'a pas encore été établie.

D'autres méthodes existent et ont été aussi définies par l'IETF issues d'autres RFC pour étendre les méthodes du protocole SIP.

- **SUBSCRIBE** demande de notification d'évènement de session (RFC 3265).
- **NOTIFY** notifie un événement pour lequel une souscription a été effectuée (RFC 3265).
- **PUBLISH** publie un état (RFC 3515).
- **INFO** permet l'échange des informations de signalisation durant la session (RFC 2976).
- **REFER** permet de transférer une session (RFC 3515).
- **RE-INVITE** permet de modifier une session.
- **Message** transfère les messages instantanés (RFC 3428).
- **UPDATE** permet à un terminal SIP de mettre à jour les paramètres d'une session multimédia (e.g. flux média et leurs codecs) (RFC 3311).
- **PRACK** acquitte les réponses provisoires (RFC 3262).

Les requêtes SIP sont acquittées par des réponses qui indiquent soit une information de progression d'appel, soit une information d'état final.
Concernant la structure des messages SIP (Figure 25), ils sont textuels et comportent trois parties à savoir la première ligne, l'en-tête et le corps du message séparés notamment par une ligne vide. La première ligne de la méthode SIP, appelée « *request line »* liste la méthode, le « *request URI* » qui indique l'entité destinatrice de la requête, et le numéro de version du protocole SIP utilisé « *Protocol version ».*

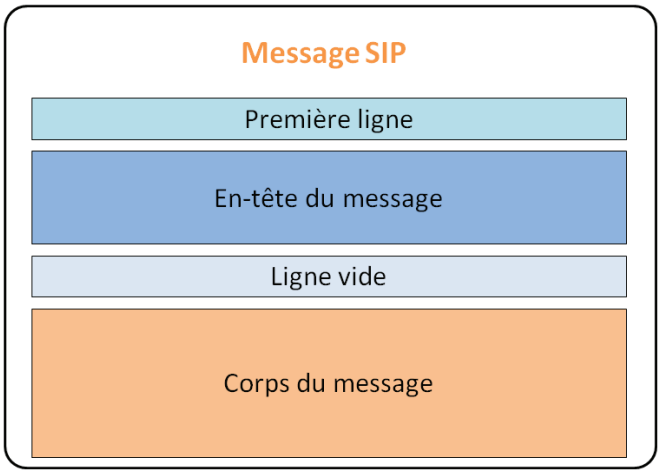



***Figure 25 :*** *Structure d'un message SIP.*

L'en-tête contient les informations permettant l'acheminement du message à savoir la référence de l'émetteur, le destinataire, la référence de la transaction et de la session, et les éléments de sécurité. Ainsi, l'en-tête permet l'établissement d'une session en termes de localisation, de nommage et d'adressage, mais c'est le corps du message qui décrit le flux multimédia mis en jeu par la session. Le corps du message est optionnel; certaines requêtes SIP n'en disposent pas. Le protocole SIP peut transporter tout type de corps, par exemple, Application /SDP et Text/ XML. Le type du corps de message doit être mentionné par le champ *content-type* de l'en-tête SIP et sa longueur par le champ *content-length*. La longueur de l'ensemble des champs ne doit pas dépasser le MTU (maximum transmission unit). Si la MTU est inconnue, elle est de 1500 octets par défaut. Cette taille permet l'encapsulation des datagrammes UDP ou segments TCP dans des paquets IP sans fragmentation. Tout message SIP doit contenir les champs obligatoires *To, From, Max-Forwards, Via, Call-ID, C-Seq et Contact*. Les autres headers sont optionnels. Le Tableau 6 ci-dessous présente les en-têtes SIP les plus fréquentes.

| Champs d'en-tête | Description |
|---|---|
| **Authorization** | Information d'authentification pour l'usage d'une ressource par un UA (un équipement). |
| **Call-ID (*)** | Identifiant unique et global d'une session. |
| **Contact** | Généralement, c'est l'URL de l'usager. |
| **Content-Length (*)** | Longueur du message en octets. |
| **Content-Type (*)** | Type du corps du message (ex : une description SDP). |
| **CSeq (*)** | Contient un numéro de séquence qui identifie une requête durant une session. |
| **Encryption** | Précise que le contenu est chiffré. |

| | |
|---|---|
| **From (*)** | Initiateur de la requête. |
| **Max-Forwards (*)** | Utilisé pour toute requête afin de limiter le nombre de serveurs SIP qui peuvent être traversés jusqu'à destination. Chaque serveur recevant une requête devra décrémenter la valeur du champ Max-Forwards. |
| **Proxy-Authenticate** | Utilisé pour formuler une demande d'authentification. |
| **Proxy-Authorization** | Permet au client de s'identifier auprès d'un proxy qui lui demande une authentification. Ce champ contient les éléments relatifs à l'identification de l'utilisateur sur le proxy et/ou le domaine des ressources demandées. |
| **Proxy Require** | Précise un mécanisme qui doit être fournie par le proxy. |
| **Timestamp** | Date d'émission de la requête. |
| **To (*)** | Indique l'adresse de destination. |
| **Unsupported** | Liste les mécanismes non supportés par le serveur. |
| **User-Agent** | Information sur l'UA qui a généré le message. |
| **Via (*)** | Indique le chemin emprunté par la requête jusqu'à l'instant présent. |
| **WWW-Authenticate** | Contient une demande d'authentification. |
| **Route** | utilisé pour forcer le routage d'une requête au travers d'une liste définie de proxy. |

***Tableau 6 :*** *Les champs d'en-tête du message SIP.*

### 7.4.2 SIP : Les attaques

Notre projet UBIS a fait le choix du protocole SIP, mais il n'est pas à l'abri des attaques [45][46]. En effet, SIP est un protocole de signalisation pour les communications et les services basés sur IP, donc il hérite des vulnérabilités et des menaces classiques liées à IP à savoir le déni de service, les attaques par rejeu, le « spoofing », le « sniffing », le « man in the middle », etc. De plus, il existe certaines attaques qui sont spécifiques à la nature du protocole SIP lui-même comme l'émission de faux messages INVITE, ou l'inondation du serveur avec des messages INVITE (Invite Flooding) et les faux messages CANCEL ou BYE. D'autres attaques sont liées à l'implémentation et sont la conséquence de la complexité des applications multimédia et VOIP.

Les attaques rencontrées dans le domaine de la sécurité du protocole SIP sont très nombreuses, plus particulièrement celles qui ont un impact sur l'authentification.

Ceci nous amène à classer les attaques potentielles (Tableau 7) selon les catégories suivantes : le déni de service, le vol de service, l'usurpation d'identité, l'interception et la modification des messages.

| Catégorie (risque) | Description | Attaque (méthode) |
|---|---|---|
| **Interruption de service ou déni de service** | Attaque entraînant l'indisponibilité d'un service/système pour les utilisateurs légitimes. | - Session tear-down using Bye message<br>- Session tear-down using Cancel message<br>- Invite Flooding |
| **Vol de service** | Attaque permettant d'utiliser un service sans avoir à rémunérer son fournisseur. | - Theft of service en utilisant des accréditations d'un client légitime<br>- Billing fooling |
| **Usurpation d'identité** | Attaque basée sur la manipulation d'identité. | - Request spoofing<br>- Masquage d'appels |
| **Interception et modification des messages** | Interception et falsification des messages pour détourner une session. | - Session Hijacking<br>- Re-INVITE / Session replay<br>- Message Spoofing<br>- Replay attack |

***Tableau 7 :*** *Classification des attaques du protocole SIP.*

Dans notre étude, nous nous intéressons particulièrement à l'authentification, c'est pourquoi, nous allons recenser les attaques potentielles qui ont un impact sur ce service.

- ***Session Tear-down using Bye Message*** (Figure 26): L'attaque par la méthode BYE consiste à générer un BYE pour interrompre une conversation. Pour réaliser cette attaque, le pirate écoute le trafic et prend les informations nécessaires (comme par exemple le Call-Id, le From ou encore le To) pour générer un BYE frauduleux correspondant à la session qui sera injecté sur le réseau. Etant donné que le message BYE ne nécessite pas une authentification, celui qui reçoit l'information l'exécute.

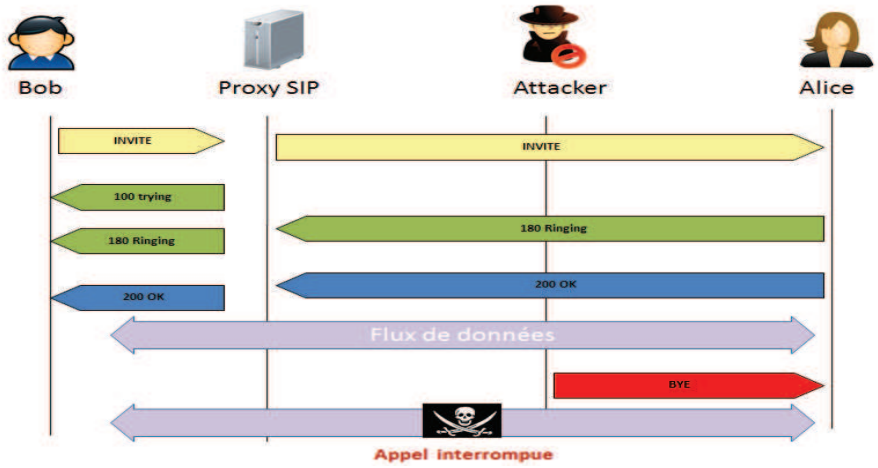



***Figure 26 :*** *L'attaque "Session Tear-down using Bye Message".*

- ***Session Tear-down using Cancel Message*** (Figure 27): L'attaque par la méthode CANCEL consiste à générer un message CANCEL façonné pendant l'établissement d'une session. Cette attaque est semblable à celle du BYE mais cette fois elle se déroule avant l'établissement de la session. Du côté des plates-formes de service ou de l'utilisateur pensent que la requête Invite a été annulée. Cette attaque est possible car le CANCEL n'est pas authentifié.

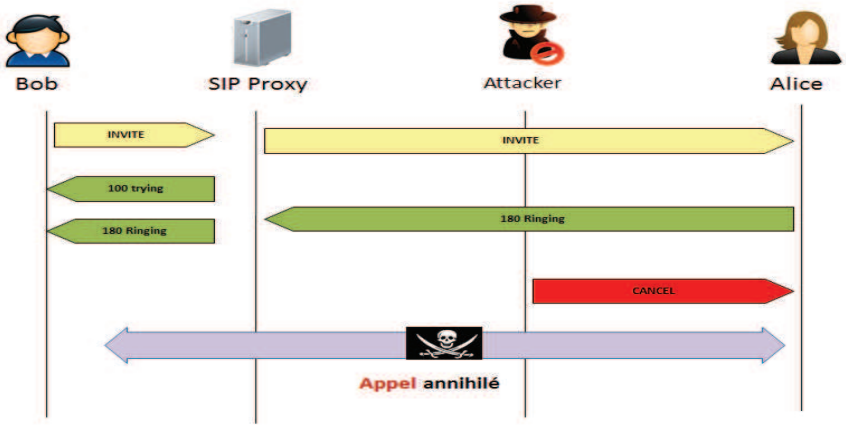



***Figure 27 :*** *L'attaque "Session Tear-down using CANCEL Message".*

- ***Session Replay*** (Figure 28): Dans cette attaque, une transmission de données valides est répétée d'une manière frauduleuse. Elle a pour but, par exemple, d'enregistrer un appel ou de modifier les paramètres de configuration d'un appel. L'attaquant doit, tout d'abord, écouter le réseau et récupérer un message INVITE transmis entre un appelant et un appelé. Il peut ensuite introduire un message INVITE façonné dans la conversation en cours, de telle sorte que les paramètres soient pris en compte (par exemple les paramètres de routage) et qu'un troisième parti soit introduit dans la conversation, par exemple pour faciliter l'enregistrement de la conversation.

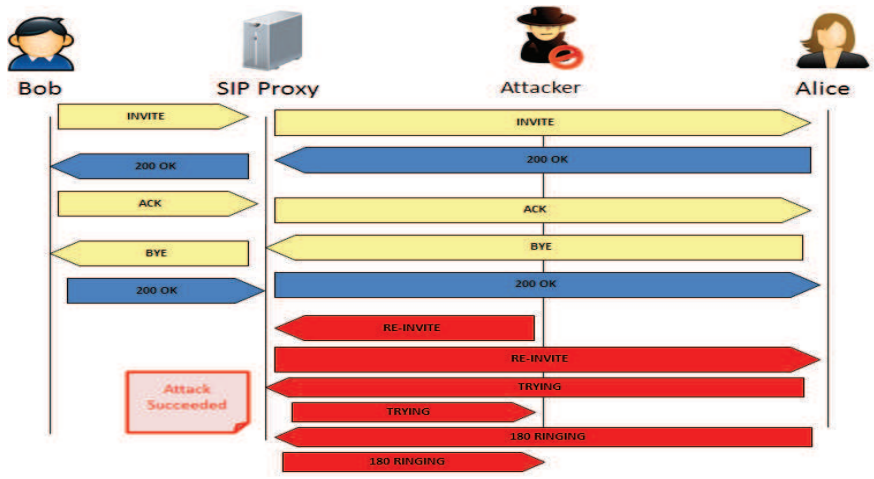



***Figure 28 :*** *L'attaque « Session Replay ».*

### 7.4.3 SIP : les mécanismes de sécurité

Pour faire face à ces menaces, Le RFC de SIP prévoit certaines solutions [46][47][48] qui ont été proposées pour assurer la confidentialité, l'intégrité et particulièrement l'authentification à travers la signalisation selon les différentes couches à savoir la couche applicative, la couche transport et la couche réseau comme le montre la Figure 29 ci–dessous qui présente les mécanismes de sécurité de SIP selon les couches.

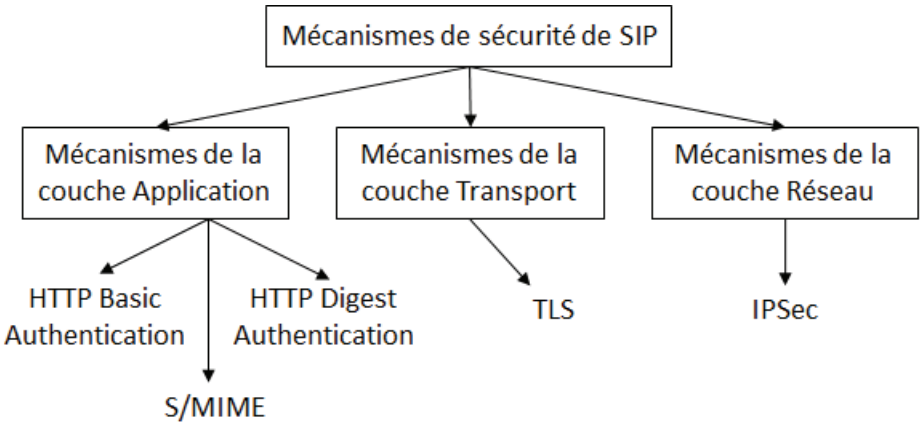



***Figure 29 :*** *Les différents mécanismes de sécurité SIP.*

SIP fait intervenir la sécurité de deux manières : imbriquée dans les messages ou assurée par les couches sous-jacentes. En effet, la structure des messages SIP est similaire au modèle requête/réponse du HTTP, donc au niveau applicatif les mécanismes de sécurité disponibles pour HTTP, comme « HTTP Basic Authentication » et « HTTP Digest Authentication », sont également applicables aux sessions SIP. De plus, le recours à S/MIME permet d'assurer l'intégrité et la confidentialité des messages SIP. Ces deux solutions S/MIME et HTTP Digest [RFC2617] sont intégrées dans les messages SIP, ce qui permet d'envisager une sécurité de bout en bout.

Au niveau de la couche transport, la combinaison de TLS (Transport Layer Security) et SRTP (Secure Real-time Transport Protocol) est largement utilisée dans la variété des services SIP comme la VoIP. TLS fournit un tunnel de transport sécurisé et SRTP authentifie et crypte les paquets RTP afin de sécuriser la transmission de données. Au niveau de la couche réseau, IP security (IPsec) peut être utilisé pour protéger les communications IP. Enfin, des pare-feux peuvent être utilisés pour sécuriser les réseaux locaux des attaques externes.

Nous nous intéressons essentiellement aux mécanismes de sécurité (authentification) au niveau de la couche application. Alors, détaillons les mécanismes d'authentification HTTP Digest et de S/MIME :

- **Authentification HTTP Digest**

L'authentification HTTP Digest [RFC2617] [49] se base sur un échange de type challenge/réponse pour vérifier que les deux parties de la communication connaissent un secret commun (mot de passe). Contrairement à l'authentification HTTP Basic, cette vérification peut se faire sans envoyer le mot de passe en clair sur le réseau, qui présente le plus grand défaut de la méthode Basic. Elle s'effectue alors en transmettant le calcul résultant de la fonction de hachage (MD5) appliquée à la clé (mot de passe) et au challenge.

Le principe de cette authentification est simple. D'abord, le client envoie sa demande d'enregistrement (requête REGISTER) au serveur. Celui-ci lui répond (réponse « 401 Unauthorized ») en lui envoyant un challenge « nonce » inclus dans le message SIP. Le client calcule alors la réponse sur la base de ce challenge et d'une clé pré-partagée avec le serveur et la renvoie dans une requête REGISTER. Si cette réponse correspond à celle calculée par le serveur, ce dernier envoie au client une réponse « 200 OK » qui confirme son enregistrement.

La syntaxe des messages SIP comporte des en-têtes destinés pour l'authentification comme l'indique le Tableau 6. Le champ « WWW-Authenticate » est utilisé dans la réponse « 401 Unauthorized » envoyée par le serveur contenant le challenge. La réponse de l'UA (User Agent) à ce challenge inclut le champ « Authorization » dans l'en-tête SIP contenant les accréditations de l'usager.

Prenons l'exemple suivant, donné par la figure ci-dessous, qui illustre l'authentification dans le cas d'un enregistrement. En recevant une requête REGISTER d'un client SIP, le serveur répond par le message, présenté dans la figure, qui comporte l'en-tête « WWW-Authenticate » spécifiant le type d'authentification attendue et intégrant le challenge « nonce ».

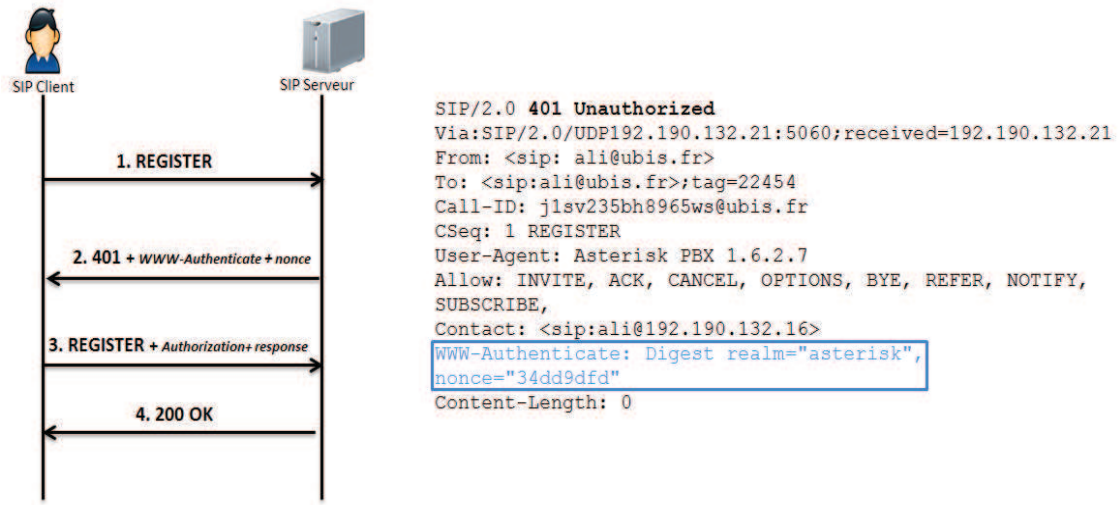



***Figure 30 :*** *L'authentification « HTTP Digest ».*

Le client doit alors calculer la réponse en se basant sur le challenge reçu et en utilisant la fonction de hachage MD5 [RFC 1321] tout en appliquant la formule suivante :

**response= H (H (username||realm||password) ||nonce|| H (Method||Request - URI))**

Dans notre exemple, la réponse est :

**response= H (H (ali||asterisk||password) ||34dd9dfd|| H (REGISTRER||SIP – ubis.fr))**

Par la suite, le client envoie un nouveau message REGISTER (Figure 31) contenant l'en-tête « Authorization » avec la réponse dans le champ « Response ».

```
REGISTER sip:ubis.fr SIP/2.0
Via:SIP/2.0/UDP192.190.132.21:5060;
From: <sip: ali@ubis.fr>
To: <sip:ali@ubis.fr>
Call-ID: j1sv235bh8965ws@ubis.fr
CSeq: 2 REGISTER
Expires: 900
Contact: <sip:ali@192.190.132.21:5060>
Authorization: digest username="ali", realm="asterisk", nonce="34dd9dfd",
Response="6c13de87f9cde9c44e95edbb68cbdea9",uri="sip:ubis.fr"
User-Agent: X-Lite
Content-Length: 0
```

Champs contenant la valeur *response (Accréditations)* calculée par l'usager SIP pour s'authentifier auprès de serveur

***Figure 31 :*** *Le message REGISTER.*

SIP offre donc une sécurité minimale à travers l'authentification HTTP Digest. Cette méthode est simple et la plus répandue malgré qu'elle présente quelques limites. Elle est

sensible aux attaques par dictionnaire basées sur les valeurs de hachage interceptées dans le cas où des mots de passe courts ou faibles sont utilisés. Un autre inconvénient est le manque de mécanismes assurant la confidentialité et l'intégrité des messages SIP. Pour des raisons de compatibilité, HTTP Digest ne favorise pas l'authentification mutuelle des participants dans un contexte SIP.

- **S/MIME**

S/MIME [RFC3851] est un standard proposé par l'IETF qui peut être utilisé afin de protéger les messages SIP en offrant plus de flexibilité vu qu'il permet de sélectionner les portions du message à protéger. SIP fait recours également à des fonctionnalités fournies par S/MIME qui permettent d'assurer la confidentialité, l'intégrité et l'authentification, en utilisant un mode tunnel et des mécanismes de chiffrement à clé publique et de signature.

Contrairement à HTTP Digest, l'authentification mutuelle des participants est envisageable. En plus, l'authentification de bout en bout est garantie en utilisant des certificats. En revanche, la confidentialité et l'intégrité de tout le message SIP ne peuvent pas être protégées. En effet, pour la confidentialité, le chiffrement de bout en bout de tout le message ne convient pas au contexte SIP vu que les nœuds intermédiaires doivent avoir accès à certains champs de l'en-tête du message SIP pour les traiter et router le message vers la bonne destination. Similairement, l'intégrité est assurée pour certains champs seulement car les serveurs SIP peuvent rajouter des champs.

De plus, cette solution présente quelques autres limites. En fait, S/MIME exige l'existence d'une infrastructure à clé publique (PKI, Public Key Infrastructure). Sinon, les clés publiques échangées vont être auto-signées, ce qui rend l'échange initial des clés susceptible aux attaques Man-in-the-middle. En plus, S/MIME fait augmenter la taille des messages SIP d'une manière considérable. S'ajoute à ceci que les paires adresses/certificats doivent être associées à l'utilisateur et non pas à la machine afin de favoriser la mobilité. En effet, dans le cas contraire, l'utilisateur sera obligé à utiliser cette machine.

***Discussion***

*En résumé, SIP est un protocole de signalisation qui assure le contrôle et la gestion de sessions. Il spécifie un ensemble de messages échangés selon le modèle requête/réponse permettant d'établir, modifier et clôturer une session. Cet échange implique alors la participation et l'intervention de différents acteurs appartenant à plusieurs zones de confiance. Ceci soulève des soucis pour la sécurisation de cet échange et la session en général.*

*SIP fait recours à des mécanismes de sécurité prédéfinis tels que HTTP Digest, S/MIME, TLS et IPsec afin d'assurer l'authentification des usagers, l'intégrité et la confidentialité. Cependant, ces solutions présentent souvent des problèmes d'interopérabilité et de compatibilité avec les implémentations existantes. Par exemple, HTTP Digest est généralement supporté et déployé avec sa version d'authentification simple tandis que l'authentification mutuelle devient de plus en plus nécessaire dans les nouvelles architectures. De plus, ces mécanismes assurent une sécurité qui est difficilement maitrisée à grande échelle. En effet, S/MIME nécessite une gestion de certificats ce qui n'est pas évident avec les architectures existantes étendues sur plusieurs domaines.*

## 7.5. Le protocole HTTP

Le protocole HTTP est devenu une interface d'accès aux applications Web. C'est un protocole simple, stable et largement déployé. Sa structure autorise le transport de différents types de messages à savoir XML-RPC ou SOAP, au même titre que de simples messages HTTP. Le principe de HTTP s'appuie sur le modèle client/serveur, mais en le limitant à la sémantique de la transaction Requête/Réponse (Figure 32). Un client HTTP émet une Requête à un serveur HTTP pour demander une ressource. Ce dernier retourne un message Réponse. A l'heure actuelle, il existe deux versions de HTTP. La première version HTTP/1.0 est un standard de l'IETF décrit dans la RFC 1945. Cette version supporte les serveurs HTTP virtuels, la gestion de cache et l'identification. La deuxième version HTTP/1.1 est décrite dans la RFC 2068 de l'IETF, puis dans la RFC 2616. Cette version ajoute le support du transfert en pipeline et la négociation du type de contenu (format de données, langue).

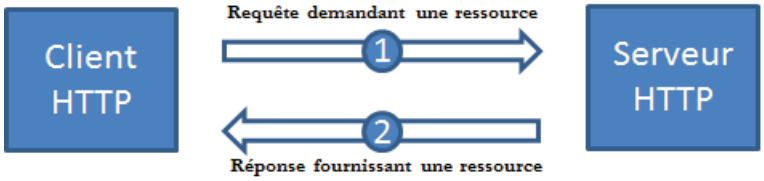



***Figure 32 :*** *Transaction Requête/Réponse.*

Le mode de fonctionnement de HTTP est simple et transactionnel comme le montre la figure ci-dessus. On peut remarquer que c'est le client qui ouvre la connexion, mais c'est le serveur qui la ferme. Ceci explique le fait que le protocole HTTP est sans connexion au niveau de la couche application. En effet, la connexion ne reste pas établie entre les différentes requêtes qu'un client envoie à un même serveur. Cela signifie que les requêtes sont traitées indépendamment les unes des autres. De plus, HTTP est un protocole sans état, aucune information n'est conservée entre deux transactions.

Au niveau de la sécurité, le protocole HTTP offre deux mécanismes de base pour assurer l'authentification et le chiffrement. L'authentification au niveau du protocole HTTP est basée généralement sur le couple login/mot de passe. Il existe deux méthodes d'authentification: Basic et Digest. Ces deux méthodes sont normalisées et définies dans la RFC 2617 []. Le serveur spécifie la méthode d'authentification dans l'entête "WWW-Authenticate" comme décrit précédemment. La différence entre ces deux méthodes apparait au niveau de cryptage des données échangées sur le réseau. En effet, la première méthode Basic est très simple à utiliser mais peu sécurisée. Le niveau de sécurité des informations échangées est faible malgré l'encodage des secrets (login/mot de passe) de l'utilisateur avec " base64", puisque le décodage est une opération facile à effectuer. La deuxième méthode Digest est plus sécurisée et permet d'éviter la transmission en clair des informations sensibles sur le réseau. Cette méthode applique le chiffrement avec l'algorithme MD5 sur le nom d'utilisateur et le mot de passe. Ainsi, les accréditations de l'utilisateur ne sont pas transportées en clair dans le réseau. Cependant, cette méthode n'est qu'une tentative pour remédier aux problèmes reconnus avec la méthode Basic, elle n'est pas notamment exempte d'autres failles.

La RFC [2617] définit le protocole HTTPS comme étant le protocole HTTP sécurisé. La phase de sécurisation est réalisée grâce à la technologie TLS/SSL. Par conséquent, les

données transférées dans le réseau sont cryptées. Cela signifie que la communication est illisible par les tiers malveillants qui visent intercepter le trafic entre le client et le serveur. En effet, le principe de cette combinaison HTTP/SSL consiste à établir un canal de communication sécurisé entre le client et le serveur. Ceci permet à l'utilisateur de vérifier l'identité de la ressource (site) auquel il veut accéder grâce à un certificat.

HTTP permet également d'établir une session sécurisée. Pour faire persister les informations de l'utilisateur sans imposer une réauthentification, plusieurs technologies peuvent être utilisées à savoir l'utilisation des champs cachés, la réécriture des URLs ou bien l'utilisation des jetons.

Dans nos contributions, le choix est mis sur le jeton de session comme mécanisme pour faire persister la session. Ce mécanisme n'est pas exempt des attaques les plus fréquentes à savoir « Session hijacking », « Replay attack », « Prédiction de session », « Attaque par force brute », etc. Mais dans cette thèse, nous n'allons pas traiter les aspects de sécurité liés à HTTP.

***Discussion***

*HTTP est un protocole sans état et sans connexion, sa structure permet le transport de messages XML-RPC ou SOAP, au même titre que de simples messages http. Au niveau de la sécurité, HTTP offre deux mécanismes de base tels que l'authentification avec les deux méthodes (Digest, Basic) et le chiffrement des données avec la combinaison HTTP/SSL. En plus, il supporte les jetons de sécurité assurant ainsi l'authentification unique (SSO) et la propagation de session. C'est un protocole simple et capable de tolérer les temps de latence et la non-qualité de service inhérents à Internet.*

## 7.6. Le protocole SOAP

### 7.6.1 SOAP : Description

SOAP (Simple Object Access Protocol) a été défini à l'origine par IBM et Microsoft avant de devenir une recommandation du W3C. La version actuelle est la version 1.2. C'est un protocole de communications défini pour l'ensemble des échanges entre les différents composants d'une architecture Web Service. Il permet de garantir l'interopérabilité entre des applications distantes basées sur des langages différents tout en restant indépendant des spécificités des plates-formes et des systèmes. En plus, il permet de pallier aux insuffisances des protocoles existants, comme CORBA (Common Object Request Broker Architecture), qui ne permettait pas le passage à l'échelle. Ainsi, SOAP offre la possibilité de construire des applications hétérogènes et réparties.

SOAP est situé au niveau de la couche applicative et utilise le standard XML pour définir la structure et le contenu de ses messages. Il repose sur le protocole HTTP pour le transport d'information. Les messages SOAP sont composés d'une enveloppe contenant un entête et un corps. L'en-tête est un élément facultatif du message SOAP qui inclut des informations complémentaires pour le traitement des données (identification de l'émetteur du message, règles de sécurité pour la lecture du message, algorithme de chiffrement à utiliser pour la lecture du message, etc.). Il est aussi utilisé par l'ensemble des spécifications WS-Security. Quant au corps du message SOAP, il inclut les données propres au message et représente la requête ou la réponse SOAP.

SOAP est considéré comme le protocole de communication standard dans le cadre des Web services qui garantit un haut niveau d'interopérabilité et le passage à l'échelle. Cependant, les messages SOAP sont exposés à des attaques potentielles dont l'altération des messages est considérée la plus importante. De plus, un message SOAP passe à travers les pare-feux et les proxys sans nécessité de changer les règles de filtrage, grâce au protocole HTTP. Ceci augmente le risque d'infiltration des barrières de sécurité établies par les organisations. S'ajoute à ceci que le langage XML, sur lequel se base le message SOAP, peut présenter des menaces. En effet, il existe de nombreuses attaques XML. On peut citer par exemple l'envoi des messages malicieux en insérant des éléments assez longs qui peuvent engendrer des perturbations de fonctionnement et des débordements de mémoire.

Pour faire face aux différentes attaques, plusieurs mécanismes sont mis en place pour assurer la sécurité des messages. La spécification *WS-Security* [9], initialement proposée par Microsoft, IBM et Verisign, représente un Framework de sécurisation des Web Services. WS-Security est normalisé par OASIS et publié en février 2006, ayant actuellement la version 1.1. Il a comme objectif de définir un cadre générique de sécurisation des messages SOAP. La flexibilité des technologies implémentées dans WS-Security assure la capacité de protéger différentes parties du message de différentes manières en fonction des contraintes imposées par les différents destinataires et intermédiaires. Toutefois, WS-Security n'offre pas une sécurité exhaustive contre l'ensemble des menaces auxquelles sont exposés les messages SOAP. Les technologies implémentées dans WS-Security ne garantissent que la confidentialité via le chiffrement, la non-répudiation et l'intégrité via la signature et le transport des éléments d'authentification via les jetons de sécurité.

Les fonctions proposées à ce titre sont, le transport des jetons d'authentification, le chiffrement et la signature. Elles sont implémentées à travers l'utilisation des extensions dans les en-têtes SOAP.

Ces extensions sont construites à partir des éléments suivants :

- Les jetons de sécurité : pour le transport des informations de sécurité et d'authentifications auprès du destinataire du message.
- *XML Encryption* : qui offre les fonctions de chiffrement des données et des informations critiques de l'en-tête.
- *XML Signature* : pour la signature et le contrôle d'intégrité de tout ou partie du message et de l'en-tête

### 7.6.2 SOAP : les jetons de sécurité

Le rôle des jetons de sécurité dans WS-Security est de transporter les informations d'authentifications liées aux acteurs de la chaine applicative. WS-Security précise l'utilisation d'un nouvel en-tête de sécurité <*wsse:Security*> qui contient l'ensemble des éléments nécessaires au traitement des informations sécurisées transportées par le message SOAP. Chaque en-tête est destiné à un acteur spécifique. Par conséquent, il est possible qu'un message SOAP contient plusieurs en-têtes <*wsse:Security*>.En revanche, si un service intermédiaire souhaite rajouter des informations sécurisées à à un destinataire déjà identifié dans un en-tête <*wsse:Security*>, cet intermédiaire doit rajouter un élément dans l'entête existant et non créer un nouvel en-tête.

Trois types de jetons sont définis par WS-Security :

- Jeton basé sur le couple login/password

- Jetons binaires
- Jetons XML : SAML

Nous décrivons, ci-dessous, ces trois méthodes en détail.

- ***Jeton basé sur le couple login /password***

Ce jeton est le plus simple mais il offre le moins de garantie en termes de sécurité. Comme son nom l'indique, il est utilisé pour transmettre l'identifiant de l'utilisateur. Le jeton est identifié par l'élément *UsernameToken* illustré dans le schéma ci-dessous. Ce jeton est mis en œuvre selon le mode d'authentification « Basic » ou « Digest ».

Dans le cas où l'authentification serait effectuée par l'application elle-même, un mot de passe peut être transporté dans le jeton en clair (Basic) ou via un hash (Digest). La première méthode est fortement déconseillée à l'exception des mots de passe à usage unique (OTP - One-Time Password). Les mots de passe « hachés » sont construits à partir d'une valeur aléatoire <nonce> afin d'éviter les attaques par re-jeu et d'un élément Timestamp <created> selon la formule suivante : Passsword_Digest = Base 64(SHA-1(nonce+created+password)).

```
<xs:element name="UsernameToken">
<xs:complexType>
<xs:sequence>
<xs:element ref="Username"/>
<xs:element ref="Password" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="Id" type="xs:ID"/>
<xs:anyAttribute namespace="##other"/>
</xs:complexType>
</xs:element>
```

***Figure 33 :*** *L'élément « UsernameToken ».*

- ***Jeton binaire***

WS-Security définit également l'élément *BinarySecurityToken* (Figure 34) afin de permettre l'utilisation générique des jetons non-XML tels que les certificats X509 et les tickets Kerberos. Ces jetons sont obtenus auprès d'une autorité de certification. Ils acceptent la signature et le cryptage.

```
<xs:element name="BinarySecurityToken">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="Id" type="xs:ID" />
<xs:attribute name="ValueType" type="xs:QName" />
<xs:attribute name="EncodingType" type="xs:QName"/>
<xs:anyAttribute namespace="##other"
processContents="strict" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
```

***Figure 34 :*** *L'élément « BinarySecurityToken ».*

- ***Jeton SAML***

SAML (Security Assertion Markup Language) est une grammaire XML normalisée par l'OASIS qui permet d'attester l'authentification à un utilisateur qui désire accéder à plusieurs

services en évitant aussi de se ré-authentifier plusieurs fois autant qu'il y a de services invoqués. SAML permet ainsi d'assurer le Single Sign On, l'authentification étant assurée par un web service dédié à cette tâche.

Une fois l'authentification est effectuée, ce service de sécurité distribue aux autres services des assertions ou des jetons de sécurité SAML, à travers desquels il certifie avoir bien mené l'authentification des accédants et s'engage sur leur identité.

Le jeton SAML est encapsulé dans un message SOAP. Il utilise un document XML qui contient les éléments suivant :

<issuer> : l'identifiant du service de sécurité qui a mené l'authentification,

<subject> : l'identifiant de l 'utilisateur qui souhaite accéder aux services,

<ds : signature> : la signature au format XML Signature, qui certifie l'authentification.

Il peut aussi contenir d'autres attributs, dont certains peuvent être chiffrés via XML Encryption afin d'assurer leur confidentialité vis-à-vis des tiers.

Le tableau suivant récapitule les avantages et les inconvénients de différents types de jetons déjà décrits.

| Type de jeton | Avantages | Inconvénients |
|---|---|---|
| **Login/ password** | - Simple à mettre en place<br>- Indépendant de l'environnement et de la technologie | Peu sécurisé |
| **Binaire** | - Très sécurisé<br>- Interopérabilité | Pour les tickets Kerberos, fonctionnent uniquement dans un environnement Kerberos. Les certificats X509 nécessitent une infrastructure à clé publique (PKI). |
| **SAML** | - Mécanismes d'échange d'information d'authentification et d'autorisation<br>- Authentification unique sécurisé | Peu d'implémentations existantes |

***Tableau 8 :*** *Types des jetons : avantages et inconvénients.*

### 7.6.3 SOAP : les mécanismes de sécurité

Pour assurer l'intégrité et la confidentialité des messages SOAP contenant les jetons de sécurité, deux mécanismes de sécurité sont utilisées la signature et le chiffrement. Pour cette raison, nous allons expliquer brièvement dans ce qui suit ces deux mécanismes.

- **Signature des messages avec *XML Signature***

Le support de XML Signature est obligatoire dans le cadre de la norme WS-Security. La principale recommandation de WS-Security au regard de l'utilisation de XML Signature concerne la possibilité d'enrichir un message SOAP de nouveaux éléments tout au long de la chaine applicative et jusqu'à la destination tout en assurant l'intégrité du message.

XML Signature est une spécification du W3C qui définit des mécanismes de signature sur un sous ensemble d'un flux XML. Il devient ainsi possible, lors de la transmission de messages SOAP, d'authentifier l'expéditeur, d'assurer l'intégrité des données et d'assurer la non-répudiation des échanges. L'utilisation de XML Signature suppose bien entendu la mise en œuvre de procédures pour distribuer des certificats numériques aux services et pour vérifier les signatures.

❖ **Chiffrement des messages avec *XML Encryption***

Le support de XML Encryption est également obligatoire dans le cadre de la norme ws-Security. C'est une spécification du W3C qui permet d'assurer la confidentialité des informations en chiffrant une partie ou la totalité d'un document XML. Elle fournit une grammaire XML standardisée pour décrire les méthodes de chiffrement des données. Elle permet d'appliquer des mécanismes de chiffrement à clé secrète ou à clé publique sur une partie de message SOAP. Il devient ainsi possible lors de la transmission de messages XML, d'assurer la confidentialité d'une partie des données vis-à-vis certains services. Une telle fonctionnalité semble nécessaire par le fait que certains web services intermédiaires peuvent ne pas être autorisés à accéder à certains contenus.

***Discussion***

*En résumé, SOAP est un protocole léger simple et extensible basé sur XML permettant l'échange d'informations dans un environnement distribué et décentralisé. SOAP s'appuie sur des protocoles de transport comme HTTP qui facilite la communication et évite les problèmes de proxy et des pare-feu, qui peuvent être franchis sans problème. Il est très souvent désagréable d'utiliser un autre protocole que http ou POP/SMTP à cause des firewalls qui doivent être reconfigurés engendrant ainsi des trous de sécurité. De plus, SOAP est interopérable, c'est-à-dire indépendant des plates-formes et des langages de programmations. En revanche, SOAP est un protocole limité par ses faibles performances. En effet, il peut alourdir les échanges par le nombre important d'informations qu'impose le format XML. En plus, si un Web Service est assez complexe, le fait d'utiliser SOAP et donc XML peut ralentir le temps de réponse (la vitesse) des échanges. Concernant la sécurité, elle est intégrée dans les messages SOAP. En effet, ce protocole supporte l'authentification unique basée sur les jetons, en utilisant le WS-Security. Il assure la confidentialité et l'intégrité des messages en utilisant respectivement les standards XML Encryption et XML Digital Signature.*

## 7.7. Conclusion

| Niveau | Besoins | Solutions existantes | Limites des solutions existantes | Défis / Verrous |
|---|---|---|---|---|
| **Service** | Authentification unique pour tous les services sollicités | **Single Sign On :**<br>- Simplification et flexibilité des procédures | - Authentification limité dans un seul domaine.<br>- Pas de flexibilité par | - Authentification unique pour tous les services sollicités pour un utilisateur mobile |

| | | | | |
|---|---|---|---|---|
| | | d'authentification.<br>- Authentification unique. | rapport aux niveaux d'authentification requis --> un seul niveau d'authentification pour tous les services dans une session. | dans un environnement hétérogène.<br>- Service d'authentification qui s'adapte au contexte de l'utilisateur et au niveau de sécurité requis. |
| | Délivrance de service par plusieurs fournisseurs<br>**Fédération :**<br>- Etablissement d'un cercle de confiance entre plusieurs partenaires.<br>Partage d'authentification dans un environnement multi-domaines. | **Open ID**<br>**Liberty Alliance**<br>**WS-federation** | | - Flexibilité d'un service d'authentification unique dans un environnement trans-organisationnel. |
| **Protocole** | Etablissement et contrôle de la session | **SIP** | - Niveau de sécurité minimal.<br>- Pas de gestion de la mobilité de l'utilisateur (changement du terminal) --> Pas de continuité de session.<br>- Signalisation assurée uniquement au niveau réseau. | - Signalisation plus flexible qui arrive jusqu'au niveau service --> QoS de bout en bout.<br>- Impact de la mobilité sur la sécurité.<br>- Session unique, mobile, dynamique et sécurisée.<br>- Renforcer le niveau de sécurité du protocole. |
| | Transport des données | **HTTP** | | - Transport du jeton lors de la consommation du service. |
| | Echange des messages entre les composants de service (Web services) | **SOAP** | | - Des services convergents de différentes natures (Web, Telco, IT). |
| **Session** | Session unique avec une continuité de sécurité | **Le jeton** | - Transfert du jeton dans un seul domaine.<br>- Il ne supporte pas la continuité de la session (changement de l'identifiant de la session lors de la régénération du jeton). | - Propagation du jeton entre plusieurs plates-formes de service (y inclus le terminal).<br>- Continuité et unicité de la session mobile.<br>- Confidentialité et intégrité du jeton. |

***Tableau 9 :*** *Tableau récapitulatif.*

# 8

# Proposition

Dans cette partie, nous définissons, en premier lieu, la session UBIS sur laquelle se base notre étude et nous identifions ses défis de sécurité (§8.1). En second lieu, nous décrivons le protocole SIP+ (§8.2) en détaillant ses principes et formats et en présentant la version SIP+ basée sur les jetons que nous avons proposée. Puis, nous consignons notre proposition en décrivant les différentes phases de notre session. Finalement, afin de faire face aux attaques potentielles de SIP+, nous proposons les mécanismes nécessaires de sécurité pour avoir un SIP+ sécurisé.

## 8.1. Session UBIS

### 8.1.1 Qu'est ce qu'une session ?

Dans le paysage des télécommunications, la notion de session décrit une succession d'interactions entre les deux extrémités de transport permettant une mise en relation lors de la transmission des données entre les émetteurs et les récepteurs.

Les services de transport assurent les communications « point à point », c'est à dire entre deux machines (M2M). Cependant, le modèle de référence OSI (Open System Interconnection) s'étend aussi aux communications multipoints à savoir celles en Etoile où la session représente l'ensemble des communications point à point avec un interlocuteur engagé dans tous les échanges (dans la mise en relation), et celles par Diffusion où tous les interlocuteurs reçoivent tous les messages.

La session est définie de plusieurs façons dans les normes selon le domaine d'application.

Dans le domaine IP multimédia, une session établit une communication entre un ensemble d'émetteurs et de récepteurs afin de transmettre des données. Ce type de session est supporté par le Core Network IP multimédia et par des supports de connectivité IP par exemple des supports de type GPRS. D'autre part, dans le domaine IP, la session de commutation de paquets est définie comme une connexion logique entre les parties impliquées dans une communication basée sur PS (Packet Switched). Elle est établie par le protocole PDP (Packet Data Protocol) dans un domaine PS pour délivrer des paquets de données.

Dans les cas des « sessions web », le terme de la session désigne une connexion de niveau applicatif, voire un contexte partagé par plusieurs connexions de niveau application sans support protocolaire. C'est un usage dérivé des systèmes d'exploitation.

Selon l'architecture IMS, une session multimédia établit des communications entre plusieurs terminaux, en utilisant un réseau IP. Elle permet de rajouter des services en temps réel au cours d'une même communication. Ainsi, un utilisateur peut ouvrir plusieurs sessions multimédia lors d'une communication**,** par exemple, rajouter une session de chat à celle de la vidéo, envoyer une photo pendant la conversation.

Par ailleurs, TINA (Telecommunications Information Networking Architecture) introduit trois types de session : la session d'accès, la session de communication et la session de service pour identifier les différents acteurs intervenant dans le service demandé par l'utilisateur.

La session d'accès couvre les modalités et les interactions nécessaires à la session (par exemple l'authentification, la sélection du profil de service) lors de la phase de connexion d'un utilisateur au système. Ce type de session permet aux utilisateurs la création et la gestion de la session service (par exemple combiner des sessions et utiliser plusieurs services).

La session de service représente un environnement regroupant les informations et les fonctionnalités liées aux capacités d'exécution, de contrôle et de gestion des services. C'est une instance d'un service spécifique incluant les informations nécessaires à la négociation de QoS, de la sécurité, l'utilisation du service, la communication des ressources ainsi que les informations pour le contrôle des relations entre les acteurs intervenant au cours de la session de service. C'est par exemple le cas pour des services à base de transfert de fichiers (FTP).

La session de communication représente le service des flux de connexion (la relation) entre le terminal de l'utilisateur et le réseau. L'existence d'une telle session assure à l'utilisateur la possibilité de communiquer avec le réseau, ce qui lui permet dans le cas échéant d'avoir les ressources de communications requises pour établir des connexions de bout en bout, donc ouvrir une session de services. Une session de communication peut gérer la QoS, établir modifier et terminer plusieurs connexions. Elle supporte aussi de divers types de connexions multipoints et multimédia.

En conclusion, la session est toujours la mise en relation temporelle, concernant un service dédié. Mais pour prendre en compte les besoins de NGN (Next Generation network), notamment les impacts de toutes les mobilités, UBIS a défini une mise en relation qui prend en compte la mobilité de service avec le Handover Sémantique ou Handover de service.

La question qui se pose est donc la sécurité de ce nouveau type de session.

### 8.1.2 Session UBIS : Définition, Le défi de la sécurité

La session UBIS se manifeste au niveau horizontal de notre architecture sous la forme d'une composition de services, et au niveau vertical sous la forme d'une agrégation de services. L'approche horizontale permet de composer les ES (Eléments de Service) à chaque niveau de traitement de l'architecture. Cette composition personnalisée qui constitue le concept dénommé VPxNs (x = E: Equipement, C: Connectivité, S: Service et U: Utilisateur) regroupe à chaque niveau les services qui sont choisis et gérés de façon dynamique et participant à la fourniture d'un service global pour la session de l'utilisateur.

Dans le cadre de cette thèse nous mettons l'accent essentiellement sur la sécurité au niveau service et au niveau équipement (Device). Elles sont supportées respectivement par

le concept de VPSN (Virtual Private Service Network) et VPDN (Virtual Private Device Network). Nous allons nous contenter d'expliquer ce concept au niveau service, sachant qu'il est similaire pour tous les autres niveaux.

Le concept VPSN désigne comme son nom l'indique un réseau privé virtuel de service. Le VPSN est un r*éseau* d'éléments de service. Il repose sur le modèle NLR (Nœud, Lien, Réseau) qui s'appuie sur le principe d'abstraction, défini (l'encyclopédie universelle), par le résultat des quatre opérations suivantes, simplification, généralisation, sélection et schématisation. Par conséquent, les éléments de services sont considérés comme des nœuds mutualisables qui sont reliés par des liens lâches suivant une logique de service et selon les préférences de l'utilisateur pour construire le réseau dénommé VPSN.

Le VPSN est un réseau *privé* puisque il est construit pour répondre à une demande de service d'un utilisateur donné selon ses exigences. Ainsi il représente l'application de l'utilisateur qui sera construite par demande de service. L'utilisateur est autorisé à accéder tout au long de sa mobilité spatiale et temporelle aux éléments de services choisis formant son VPSN. Le VPSN établit dynamiquement et sans couture une session de service qui respecte la logique de services sollicités, le contexte ambiant et les préférences de l'utilisateur en tenant compte de tout changement du à la mobilité ou à une dégradation de QoS ou une faille de sécurité.

D'autre part, le VPSN est un réseau *virtuel* puisqu'il repose sur des éléments de service mutualisables, donc partagés par plusieurs utilisateurs dans différents VPSN. C'est un réseau de *service* qui est supposé rendre des services de natures différentes, à savoir des services applicatifs, des services de personnalisation, mais aussi des services de gestion, de sécurité, etc. Ces éléments de services sont sélectionnés et pré-provisionnés selon une QoS répondant aux besoins de l'application, indépendamment des souhaits de l'utilisateur.

Pour finir, le concept VPxN (Figure 35) représente une composition horizontale personnalisée qui est flexible et à couplage lâche, et elle s'adapte d'une manière transparente dynamique sans couture à tout évènement (QoS, Sécurité, mobilité) en temps réel. En plus de l'approche horizontale basée sur la composition de services, l'approche verticale est basée sur l'agrégation de services. En effet, cette approche consiste à mettre en place la session User-Centric dynamique et à établir les liens entre les ESs choisis dans chaque niveau de visibilité. Donc cette vue verticale consiste à pré-provisionner les éléments de service de types SE, les éléments de réseaux (réseaux virtuels, etc.), et les équipements (terminaux, routeurs, etc.) qui répondent aux mieux aux préférences fonctionnelles et non-fonctionnelles (QoS, sécurité) des utilisateurs et de leurs contextes. Par la suite, la session de l'utilisateur est établie et il peut accéder alors à tous les services pré-provisionnés faisant partie de sa session. Cette agrégation verticale représente la session UBIS qui s'adapte d'une manière dynamique et transparente à tout changement lié aux besoins de l'utilisateur et à leur contexte.

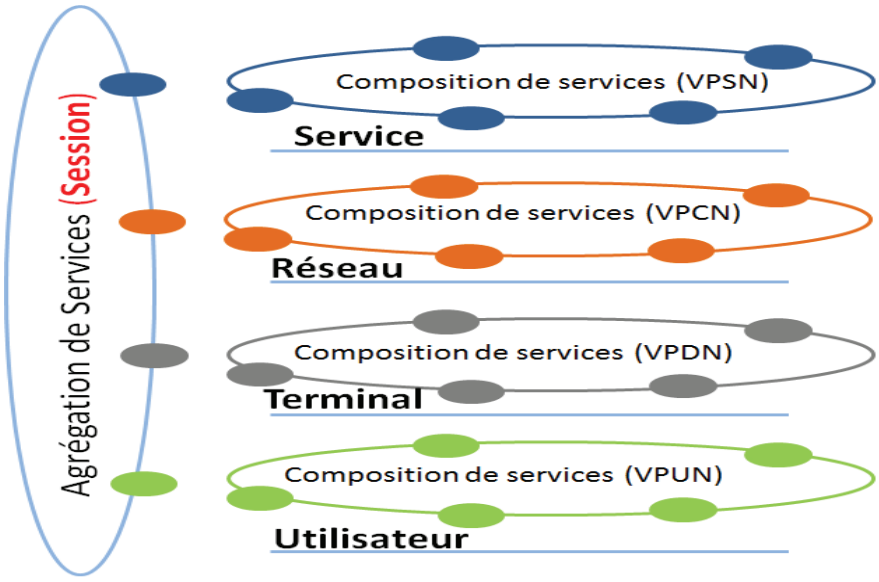



***Figure 35 :*** *VPxN et la session.*

Dans cet environnement ouvert, mobile, hétérogène et User-Centric, l'utilisateur est au centre du dispositif. Il se déplace bien sûr, mais il peut aussi changer de terminal, avoir plusieurs préférences, solliciter des services auprès de plusieurs fournisseurs. De plus, l'utilisateur d'aujourd'hui désire avoir accès à tout type de services dans une session unique qui se veut sécurisée et qui offre une continuité de service et de QoS.

Mais comment respecter ces objectifs de sécurité, de continuité ainsi que de l'unicité de la session dans un contexte mobile à plus d'un titre ?

En effet, nous avons d'abord la *mobilité du terminal* qui traduit le changement de localisation de ce terminal sans changement d'accès au service. L'utilisateur doit continuer à accéder à toute la liste des services auxquels il a souscrit. Nous pouvons aussi avoir la *mobilité de l'utilisateur* qui représente le passage de l'utilisateur d'un terminal à un autre, au cours de la même session qui doit maintenir son niveau de sécurité. Et puis, il y a aussi le cas de la «*mobilité de service*» qui est utilisé par le système pour respecter la QoS de bout en bout ou pour satisfaire les préférences de l'utilisateur. On constate donc que la mobilité impacte fortement la délivrance du service de bout en bout sécurisé et impose l'introduction de mécanismes pour pallier les coupures ou interruptions de service au sens large du terme.

Nous nous intéressons à la continuité et au maintien de la sécurité tout au long de la session.

## 8.2. Le protocole proposé SIP+

### 8.2.1 SIP+ : Principes et Formats

Afin de satisfaire les besoins de continuité de sécurité et de QoS dans la session centrée utilisateur, une signalisation plus flexible (niveau de service) s'avère nécessaire pour échanger des informations de QoS, des informations de sécurité (jeton) et des informations de l'utilisateur lors des différentes mobilités.

Pour ce faire, nous avons proposé une extension de la portée du protocole SIP existant, au niveau service de l'architecture pour offrir à l'utilisateur un service adapté à ses besoins et

à son contexte ambiant. Le protocole étendu est nommé SIP+ [50]. L'intérêt majeur du protocole SIP+ est de permettre une négociation de la qualité de service et de la sécurité pour chaque service demandé par l'utilisateur pendant la phase d'initiation de la session user-centric.

Ce protocole est utilisé bien évidement pour établir, modifier ou bien terminer des sessions « user-centric ». Il permet de contrôler la création du VPSN pendant la phase d'établissement de session via le message SIP+ INVITE, de gérer le VPSN pendant la phase d'exploitation via le message SIP+ NOTIFY et également de libérer les ressources du VPSN via le message SIP+ BYE.

Afin de réaliser la continuité de sécurité (l'authentification unique), SIP + a la capacité de faire circuler le jeton entre les Security Agent de chaque plate-forme hébergeant les composants de service sollicités. Un jeton étendu permettra d'assurer les autorisations nécessaires pour construire le VPSN, et l'enchainement de ces composants selon la logique de service. Dans la même logique, nous utilisons le SIP+ pour construire le VPDN qui représente le réseau de terminaux de l'utilisateur, c'est à dire le PAN actif durant la session.

La structure des messages SIP+ est similaire à ceux de SIP au niveau des en-têtes mais avec un corps de message enrichi par des informations relatives à la QoS et à la sécurité. La figure illustrée ci-dessus décrit la structure d'un message SIP+ INVITE.

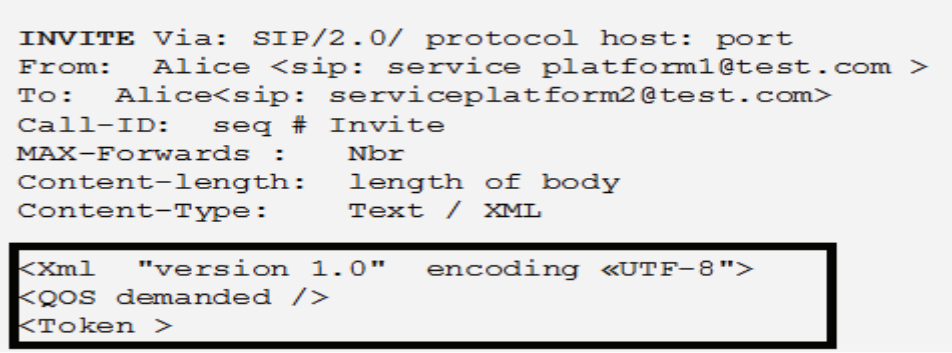



***Figure 36 :*** *Structure d'un message SIP+ Invite.*

Dans le paragraphe suivant nous allons décrire la structure du jeton de sécurité dans un message SIP+.

### 8.2.2 « Token-based SIP+ »

Afin d'assurer la gestion d'une session « user-centric », unique et sécurisée de bout en bout, nous proposons SIP+ basé sur le jeton. Ce mécanisme garantit la continuité et l'unicité de la session et donc permet l'authentification unique de l'utilisateur à tous les services sollicités. Ce choix s'appuie sur le fait que les jetons sont flexibles et peuvent circuler d'une plate-forme à une autre en appliquant le niveau de sécurité le plus adéquat à ces derniers. En effet, nous pouvons ajouter de nouveaux champs qui décrivent la session dans le jeton. De plus, nous pouvons traiter les champs des jetons pour assurer la sécurité des données.

#### 8.2.2.1 « Token-based SIP+ » : Principe et Structure

Un des éléments importants, dans notre proposition, qui participe au management de la sécurité de la session et qui garantit son unicité, est le *jeton*. Il est utilisé pour le contrôle

d'accès, à savoir l'authentification, et pour le maintien d'une session sans couture (seamless). Il est créé et mis à jour par le *Service jeton* lors de la création du VPDN et VPSN c'est-à-dire lors de la création de la session de l'utilisateur.

En effet, le *Securityware* offre aux utilisateurs un service d'authentification unique basé sur le *Jeton*. Ce dernier est l'un des mécanismes le plus puissant et le plus utilisé dans les environnements trans-organisationnels et orientés service pour l'échange des informations de sécurité entre les différentes plates-formes de service ou terminaux.

En fait, le *jeton* représente une collection d'informations transmise par le *Securityware* au *Security Agent*, déployé sur le terminal de l'utilisateur, une fois que ce dernier est authentifié avec succès. Or, l'utilisateur sollicite des services hébergés sur différentes plates-formes de services qui sont protégées par des Security Agents. Pour cela, le jeton est propagé entre ces plates-formes et permet d'identifier l'utilisateur à chaque passage d'une plate-forme à une autre sans que l'utilisateur ait besoin d'une réauthentification.

Ce *jeton* est récupéré à chaque demande de service à partir du terminal de telle sorte que la plate-forme de services (qui peut être le terminal lui-même) puisse reconnaitre les demandes d'un même utilisateur. En conséquence, la session unique ainsi que la sécurité et la continuité de service sont maintenues automatiquement grâce à l'échange de *jeton* à chaque demande.

La structure et le modèle de données du *jeton* sont illustrés dans la Figure 37. Malgré le fait que les *jetons* ont (la même structure) une structure commune, ils peuvent être différents dans l'usage et dans la façon dont ils sont générés. Certains jetons peuvent contenir également des champs facultatifs (optionnels).

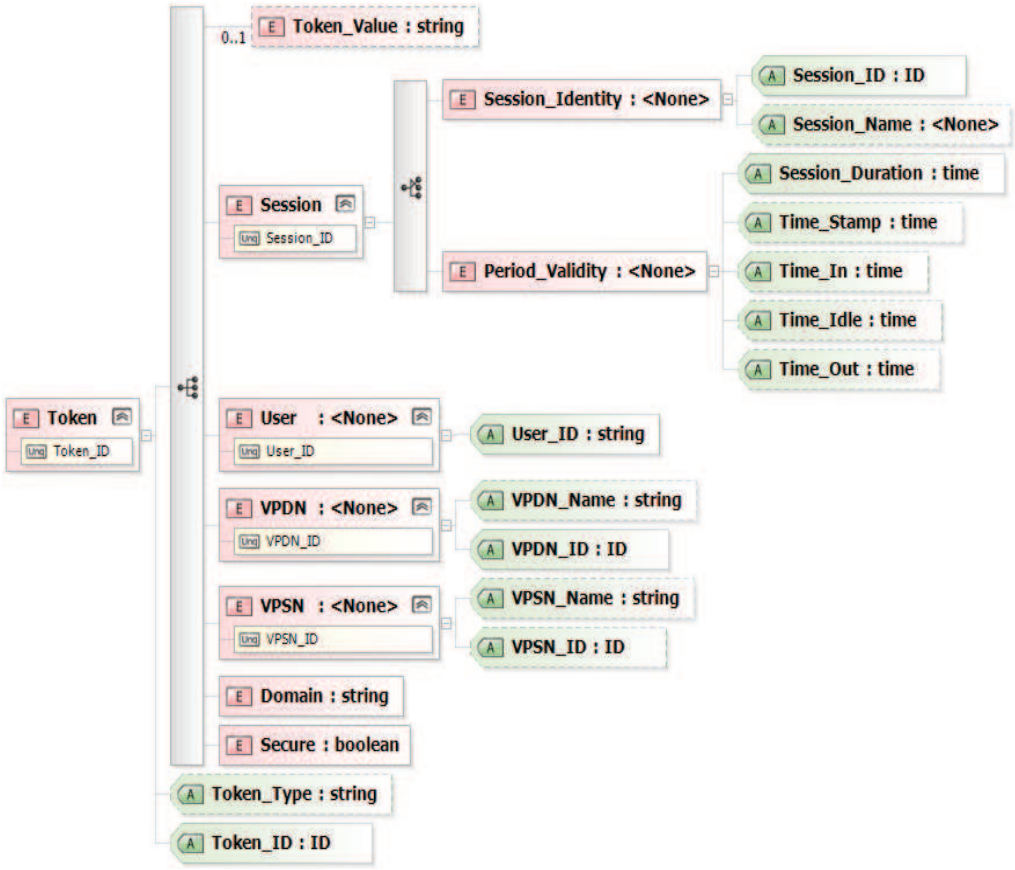



***Figure 37 :*** *La structure du jeton.*

La structure de la partie commune d'un jeton est la même quel que soit son type, elle contient les attributs et les éléments suivants :

- ***Token_Type*** : C'est un attribut facultatif qui précise le type du jeton.
- ***Token_ID*** : c'est un attribut obligatoire qui représente l'identifiant unique du jeton. Il permet de distinguer les jetons les uns des autres.
- ***Token_Value*** : C'est un élément facultatif qui peut être utilisé dans le cas de l'authentification.

✓ ***Les informations liées à la session***

Une session est caractérisée par ses attributs. Nous avons défini ceux qui permettent d'identifier, d'une manière unique, la session et ceux qui sont liés à la durée de validité de la session.

- ***Session_ID*** : un identifiant unique de la session de bout en bout. Cet attribut permet de distinguer les sessions des différents utilisateurs. C'est un attribut du jeton de session qui sera échangé entre le *Security Agent* et le *Securityware.*
- ***Session_Name*** : cet attribut a une valeur locale au niveau de notre *Securityware* qui permet de différencier les sessions du même utilisateur.
- ***Session_Duration*** : indique la durée de validité d'une session. Si cette durée s'est écoulée, la session courante n'est plus valide. Toutes les requêtes qui utilisent le jeton de cette session seront refusées par le Security Agent et le SecurityWare. Une nouvelle session sera par la suite initialisée.

- ***Time_In*** : indique la date de création d'une session.
- ***Time_Stamp*** : indique la date de la dernière requête demandée par l'utilisateur. Ce paramètre est obligatoire pour calculer le " Time_Idle".
- ***Time_Idle*** : définit la durée de temps à partir de laquelle le *Securityware* demande une réauthentification d'un utilisateur donné s'il n'a reçu aucune requête de sa part durant cette période. La réauthentification requise ne modifie pas les valeurs de la session courante.
- ***Time_Out*** *:* indique la date limite de terminaison d'une session. Cet attribut est mis à jour à chaque nouveau message SIP+. Si la valeur de cet attribut est nulle, cela veut dire que la session a expiré.

✓ ***Les informations liées à l'utilisateur***

- ***User_ID:*** c'est l'identifiant de l'utilisateur dans la session courante. Cet attribut peut être une valeur aléatoire générée par le Securityware comme il peut être une valeur parmi les caractéristiques de l'utilisateur comme son adresse IP ou son login.

✓ ***Les informations liées au VPDN***

- ***VPDN_ID****:* désigne l'identifiant unique de l'ensemble des terminaux sollicités par l'utilisateur durant sa session et qui construisent le VPDN. Cet identifiant reste unique, même lorsque le jeton est régénéré pour une session de longue durée.
- ***VPDN_Name***: cet attribut est une valeur locale au niveau de notre *Securityware* qui permet de différencier les VPDNs du même utilisateur.
- ***VPSN_ID*** *:* désigne l'identifiant unique de l'ensemble de services sélectionnés durant la session User-Centric. Si la session dure longtemps, le jeton sera régénéré dans un créneau horaire T donné. Nous devons avoir un identifiant unique pour cette session. Cet identifiant est le VPSN-ID qui sera encapsulé dans le jeton.
- ***VPSN_Name*** *:* cet attribut est une valeur locale au niveau de notre Securityware qui permet de différencier les VPSN du même utilisateur.
- ***Domain*** : Pour différencier entre les jetons des différents *Securityware*, chaque jeton doit être associé à un numéro de domaine.
- ***Secure*** : C'est une valeur Booléenne qui indique si le canal de transport est sécurisé.

#### 8.2.2.2 Les différents types de jetons

Dans notre contexte, nous avons défini trois types de jetons ayant une structure de données commune et contiennent différents profils durant les différents processus de la création de la session User-Centric comme le montre la Figure 38 à savoir l'initialisation de la session, la création du VPDN et enfin la création du VPSN.

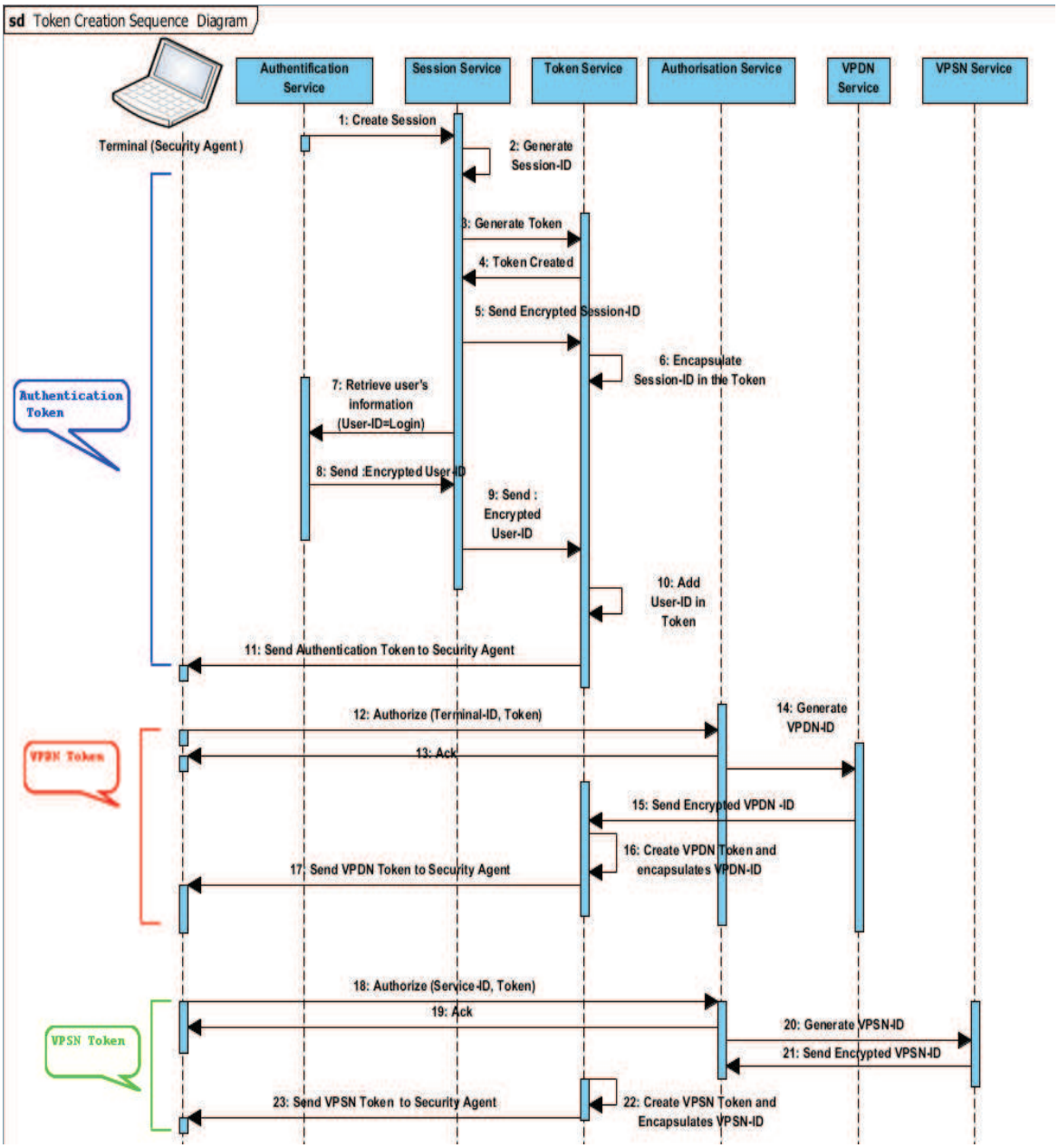



***Figure 38 :*** *Diagramme de séquence de la création des différents jetons.*

➢ **Jeton d'authentification** :

Quand l'utilisateur veut établir une session à partir de son terminal, il doit obligatoirement s'identifier et puis s'authentifier. Une fois passée la phase de l'authentification avec succès, l'initialisation de la session est effectuée par le *Session Service*. Il génère alors les attributs de la session courante. En effet, il crée les attributs qui permettent d'identifier la session à savoir le *Session_ID* qui est un identifiant unique et aléatoire. Après, le *Service Session* génère les attributs liés à la durée de validité de la session telles que le *Session_Duration* qui établit la durée de vie de la session, *Time _IN*, *Time_Stamp*, *Time_Idle* et *Time_Out.*

Ensuite, un jeton est créé par le *Service Token* dénommé ***Jeton d'authentification***. Il contient les informations nécessaires de la session et de l'utilisateur. Il encapsule les attributs « Session ID », « Session_Duration » et « User_ID ». Ce dernier attribut est

récupéré du service d'authentification. Il est l'identifiant de l'utilisateur dans la session courante.

- **Jeton VPDN:**

L'utilisateur initie sa session à partir de son terminal, le Service d'Autorisation doit alors vérifier s'il a les droits nécessaires pour créer son VPDN. Donc, une demande d'autorisation par rapport à l'utilisation du terminal est envoyée par le *Security Agent* au *Securityware*. Si la réponse est positive, le VPDN Service génère le *VPDN_ID.* Le *Token Service* récupère l'identifiant du VPDN et met à jour le jeton. Ce dernier est dénommé ***Jeton VPDN.*** Il repose sur le jeton d'authentification et est complété par les informations liées au VPDN en plus des informations liées à la session et à l'utilisateur.

- **Jeton VPSN**

Pour créer le VPSN de l'utilisateur, le *Security Agent* envoie une demande d'autorisation auprès de *Securityware* pour la création du VPSN. Si la réponse est positive, le « VPSN service » génère un *VPSN_ID*. En fait, le *Token Service* doit délivrer des jetons non prédictibles et uniques de session à chaque utilisateur pour faire face à certaines attaques. Ce jeton est composé par un ensemble de valeurs gérées par le *Securityware*. Pour cela, il doit, de manière transparente et continue, faire expirer et régénérer ces jetons dans la même session courante si celle-ci dure longtemps. Ce mécanisme permet de réduire la probabilité des tentatives de vol des sessions.

Le renouvellement des jetons de session est effectué suite à un nombre déterminé de requêtes, ou après une période de temps définie, etc. C'est pour cette raison, qu'il fallait avoir un identifiant unique qui ne sera pas modifié durant la même session d'un même utilisateur. Cet identifiant unique est le *VPSN_ID.*

Le *Token Service* récupère l'identifiant VPSN et l'encapsule dans le jeton. Celui-ci, après une mise à jour, contient toutes les informations et les attributs liés au VPSN, au VPDN, à la session et à l'utilisateur qui seront cryptés et chiffrés. Ce jeton est dénommé ***Jeton VPSN*.**

### 8.2.3 Les différentes phases de la session

En se basant sur le fait que notre architecture est composée de quatre niveaux de visibilité (service, réseau, terminal et utilisateur), la session « User-Centric » est considérée comme une agrégation de l'ensemble des composants de service de ces différents niveaux. Cette session sera maintenue pour préserver la continuité de la communication de l'utilisateur non seulement quand il change de réseau d'accès ou de terminal mais aussi lorsqu'il change ses services applicatifs. De plus, elle doit être « sans couture », unique et sécurisée.

Nous expliquons ci-dessous ces critères qui sont assurés durant les différentes phases de la session comme le montre la Figure 42.

- *Phase d'initialisation (Pré-provisionnement) :*

Lorsqu'un utilisateur désire initialiser sa session de service, le *Securityware* active les services d'identification et d'authentification selon ses exigences de sécurité et son contrat SLA. Une fois l'utilisateur identifié et authentifié avec succès, l'initialisation de la session est accomplie. Comme il existe plusieurs composants de service ubiquitaires fournissant la même fonctionnalité avec différents niveaux de QoS et de sécurité, le fournisseur de service

permet l'identification et la sélection des composants de service respectant les besoins de l'utilisateur en termes de sécurité et de QoS.

*1 Phase d'établissement (Provisionnement) :*

Cette phase implique deux étapes majeures: l'autorisation et la négociation de QoS. Au début, le *Security Agent*, qui est déployé dans chaque plate-forme de service (appartenant à un fournisseur de service) vérifie les autorisations de l'utilisateur auprès du *Securityware* pour avoir une décision concernant la création du VPSN (session de service).

Ensuite, lorsque l'utilisateur veut établir sa session de service (illustrée dans la figure), un processus pour la négociation de la QoS est exécuté afin d'identifier les composants de services qui participeront à la session selon leur niveau de QoS.

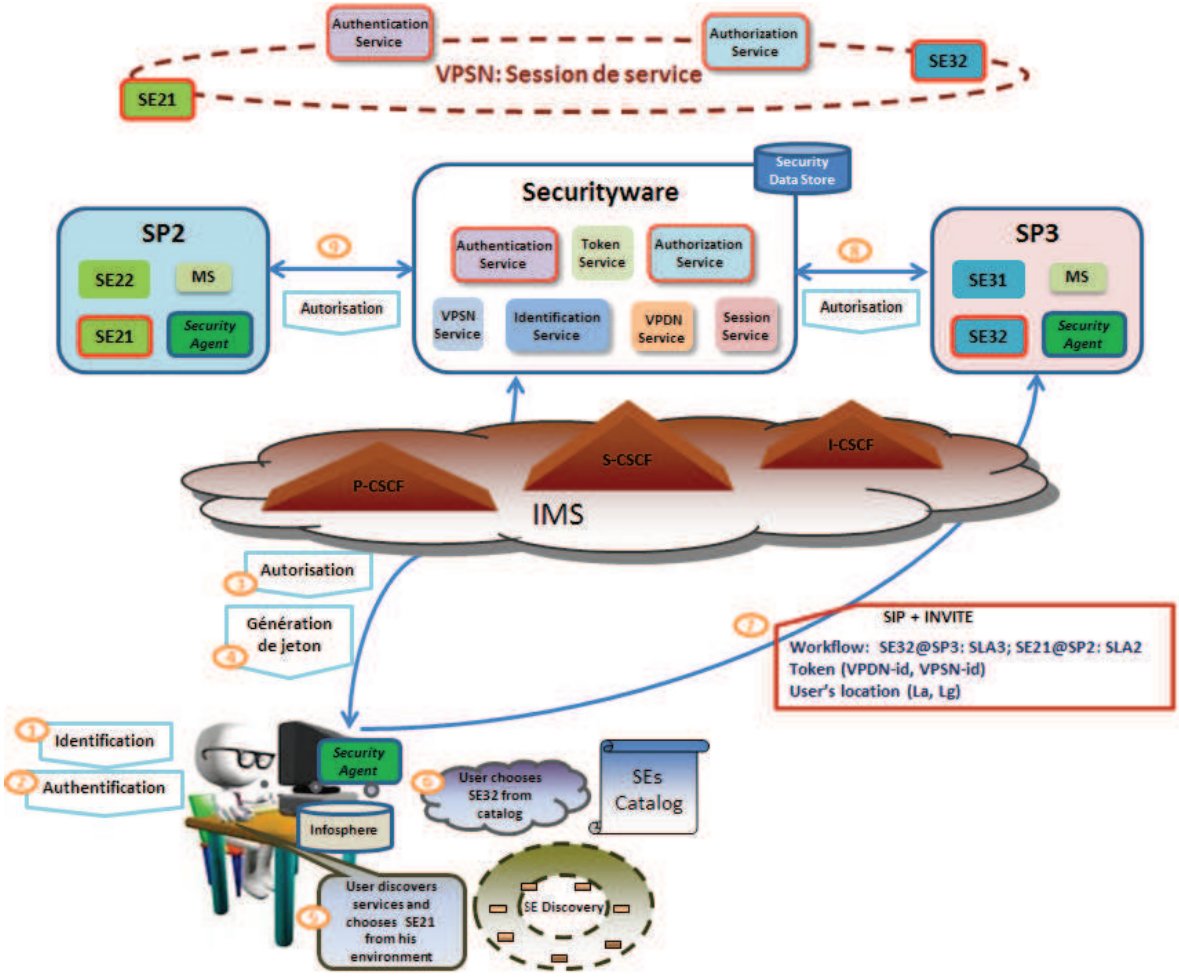



***Figure 39 :*** *Session de service.*

Pour ce faire, les informations de QoS et de sécurité sont encapsulées dans un jeton transféré via le protocole de signalisation SIP+ dans un message *SIP+ Invite* en sollicitant IMS (IP Multimedia Subsystem) pour accéder aux plates-formes de service. Cette méthode *SIP+ Invite* permet de créer la session de service de l'utilisateur (Figure 39). En fait, le composant de service réserve les ressources requises pour répondre à la requête de l'utilisateur. Chaque composant de service contient une file qui conserve les requêtes

acceptées des différents utilisateurs. Au cas où une réservation est annulée, un message *SIP+ Cancel* est utilisé.

Prenons le cas d'usage de la création de la session de service (VPSN) illustré dans la Figure 40.

L'établissement de la session de service est lancé par la plate-forme de services SP1 après l'autorisation et la génération du VPSN-ID. Par la suite, SP1 envoie le message SIP + INVITE qui contient toutes les informations concernant les composants de services sollicités qui vont éventuellement participer à la session « user-centric ». Elles sont introduites dans le « Body » du message SIP+ NVITE.

Le corps du message SIP+ INVITE contient donc principalement la logique des services, qui définit la liste des transactions des services et leur ordre d'exécution au cours de la session user-centric {SE21@domain2 ; SE31@domain3}. Il contient également le jeton crypté et chiffré qui contient les identifiants du VPSN et du VPDN. Le passage d'une plate-forme à l'autre est assuré par le routage sémantique qui possède l'adresse du composant de service qui suit sémantiquement selon la logique de service. A la réception de ce message, chaque Security Agent qui protège les plates-formes de services telles que SP2 et SP3, va communiquer l'autorisation des composants de services résidant sur celles-ci auprès de Securityware tout en vérifiant la validité du jeton.

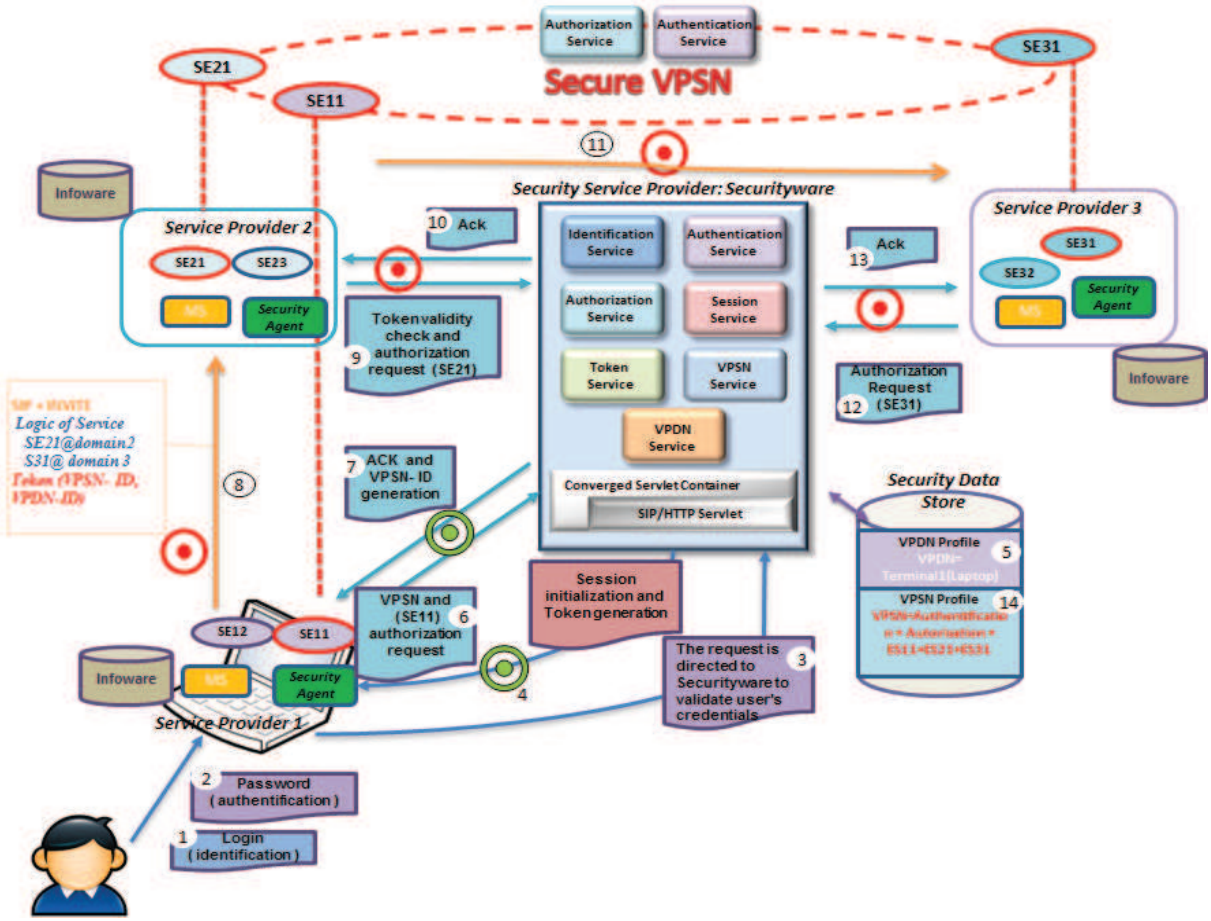



***Figure 40 :*** *Création de la session de service (VPSN).*

*2 Phase de traitement :*

Dans cette phase, nous définissons deux types de traitement. Le traitement basique est celui de la consommation des services composés par l'utilisateur dans sa session. Ces services peuvent être consommés en utilisant SOAP, HTTP ou Rest.

Le deuxième est le traitement de la continuité des services. Il consiste à gérer et contrôler les niveaux de QoS et de sécurité demandés.

Pour des raisons de sécurité, chaque variation au niveau du contexte de l'utilisateur (par exemple la mobilité de l'utilisateur qui s'agit du changement du terminal) ainsi que les changements liés à un composant de service (par exemple la disponibilité, une attaque malveillante, une faille de sécurité) doivent être surmontés par un ajout, une suppression ou un remplacement d'un ou plusieurs composants dans la session de service de l'utilisateur.

Afin de satisfaire les besoins des utilisateurs quand il a besoin de passer d'un terminal à un autre (mobilité de l'utilisateur). Le protocole SIP+ est en mesure de maintenir la continuité de la session et d'assurer la QoS de bout en bout ainsi que la continuité de la sécurité. Pour ce faire, un message *SIP+ REFER* informe le *Securityware* que l'utilisateur veut transférer sa session à un autre terminal. Ensuite, le *Securityware* envoie un message *SIP+ Invite* contenant le jeton de sécurité et la QoS demandé à savoir la fiabilité, la disponibilité, le délai et la capacité.

Pour des besoins de QoS, nous avons intégré dans chaque composant de service un agent de QoS qui contrôle la QoS courante devant satisfaire le contrat SLA. Un composant de service affecté devra être remplacé par un autre composant ubiquitaire équivalent en fonction et en QoS.

Prenons le cas d'usage suivant (Figure 41) pour illustrer et montrer comment SIP+ est capable de maintenir une session « sans couture », continue et sécurisée malgré les variations possibles des préférences et des emplacements de l'utilisateur, et ses différents types de mobilité, en particulier la mobilité de l'utilisateur.

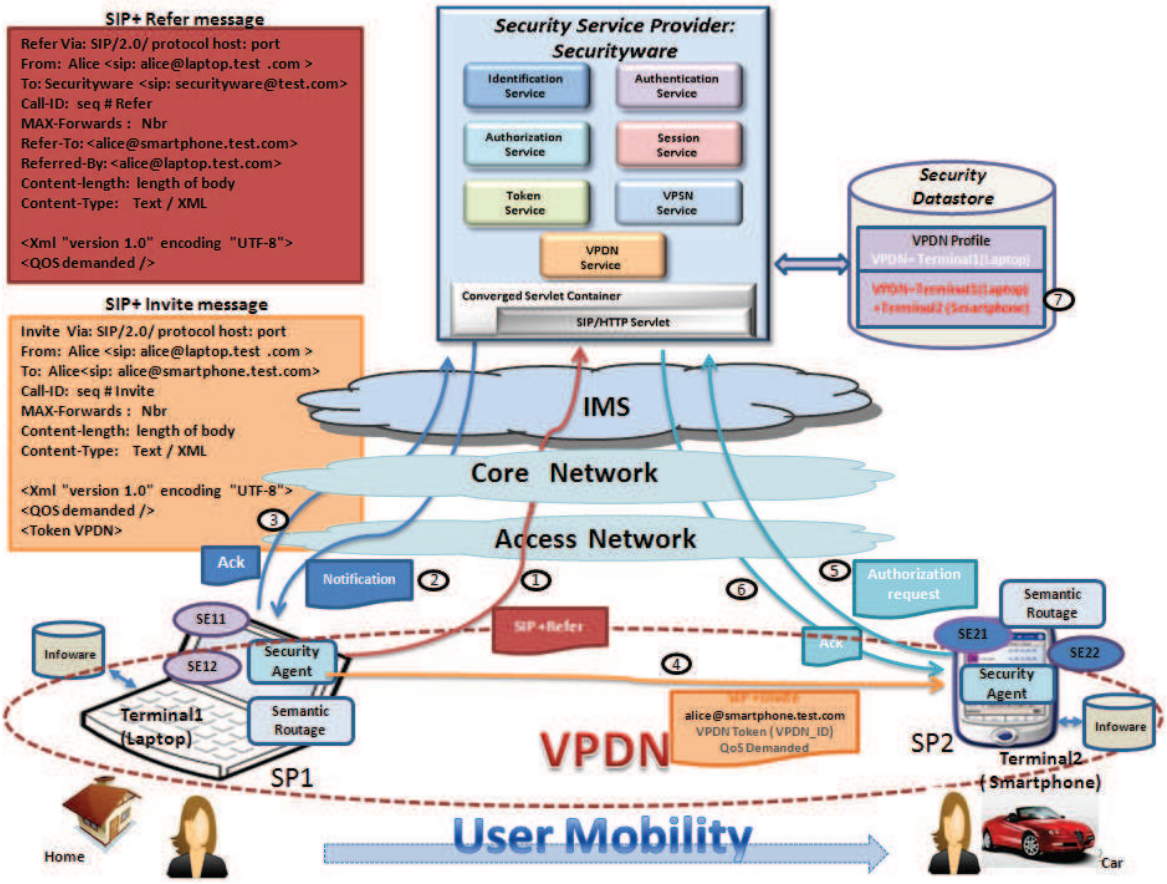



***Figure 41 :*** *Scénario Mobilité de l'utilisateur.*

Alice est dans sa maison, elle désire utiliser son ordinateur portable (laptop). Après la création du VPDN qui consiste à l'identification, l'authentification, l'autorisation et la génération du jeton, Alice est capable d'utiliser son « laptop » (terminal 1), ouvrir sa session et composer ses services après qu'elle soit autorisée. Ce terminal joue deux rôles: Le premier est celui d'une plate-forme de services (SP1) qui offre un ensemble de services protégée par un « Security Agent ». Nous considérons, dans ce scénario, le service d'affichage (SE11) qui est un service « terminal » et le service mail (SE12) qui est un service applicatif. Le second rôle est celui d'un composant de service de type équipement qui fait partie du VPDN.

Alice utilise son « laptop » avec son composant d'affichage (SE11). Elle peut également recevoir ses mails (SE12) à travers ce terminal. Dans ce cas, le VPDN d'Alice est formé seulement par son « laptop » :

$$VPDN\ (Alice) = \{Laptop\}.$$

Supposons qu'Alice va sortir de sa maison pour se rendre à un anniversaire de son amie en voiture. Elle désire basculer sa session de son « laptop » à son Smartphone en préservant la continuité et la sécurité de sa session. Pour ce faire, le Security Agent 1 (terminal1) envoie un message SIP+ Refer au Securityware qui vérifie si le terminal fait partie du PAN d'Alice. Si la réponse est positive, alors le Securityware établit un canal sécurisé entre les deux terminaux. Ensuite le Security Agent1 envoie directement une requête SIP+ Invite au terminal2 contenant le jeton de session (il contient l'identifiant du VPDN). Le Security Agent2 intercepte la requête et vérifie auprès de Securityware si le terminal2

(Smartphone) fait partie du VPDN et si Alice a les droits et est autorisé à composer ses services à partir du terminal2. De plus, le Securityware vérifie la validité du jeton et la session. Si la réponse est positive, un ack est envoyé par Securityware et en même temps la table du VPDN est mise à jour. Le VPDN d'Alice devient :

VPDN (Alice) = {Laptop; Smartphone}

Alice passe de son Laptop à son Smartphone tout en gardant sa session (Mobilité de l'utilisateur). Elle peut ainsi continuer la communication avec son amie tout au long de son trajet en voiture. Le changement du terminal implique l'invocation du composant (ES21) qui correspond au service d'affichage pour le Smartphone. En fonction de ses besoins, Alice change la composition de ses services pour construire une application répondant à une nouvelle logique de service. On peut considérer deux scénarios suite à la mobilité de l'utilisateur :

1er Scénario :

Le terminal1 (Laptop) est considéré comme une plate-forme de service qui offre les services (ES11, ES12). L'utilisateur changeant de terminal, il se peut qu'il ait encore besoin du service ES12 (c'est un service de localisation par exemple). Il a bien sûre besoin du service ES21 (service d'affichage). Le Policy Agent vérifie auprès de Securityware si l'utilisateur a le droit ou non par rapport à son rôle. Dans ce scénario, le terminal1 reste actif.

2ème Scénario :

Dans le cas où le terminal1 n'est plus sollicité, Alice n'a plus besoin du service SE12, ce dernier passe à l'état disponible (il n'est plus activé), il doit envoyer une notification à Securityware. Le Security agent vérifie seulement la validité du jeton.

3 Phase de terminaison :

Cette phase permet la libération des services et des ressources impliquées dans la session de l'utilisateur. A cet effet, le message *SIP+ Bye* est utilisé afin de libérer les ressources et terminer la session de service.

### 8.2.4 Les attaques de SIP+

Vu que le protocole SIP+ est déployé dans un environnement dynamique, mobile, hétérogène et trans-organisationnel, il est exposé à des menaces et des attaques potentielles héritées de SIP et XML. En plus, la nature de ce protocole, lui-même, le rend vulnérable à quelques attaques comme l'exploitation malveillante de ses messages. Alors, nous identifions, dans cette section, les attaques majeures dans SIP+.

- *Inondation des messages SIP+ Invite :*
  Il s'agit d'envoyer un grand nombre des messages SIP+ Invite à une plate-forme de service cible. Par conséquent, celle-ci sera inondée et incapable de recevoir et servir des requêtes légitimes.

- *Session Teardown :*
  Elle se produit lorsqu'un attaquant envoie un message « Cancel » ou « Bye » falsifié à une plate-forme de service impliquée dans la session de service de l'utilisateur.

Ceci mène à terminer brusquement et illégitimement la session. Cette attaque est due au niveau faible d'authentification pour les messages « Cancel » et « Bye ».

- *L'attaque par Rejeu (Replay):*
  C'est la retransmission d'un message authentique d'une manière frauduleuse. Elle se produit lorsqu'un attaquant est capable d'intercepter une requête SIP+ Invite et ainsi récupérer un jeton de session valide. Ceci implique l'usurpation d'identité de l'utilisateur authentifié. Par conséquent, si l'attaquant rejoue le message, il obtiendra et bénéficiera des mêmes droits que l'utilisateur.

- *Vol d'identité :*
  Elle se produit lorsque des informations sont volées et utilisées pour avoir l'accès à des services par tromperie. Ceci permet la consommation des services sans devoir payer le fournisseur.

- *Détournement (hijacking) de session :*
  Il s'agit de l'exploitation des mécanismes de contrôle de session, tel que le jeton de session, afin de profiter d'un accès non autorisé aux services.

- *Falsification des messages :*
  Elle se produit lorsqu'un attaquant modifie des données légitimes passant par des plates-formes de service. Chaque message (données) envoyé à partir d'une plate-forme de service peut être manipulé. Cette attaque est réussie si les données falsifiées atteignent la destination finale.

- *Injection XML :*
  C'est une technique malveillante utilisée pour manipuler la logique (structure) d'un document XML.

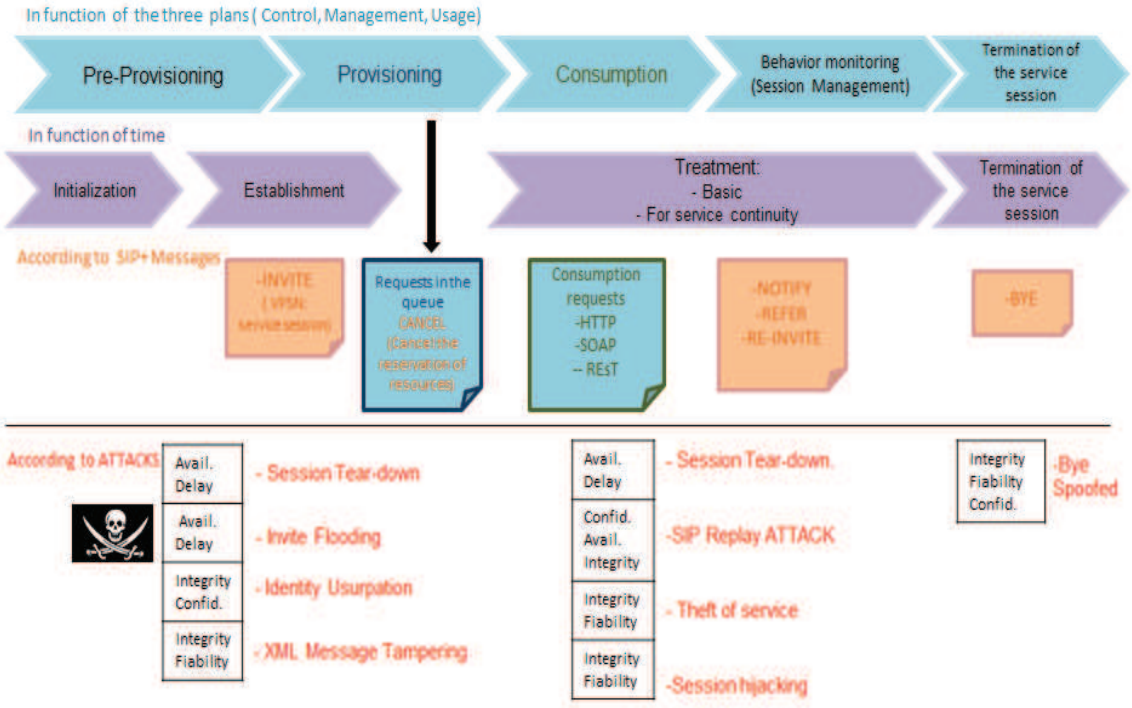


***Figure 42 :*** *Les attaques durant les différentes phases de la session.*

### 8.2.5 SIP+ sécurisé

Dans cette section, nous décrivons la manière avec laquelle cette session mobile et unique établie par SIP+ peut être protégée des attaques et vulnérabilités majeures.

Un des principaux messages SIP+ participant à l'établissement de la session de l'utilisateur est le SIP+ Invite. Il est composé de deux parties : l'entête du message qui contient le champ de la route indiquant la logique de service et l'ordre d'exécution des composants de services dans la session « user-centric » ; et le corps du message qui contient la description de QoS demandé de chaque composant de service ainsi que les informations liées à la sécurité (jeton).

Le corps du message est basé sur le langage XML. C'est pourquoi les messages SIP+ héritent particulièrement les attaques ciblant le langage XML. Afin de prévenir ces attaques, nous proposons tout d'abord de signer le corps du message.

Pour ce faire, nous utilisons *XML Digital Signature* qui permet de signer un document XML entier ou uniquement une partie de ce document. Ceci représente un avantage capital. En effet, dans le cas où une seule partie du message est signée, le reste de ce message peut être modifié. Grâce à ceci, deux participants peuvent ainsi signer différentes parties du même message. Chacun peut signer sa partie et assumer par la suite sa responsabilité. Ceci est très utile dans notre cas qui implique la composition de service dans un environnement trans-organisationnel.

La structure de la signature est composée des quatre éléments suivants : *SignedInfo*, *SignatureValue*, *KeyInfo* et *Object*.

- *SignedInfo*: c'est un élément obligatoire qui spécifie ce qui est signé et quel algorithme est utilisé. Premièrement, il contient l'élément *CanonicalizationMethod* qui indique la forme utilisée normale ou canonique pour obtenir la même représentation du message. Deuxièmement, l'élément *SignatureMethod* définit l'algorithme utilisé pour transformer l'élément canonique *SignedInfo* à *SignedValue*. Enfin, l'élément *Reference* identifie la ressource en utilisant l'URI (Unique Resource Identifier) et inclue les éléments *Transforms*, *Digest Methods* et *Digest Value*. L'élément *Transforms* définit la liste des transformations appliquées à la ressource avant le calcul du digest. L'élément *DigestMethod* identifie l'algorithme utilisé pour calculer le digest. Le dernier élément dans l'élément *Reference* est le *DigestValue* qui contient une valeur digest de la ressource.
- *SignatureValue*: c'est un élément obligatoire qui contient la valeur digest de l'élément *SignedInfo* crypté.
- *KeyInfo*: c'est un élément optionnel qui est utilisé pour permettre aux destinataires de recevoir les clés nécessaires pour le calcul et la validation de la signature.
- *Object*: c'est un élément optionnel qui contient les données signées dans le cas d'une signature enveloppée.

La construction de la signature, respectant la signature *XML Digital Signature*, nécessite un certain nombre d'étapes présentées par les deux algorithmes suivants :

**Du coté de l'émetteur : Signature du document avec XML Digital Signature**

```
1 Begin
2   Implémenter les changements nécessaires à la ressource;
3   Calculer la valeur du DigestValue codé;
4   Créer l'élément Reference;
5   Créer l'élément SignedInfo en se basant sur les éléments
    SignatureMethod, CanonicalizationMethod et Reference;
6   Créer l'élément canonique SignedInfo;
7   Calculer l'élément SignatureValue;
8   Ajouter les informations relatives à la clé;
9   Encapsuler l'élément Signature;
10 End
```

**Du coté du récepteur : Validation de la signature avec XML Digital Signature**

```
1 Begin
2   Obtenir les informations de vérification de la clé ;
3   Appliquer l'algorithme CanonicalizationMethod sur l'élément SignedInfo;
4   Vérifier la valeur de SignatureValue en utilisant la forme canonique de la
    SignatureMethod;
5   Pour chaque élément Reference faire
6     Obtenir les données chiffrées;
7     Déchiffrer les données en utilisant l'algorithme DigestMethod associé;
8     Comparer la valeur calculée avec la valeur non chiffrée de l'élément DigestValue ;
9 End
```

***Figure 43 :*** *Algorithmes de la signature XML Digital Signature.*

Le premier algorithme décrit les étapes qui sont effectuées au niveau de l'émetteur du message qui crée la signature. Pour le destinataire, il doit vérifier que le message est valide, ce qui veut dire que le message n'est pas altéré. La validation de la signature se fait en deux étapes : d'abord, la signature calculée en se basant sur le contenu du *SignedInfo* doit être validée ; puis, chaque digest des objets signés doit être validé.

Ainsi, la signature XML assure l'intégrité du document qui contient le jeton et les informations de QoS, garantit la non-répudiation et protège l'identité de l'utilisateur.

Après la sécurisation du corps du message basé sur le langage XML, nous devons maintenant sécuriser le message SIP+ en entier et définir des mécanismes pour prévenir autres attaques potentielles. Cependant, nous sommes confrontés à quelques contraintes. D'abord, l'entête du message SIP+ ne peut pas être crypté car elle doit être lisible par les nœuds intermédiaires pour des raisons de traitement et de routage. De plus, quelques champs (via, route…) dans l'entête sont éventuellement modifiés durant la transmission donc ils ne peuvent pas être signés. En outre, comme nous adoptons l'authentification unique dans notre session de service en utilisant le jeton de sécurité, ce dernier doit durer toute la session de l'utilisateur, ainsi, la technique d'horodatage (time-stamping), par exemple, ne peut pas être utilisée pour se protéger contre l'attaque par rejeu.

Pour surmonter ces contraintes, nous proposons quelques mécanismes adéquats et efficaces de sécurité. Comme contre-mesure de l'attaque par rejeu, nous proposons de signer les champs Call-ID (un identifiant global et unique de la session) et Cseq (numéro de séquence utilisé pour faire correspondre les requêtes et les réponses) dans l'entête. Par conséquent, si un attaquant envoie le même message, le destinataire rejettera ce message puisqu'il est dupliqué. Et si l'attaquant incrémente le numéro Cseq, la signature sera invalide.

Pour l'attaque Session Teardown, nous signalons que la principale cause est le manque d'authentification des messages Cancel et Bye. Alors, nous proposons de signer ces messages afin de prouver l'authenticité et l'identité de l'utilisateur.

Pour résumer, la signature du message SIP+ (Figure 44) assure la protection contre l'attaque par rejeu, l'attaque Session Teardown et l'attaque de falsification des messages. Ainsi, nous proposons de signer tout le message SIP+ à l'exception des champs qui sont modifiables au cours de la route tel que le champ Route et le champ Via.


***Figure 44 :*** *Signature du message SIP+.*

❖ **Scénario**

Nous présentons dans cette section un scénario qui explique comment protéger la session de l'utilisateur établie par SIP+ contre les attaques et les vulnérabilités.

Ce scénario, comme le montre la Figure 45, se déroule dans un contexte trans-organisationnel ayant plusieurs plates-formes de service. La création de la session de service « user-centric » est partagée entre les différentes plates-formes impliquées.

Décrivons maintenant notre scénario :

D'abord, Bob doit être identifié et authentifié. Après la saisie de son login et son mot de passe, le Security Agent de son terminal envoie une requête au Securityware qui se charge de la vérification et la validation des accréditations de l'utilisateur. Suite à une authentification réussie, le Securityware génère un jeton de sécurité pour Bob contenant des informations concernant sa session.

A cette étape, Bob peut créer sa session de service via son terminal. Le Security Agent envoie un message Invite SIP+ à la première plate-forme de service. Ce message contient le jeton de sécurité et les mesures exigées de QoS des différents services demandés par Bob. Il indique aussi les différentes plates-formes de service impliquées et l'ordre d'exécution des services grâce au champ « Route » dans l'entête. Dans notre cas, le service personnalisé de Bob nécessite l'élément de service SE11 de la plate-forme de service 1 (SP1) et l'élément de service SE21 de la plate-forme de service 2 (SP2).

Lorsque SP1 reçoit la requête Invite SIP+, son Security Agent vérifie la validité du jeton et les droits de Bob auprès du Securityware. Puis, la requête Invite SIP+ est transférée à SP2 qui vérifie à son tour le jeton et les droits de Bob.

Dans ce scénario, nous identifions les attaques majeures qui peuvent se produire et nous appliquons nos contremesures de sécurité proposées.

En premier lieu, les informations encapsulées dans le jeton et échangées entre le Security Agent et le Securityware durant la phase d'authentification peuvent être usurpées par un attaquant. Pour faire face à une telle usurpation d'identité, nous établissons un canal sécurisé en utilisant le mode TLS (Transport Layer Security) [51] entre le Security Agent intégré dans chaque plate-forme de service (y inclus le terminal) et le Securityware.

En deuxième lieu, la création de session de service est partagée entre des plates-formes de service multiples. Alors, le message Invite SIP+, contenant le jeton et les valeurs de QoS, circulera entre les différentes plates-formes de service impliquées dans la session. Ceci mène à exposer ce message à différents types d'attaques. Par exemple, le jeton peut être intercepté par un attaquant sur son chemin. Par conséquent, le jeton sera transmis dans un message similaire ce qui engendre l'attaque par rejeu.

Une autre attaque où le jeton peut être réutilisé est de construire un message falsifié pour terminer ou annuler la session. Elle s'appelle « session teardown attack ».

Un attaquant peut aussi modifier le corps du message en XML (le jeton ou les valeurs de QoS). Cette attaque peut être appelée « message tampering » ou « XML injection ».

Pour se protéger contre ces attaques, le Securityware, en générant le jeton, applique la signature XML sur le corps du message Invite SIP+ pour assurer l'authenticité, l'intégrité et la non-répudiation des informations contenues dans le jeton et de QoS. Ensuite, le Securityware signe le message Invite SIP+ à l'exception de deux champs : Via et Route qui contient la logique de service.

Comme une deuxième contremesure, un pare-feu est déployé à chaque plate-forme de service pour empêcher les attaques de déni de service comme, par exemple, l'inondation des messages Invite SIP+.

A la réception d'un message signé, le Security Agent de chaque plate-forme de service doit vérifier la validité du certificat de l'utilisateur et examiner sa signature. Puis, il vérifie la validité du jeton auprès du Securityware.

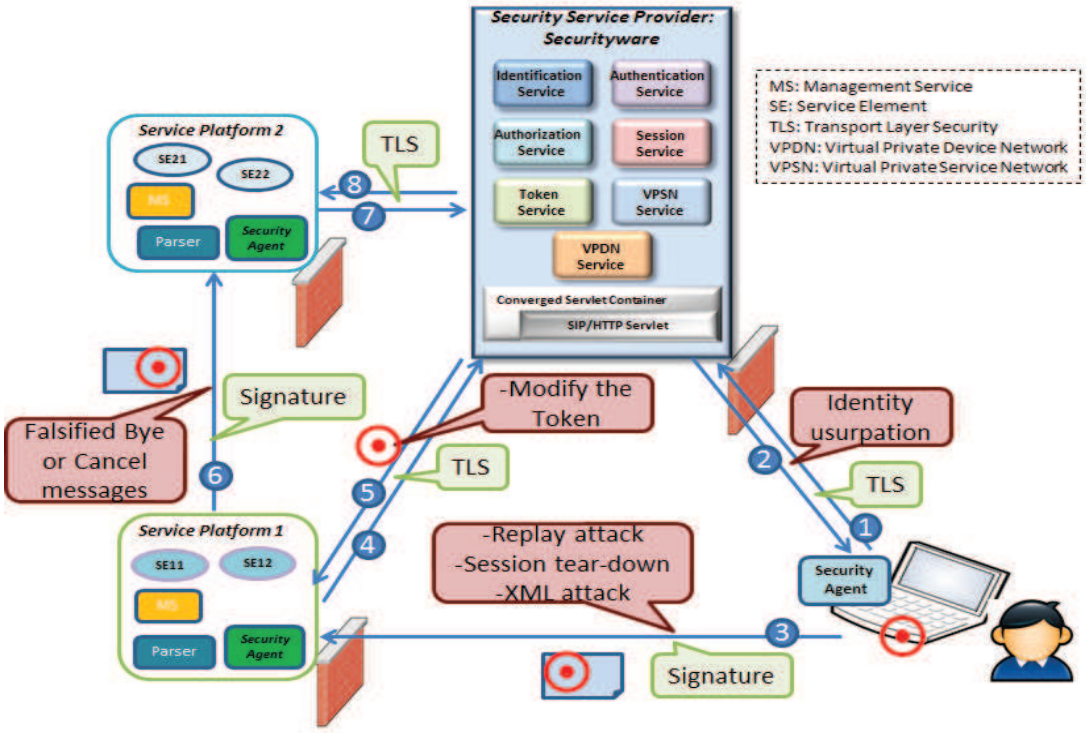



***Figure 45 :*** *Scénario.*

## 8.3. Conclusion

Dans ce chapitre, nous avons consigné nos propositions pour satisfaire les besoins de continuité de service et plus particulièrement la continuité de sécurité et QoS durant la session centrée utilisateur dans un contexte mobile et hétérogène. Or, cette session doit être unique sécurisée sans couture et sans coupure.

En s'appuyant sur les travaux existants de notre groupe de recherche qui traitent la dimension protocolaire dans le cadre du projet UBIS et le fait que notre architecture repose sur le cœur IMS, nous avons proposé une extension de la portée SIP existant, nommée SIP+. Ce dernier permet d'assurer une interaction entre les différents acteurs en échangeant les informations de QoS et plus particulièrement les informations de sécurité (jeton), et cela pour faciliter la création et la modification d'une session de service unique continue et sécurisée. Une étude approfondie a été faite sur la problématique de la session unique sans couture et sans coupure, qui nous a permis de proposer les jetons basés sur SIP+ pour assurer d'une part la continuité de la sécurité, et d'autre part, l'authentification unique basé sur ces jetons pour tous les services sollicités afin d'assurer l'unicité et la propagation de la session sans avoir besoin d'une réauthentification. De plus, pour assurer la délivrance de service de bout en bout sécurisée dans un contexte trans-organisationel, nous nous sommes concentrés sur la problématique de sécurisation de SIP+ ainsi que le jeton duran l'échange et lors des différentes phases de la session. Nous avons tout d'abord identifié les attaques potentielles pour aboutir à la proposition du SIP+ sécurisé.

*Partie 3*

# Valorisation et Faisabilité

# 9 Plates-formes et scénarii

## 9.1. Plate-forme d'expérimentation

Pour prouver la faisabilité de l'ensemble de nos propositions, nous nous sommes basés sur la plate-forme du démonstrateur du projet UBIS (comme le montre la Figure 46) dans lequel se situe ce travail. Elle est structurée selon les différents niveaux de visibilité : utilisateur, terminal, transport, contrôle et service. Elle traite les aspects de signalisation, de gestion, de sécurité et de maintenance des informations dans les bases de connaissances.

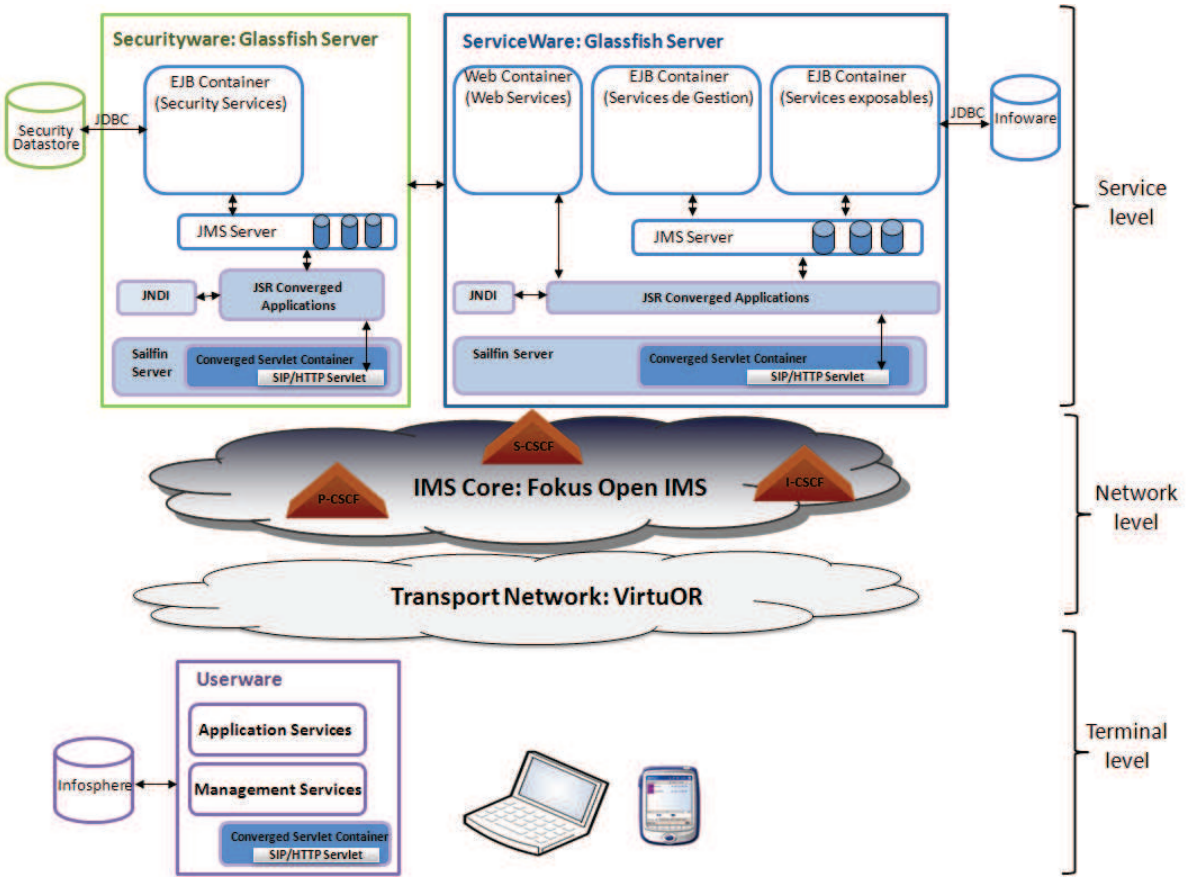



***Figure 46 :*** *Architecture du démonstrateur sur les différents niveaux de visibilité.*

Au niveau Utilisateur, les abonnements et les préférences de l'utilisateur en termes de personnalisation de service et de mobilité sont enregistrés dans les bases de connaissances Infosphère et Infoware.

Au niveau Terminal, l'architecture « Userware » fournit les fonctionnalités et les capacités pour permettre à l'utilisateur de personnaliser ses services en lui offrant un accès orienté usage qui est transparent. Elle est intégrée dans chaque terminal et contient les services applicatifs et les services de management (ex. Security Agent).

Au niveau Réseau du transport, nous utilisons la solution VIRTUOR [52] qui permet d'avoir plusieurs entités virtuelles de différents types à savoir les routeurs IPv4, IPv6, point d'accès virtuel, serveur SIP, etc. Cette solution offre la possibilité de mettre en place plusieurs réseaux virtuels, spécifiques aux applications supportées, qui sont vus de la part des utilisateurs comme des réseaux physiques distincts.

Au niveau de la couche de contrôle, nous nous appuyons sur OpenIMS de Fokus [53] pour mettre en place l'architecture IMS qui permet de contrôler les sessions actives (établissement, modification et libération) en se basant sur le protocole de signalisation SIP.

Au niveau Service, nous avons le « Serviceware » qui représente une architecture de service hébergeant différents types de service à savoir des services applicatifs et des services de management. Pour mettre en place l'architecture du « Serviceware », nous avons utilisé un serveur d'application GlassFish V3 [54] pour développer et déployer les services UBIS qui reposent sur l'architecture SOA. Nous avons utilisé le langage Java pour développer les composants de service de type gestion et applicatif comme des EJBs indépendants. Il faut mentionner que EJB3.0 permet le développement des composants de services autonomes avec un couplage lâche. Ces composants sont déployés dans le serveur Glassfish certifié JEE6, qui supporte différentes APIs comme JMS, JNDI et JDBC et SIP servlet. Java Message Service (JMS) est une API qui permet d'envoyer et de recevoir des messages de manière asynchrone entre applications ou composants Java. Un client peut également recevoir des messages de façon synchrone dans les communications P2P. Java Naming and Directory Interface (JNDI) est une API Java qui permet la connexion à des annuaires, notamment des annuaires LDAP. Java DataBase Connectivity (JDBC) est une API qui définit comment le client pourra accéder à la base de données. Elle fournit des méthodes pour lancer des requêtes et des mises à jour pour les données dans la base.

Au niveau des Informations, nous avons une base de connaissance Infoware [55] du côté fournisseur et Infosphere [56] du côté utilisateur/terminal.

Au niveau de la sécurité, une plate-forme de services spécifique « Securityware » qui propose un ensemble de services de sécurité tels que l'identification, l'authentification, l'autorisation, etc. Ces services sont développés en JAVA (EJB3.0) et déployés dans un serveur Glassfish. Notre Securityware est une extension de la solution OpenSSO [57] de SUN/Oracle (un des partenaires du projet UBIS et qui est devenu actuellement OpenAm).

De plus, un Security Agent est déployé dans chaque terminal et plate-forme de services pour protéger l'accès aux services. Le SecurityDataStore est un annuaire LDAP (Lightweight Directory Access Protocol Directory) (OpenDS) qui contient toutes les informations de l'utilisateur et qui est relié au Securityware.

Pour les besoins de signalisation, le serveur Sailfin [58] s'ajoute au serveur d'application Glassfish comme serveur de signalisation. Les requêtes du protocole SIP+ sont créées avec le client SIPP [59] pour créer, modifier et terminer les sessions de services personnalisés, et les requêtes HTTP sont utilisées pour la consommation des services. Nous avons une interface convergée SIP/HTTP «JSR Converged Applications» qui représente un ensemble

de Sip Servlet et Http Servlet qui sont gérés par le même conteneur pour assurer l'interfaçage des différents composants EJB ou Web Services.

## 9.2. Scénario applicatif

Nous décrivons, dans cette section, un cas d'usage qui met en valeur la gestion de la sécurité d'une session « user-centric » dans un contexte NGN/NGS et trans-organisationnel.

Dans ce scénario, nous supposons qu'un utilisateur Bob désire acheter une maison à Nice. Afin de trouver une offre qui convient à son besoin, il choisit à partir de son catalogue les deux services exposables suivants : le service *SquareHabitat*, fourni par *Crédit Agricole*, qui lui permet de rechercher un bien correspondant à ses critères de choix, et le service *Maps*, fourni par *Google*, qui lui permet de visualiser la localisation des biens trouvés sur la carte.

Le scénario se déroule comme l'indique la figure ci-dessous.

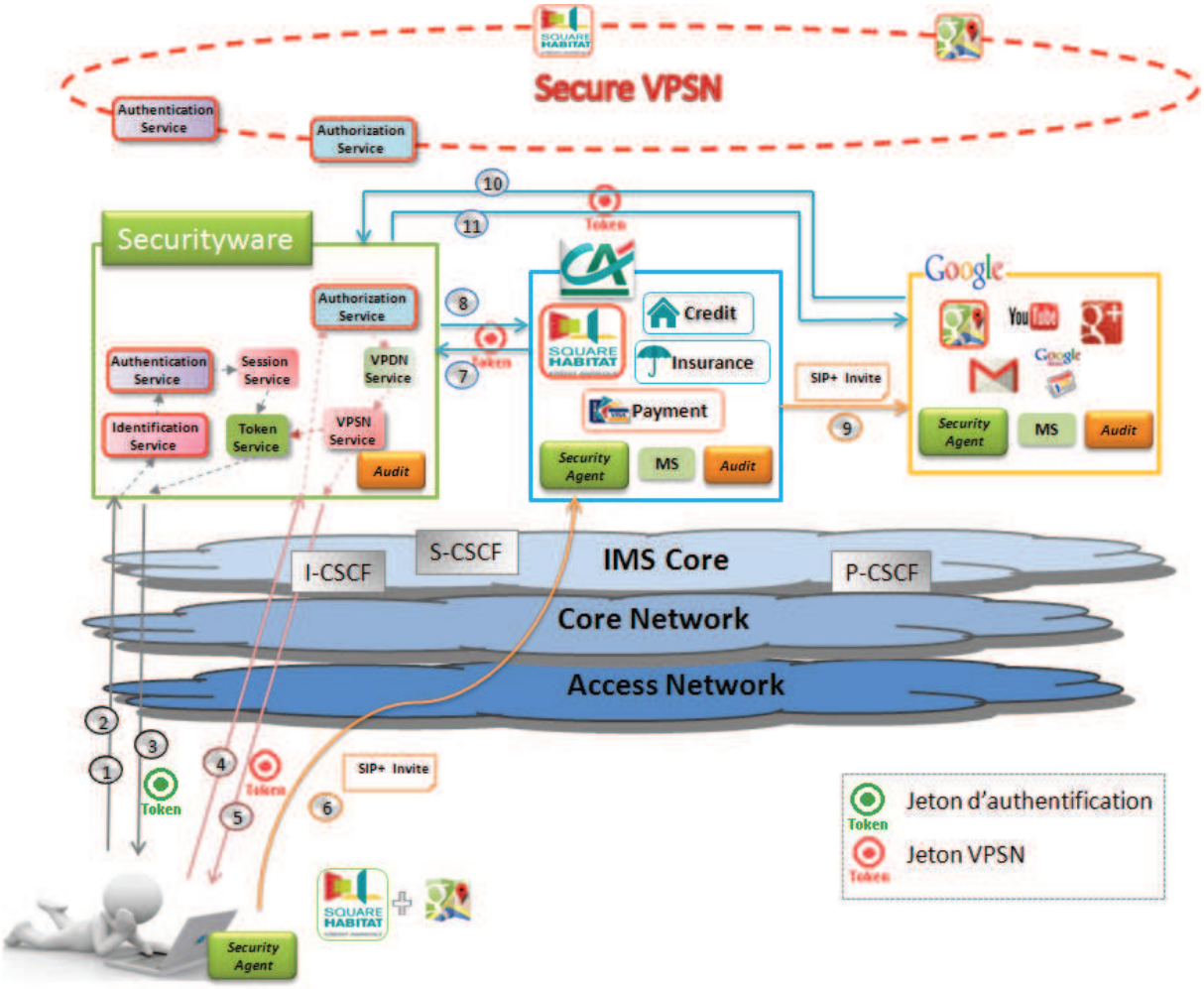



***Figure 47 :*** *Scénario applicatif – Création du VPSN.*

Bob est connecté à partir de son PC chez lui. Etant un client UBIS, un Security Agent est déployé au niveau de tous ses terminaux y inclus son PC.

Dans un premier temps, Bob s'identifie et s'authentifie auprès de Securityware. Un jeton d'authentification est alors créé.

Puis, afin de pouvoir consommer les services choisis à partir de son terminal, une demande d'autorisation est envoyée par le Security Agent au Securityware. Suite à cette autorisation, un jeton VPSN est créé et envoyé au Security Agent.

Maintenant, le Security Agent envoie un message SIP+ Invite contenant le jeton afin de solliciter les deux services demandés par Bob.

Nous mettons l'accent, ci-dessous (Figure 48), sur notre proposition concernant l'attribution de l'autorisation par rapport au premier service (*SquareHabitat*). Nous notons que ce service est décomposable en quatre services élémentaires : *Get, Set, Find et Location.* Le message SIP+ Invite contenant le workflow des services demandés et le jeton est intercepté par le Security Agent au niveau de l'EJB Container dans lequel se trouve le service *SquareHabitat*. Une demande d'autorisation est alors envoyée au Securityware afin de vérifier les droits de l'utilisateur en se basant sur le jeton déjà communiqué. Si la réponse est positive, le service est ajouté au VPSN de Bob. C'est justement ce VPSN qui maintient et supporte les droits d'accès quelque soit l'emplacement physique des composants de service autorisés. Avec OpenSSO, l'autorisation est faite par serveur d'application, c'est-à-dire, sur l'ensemble des services déployés dans ce dernier et qui appartiennent au même domaine. Or, la session *user-centric* dans notre cas est composée de services qui sont déployés dans différents serveurs d'applications, c'est-à-dire, fournis par plusieurs plates-formes de service et appartenant à plusieurs domaines. C'est ainsi, que le service support de sécurité (VPSN) au niveau de la couche Service intervient pour que l'autorisation se fasse par service.

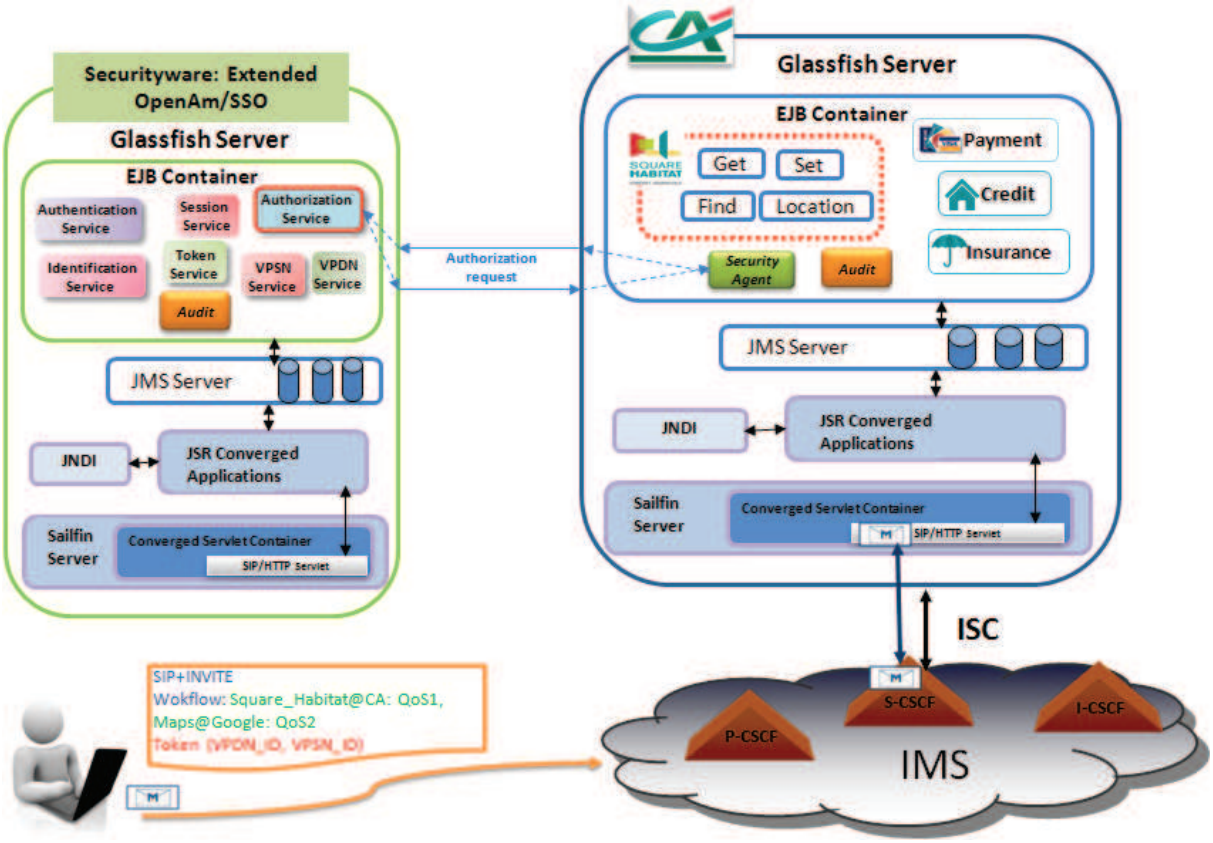



***Figure 48 :*** *Service d'autorisation.*

Le message SIP+ Invite est ensuite transféré à la deuxième plate-forme de service *Google*. La même opération est exécutée afin d'ajouter le service *Maps* au VPSN de Bob.

Montrons maintenant la mise en œuvre du VPDN. Supposons pour cela que Bob a un rendez-vous et il désire poursuivre la visualisation des offres des biens immobiliers au cours de son chemin. Il transfère alors sa session sur son Smartphone et demande un service *Taxi* fourni par *SFR* pour se rendre à son rendez-vous.

La figure ci-dessous illustre ce scénario de mobilité. Dans OpenSSO, le token de la session est stocké sous forme d'un « cookie » dans le browser de l'utilisateur. Lorsque l'utilisateur change de terminal, l'agent n'arrive plus à trouver ce cookie (il s'agit plus du même browser), d'où une réauthentification et une ré-autorisation sont demandées. Pour résoudre ce problème, le service support de sécurité (VPDN) au niveau de la couche terminal permettra d'identifier les terminaux auxquels l'utilisateur a l'autorisation. En cas de passage d'un terminal à un autre, le jeton de la session sera envoyé au nouveau terminal si l'utilisateur a déjà été autorisé à l'utiliser (il fait alors parti de son VPDN).

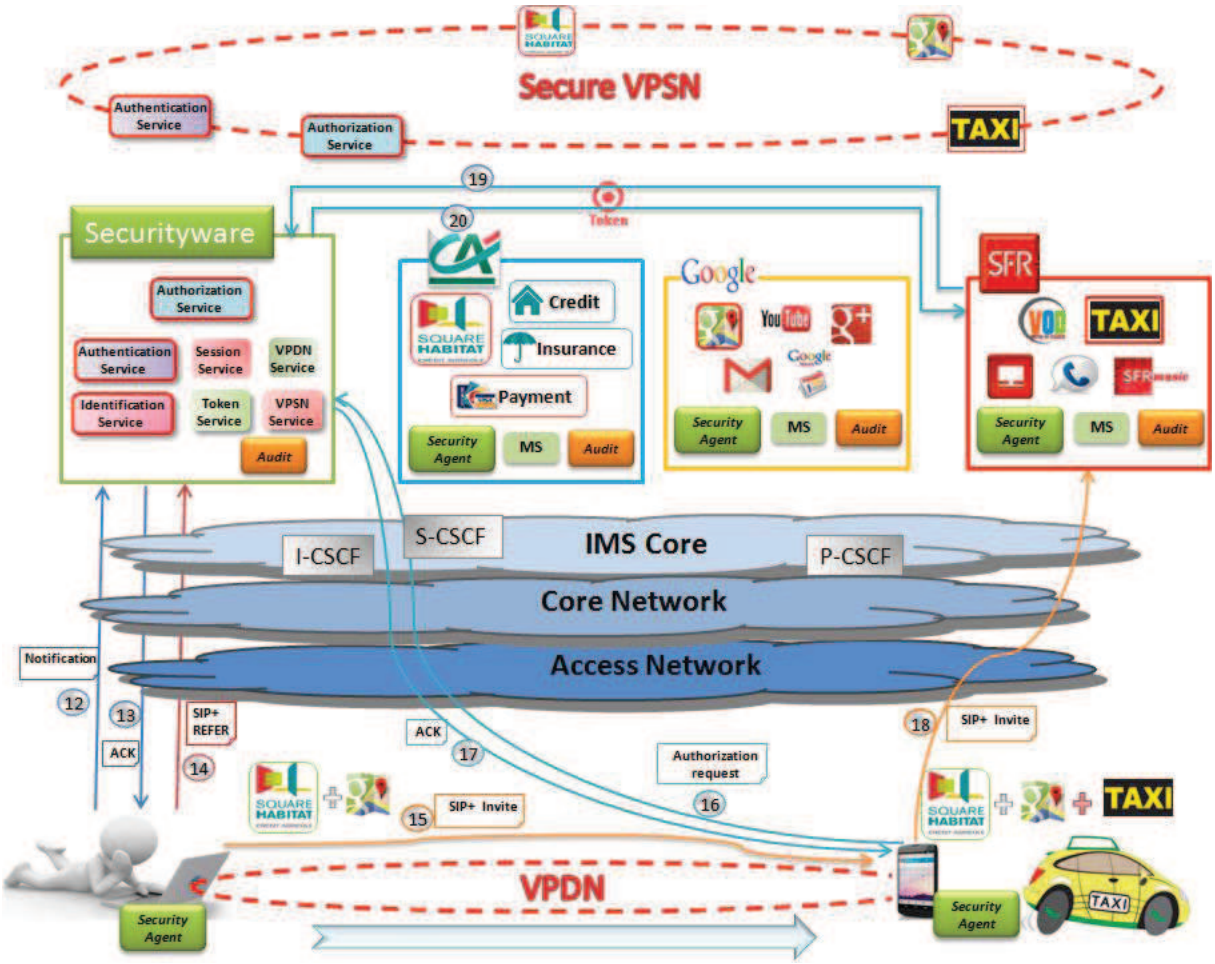



***Figure 49 :*** *Scénario applicatif – Mobilité de l'utilisateur.*

Nous traitons maintenant le cas où une attaque de déni de service survient à l'un des services, par exemple, le service *SquareHabitat_Get* (Figure 50).

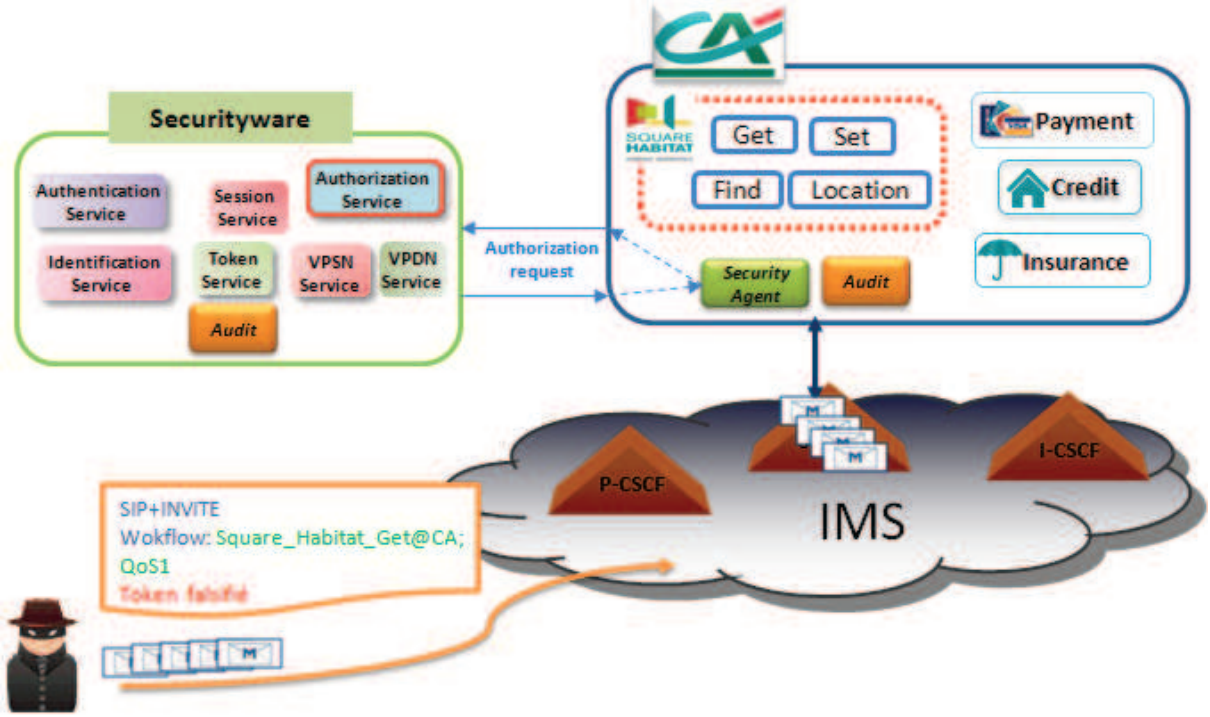



***Figure 50:*** *Attaque de déni de service (1).*

L'attaquant procède à l'inondation du serveur par des messages SIP+ Invite destinés au service cible. Pour chaque message, le Security Agent de la plate-forme *Crédit Agricole* va demander l'autorisation d'accès au service auprès de Securityware. Ce dernier détecte, grâce au service Token, que l'identité de l'expéditeur n'est pas valide. La demande d'autorisation est alors rejetée et un message de notification est envoyé au Security Agent. Par conséquent, ce dernier refuse la demande d'accès. (Voir Figure 51)

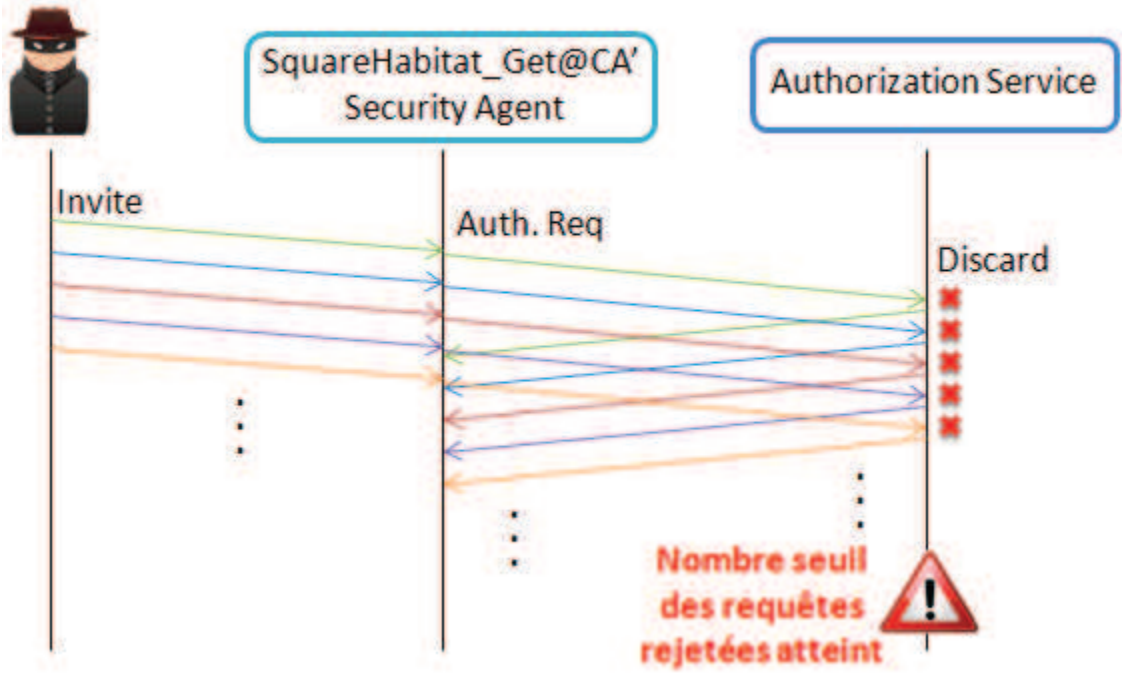



***Figure 51 :*** *Attaque de déni de service (2).*

Suite à un certain nombre de requêtes d'autorisation, l'agent de QoS situé au niveau du service d'autorisation détecte un nombre seuil de requêtes rejetées (IN Discard) mesuré par le service Monitoring. L'agent de QoS signale ainsi un « Out Contract » qui sera envoyé au service Audit. Ce dernier analyse la notification reçue et il trouve que les requêtes rejetées sont originaires du même expéditeur. Il déduit alors que c'est une attaque et il envoie une alerte au Security Agent de la plate-forme de service pour rejeter les messages reçus de la part de l'adresse de cet attaquant qui est ajoutée à la *black list*. (Voir Figure 52)

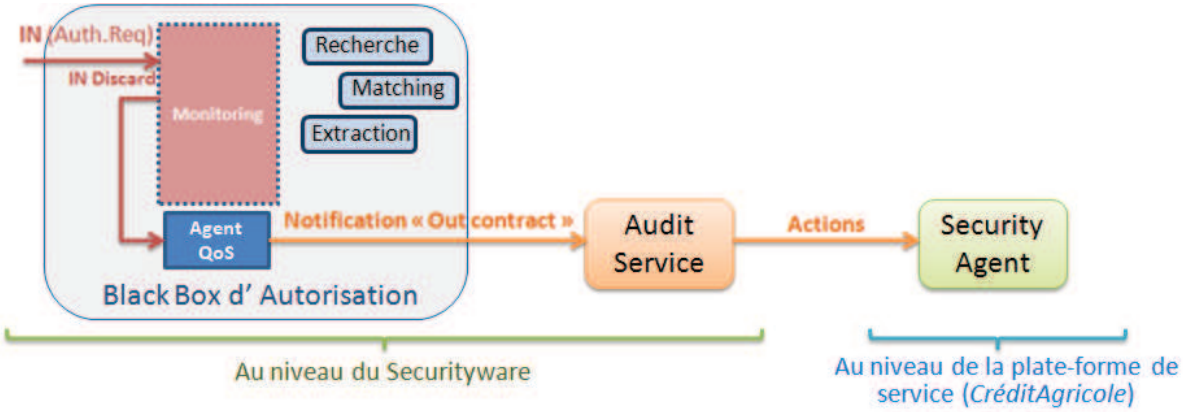



***Figure 52 :*** *Attaque de déni de service (3).*

# 10 Modèle informationnel

Dans le but d'avoir une représentation efficace du monde réel caractérisé par son hétérogénéité, nous devons nous appuyer sur une structure uniforme contenant les informations nécessaires pour une décision adéquate. Cette représentation permet ainsi de décrire les ressources ainsi que leurs comportements basés sur la QoS. Le modèle informationnel préconisé dans le projet UBIS est générique. Il permet de décrire n'importe quelle ressource selon le niveau de visibilité. Chaque ressource offre, de nature, un service, pour cela, on la dénomme « SE Component ». Le modèle informationnel représente une base de connaissance complète et seule pour chaque ES contenant les informations pertinentes et synthétiques afin de prendre les bonnes décisions aux bons endroits et aux bons moments. Il englobe ses informations (l'adresse, les contraintes, la table de routage, les valeurs de qualité de service demandées par chaque utilisateur…) et toutes les opérations nécessaires pour répondre aux requêtes sur les trois interfaces. Le composant aura, à un instant T, un des quatre états suivants : Indisponible, Disponible, Activable ou Activé.

Les classes de ce modèle nous permet de bien montrer comment les composants de services sont interopérables entre eux et respectent le contrat de chaque utilisateur tout en assurant l'autogestion.

Pour chaque Composant de service, le modèle informationnel décrit son architecture ainsi que le service offert.

L'architecture est composée du « Software » qui assure la fourniture des services, et des éléments « Support » qui représentent les contraintes du composant. Cette classe désigne et enregistre les valeurs de conception de la QoS du composant de service.

Le « Software » est formé d'une classe « Management » qui gère les services offerts par le composant et qui contient les valeurs seuils de QoS ; d'une « Entity » qui présente les fonctions cœurs du composant et qui utilise, modifie et enregistre les valeurs de la QoS courantes ; d'une table « Connection » qui matérialise les relations avec les composants de même niveau de visibilité et qui permet le routage selon des tables de QoS tout en enregistrant les valeurs courantes de la QoS ; et des points d'accès au service « SAP » à travers lesquels les services du composant sont fournis ou demandés. La partie Service représente les interfaces qui permettent à l'élément de service de répondre aux demandes de services ou émettre des messages aux entités du monde extérieur. Selon la nature des opérations invoquées et du service rendu, l'interface invoquée peut être celle de l'usage, du contrôle ou de management.

L'interface d'usage assure l'interopérabilité avec d'autres éléments de service. Le plan de gestion avec la classe Management assurent l'autogestion d'un composant en fonction de la qualité de service. En effet, en comparant périodiquement les valeurs courantes et les valeurs de conception de QoS, nous pouvons gérer le SLA. Les valeurs seuils de QoS servent à l'autogestion du composant.

Nous présentons dans cette partie les modèles informationnels associés aux services de sécurité « Identification », «Authentification » et « Autorisation »

Dans la figure qui suit, nous représentons le service de sécurité de base « Identification » selon son modèle informationnel.

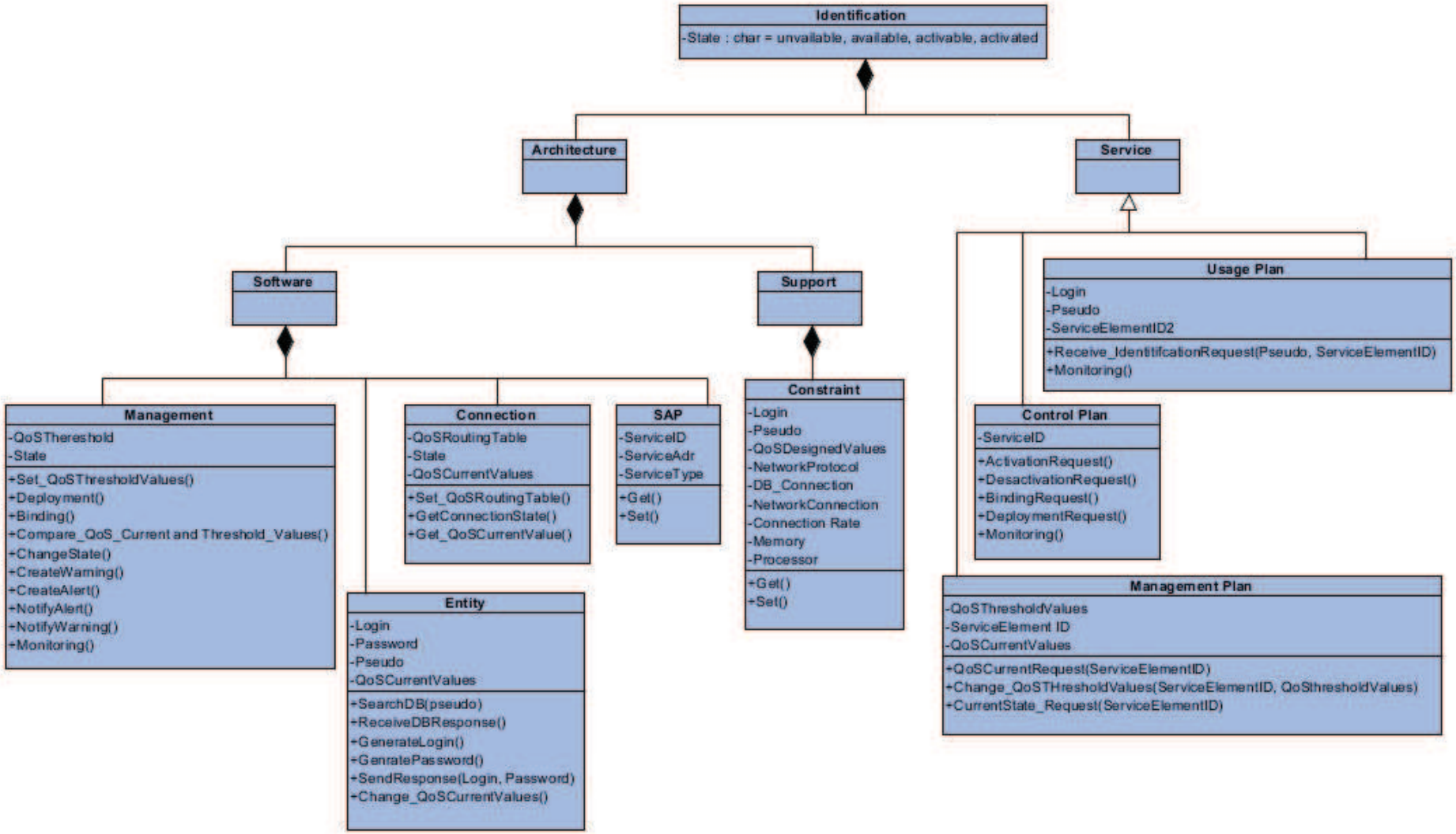



***Figure 53 :*** *Modèle informationnel du service d'identification.*

Le service Authentification est représenté selon son modèle informationnel dans le schéma suivant :

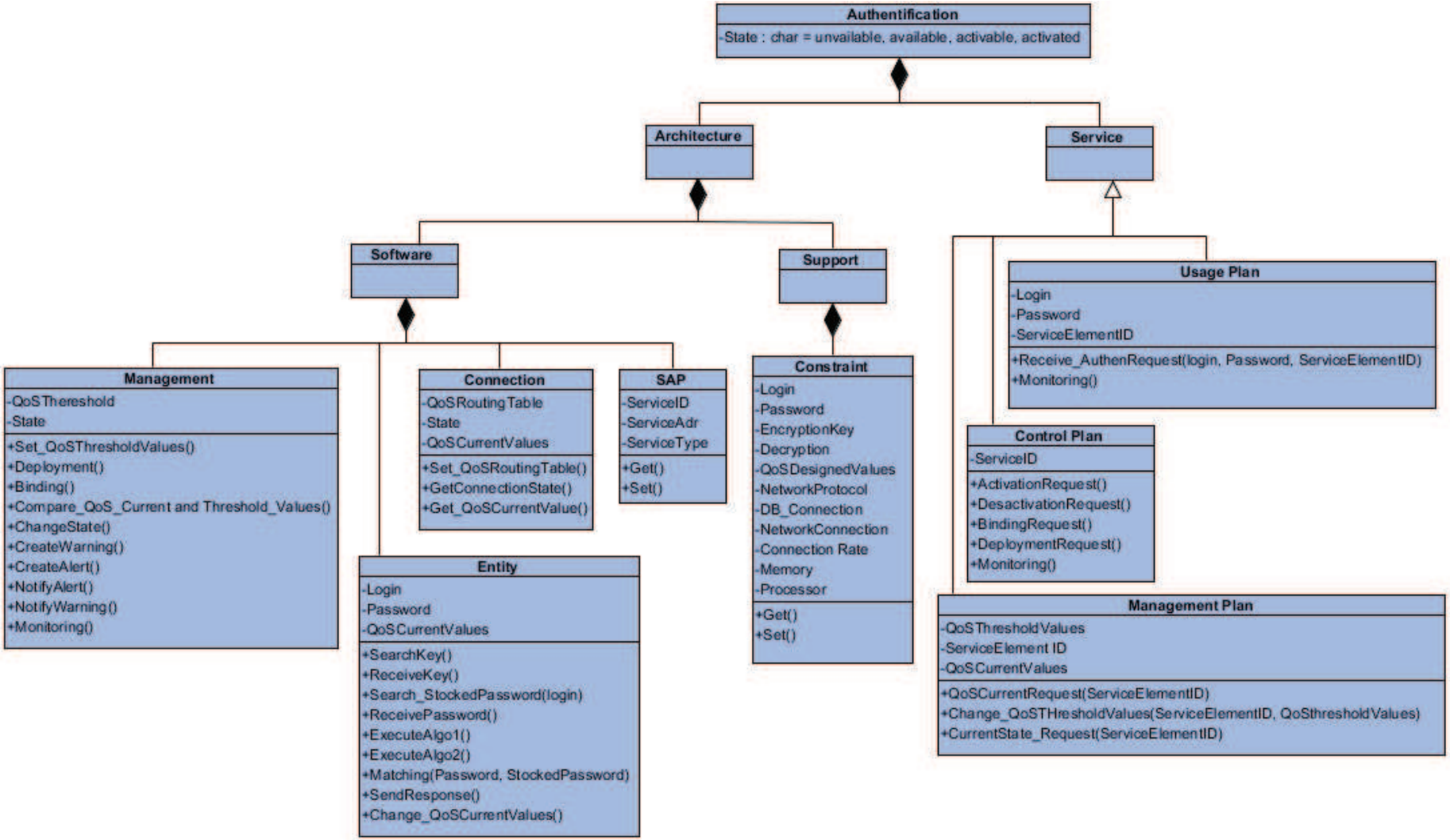



***Figure 54 :*** *Modèle informationnel du service d'authentification.*

Le service Autorisation est représenté selon son modèle informationnel dans le schéma suivant :

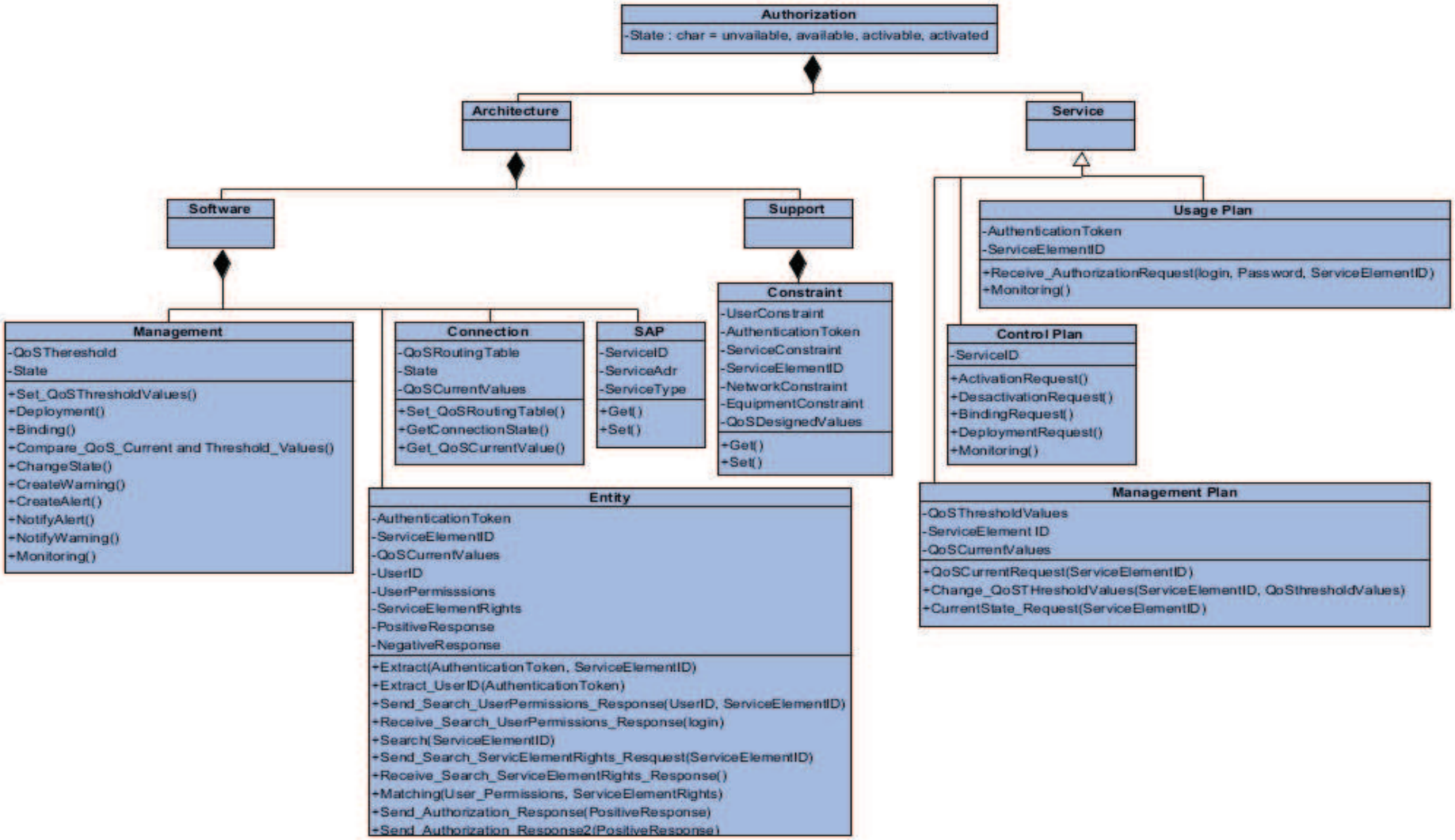



***Figure 55 :*** *Modèle informationnel du service d'autorisation.*

# 11

# Conclusions et Perspectives

Cette thèse avait pour objectif d'apporter des contributions au niveau de la gestion de la sécurité d'une session « user centric » dans un contexte NGN/ NGS mobile et hétérogène en proposant une approche architecturale et protocolaire permettant d'assurer la sécurité de **bout en bout** de cette session, nous allons résumer nos propositions (§11.1) et définir nos perspectives (§11.2).

## 11.1. Contributions

Notre objectif était d'assurer la sécurité de la session « user-centric » dans un environnement mobile et hétérogène. Dans notre contexte, tout est fourni à travers des composants de services. La composition de services, permet la personnalisation de la session selon les préférences et les exigences de l'utilisateur. Pour atteindre notre objectif, nous avons d'abord délimité le périmètre de notre contexte en s'intéressant à celui qui associe la convergence des problématiques de mobilité, d'hétérogénéité et des besoins centrés sur l'utilisateur. Ensuite nous avons analysé leurs impacts sur la sécurité de la session et sur sa continuité. Ceci nous a amené à identifier *les verrous* et à proposer des solutions d'ordre architectural et protocolaire.

Pour les aspects architecturaux nous avons proposé :

- Une approche « as a Service ». *Dans un tel contexte, fournir la sécurité comme service s'avère nécessaire.* Ceci nous a conduits à repenser la sécurité selon une composition de services, et de proposer une nouvelle architecture de sécurité qui s'appuie sur trois piliers : le Securityware, les Security Agents et le Security Datastore. Cette architecture fournit des services de sécurité autonomes, interopérables, mutualisables, auto-gérables et stateless.

- Une session sécurisée qui s'aligne à l'approche « User Centic ». *Face au besoin de la sécurisation de la session mobile et dynamique basée sur une composition de service,* notre contribution consiste à intégrer la sécurité dans la session de l'utilisateur sous forme de services composables d'une manière transparente. Ces services de sécurité interviennent lors de la session de l'utilisateur pour la composition des services applicatifs (VPSN) et des services au niveau terminal (VPDN). Ils sont sollicités également lors des changements induits par les différentes «mobilités» et les variations du contexte.

- Un service d'audit. *Afin d'assurer le contrôle et le maintien du niveau de sécurité requis,* nous avons proposé un Audit de sécurité basé QoS. Cette proposition

s'appuie sur un service de monitoring déployé au niveau de chaque service de sécurité (authentification, autorisation, etc.) permettant d'effectuer les mesures comme dans tous les services de notre système. L'évaluation de ces mesures, liée au comportement du service, effectuée par l'agent de QoS est considérée comme un premier diagnostic sur lequel s'appuie notre Audit de sécurité. Il permet ainsi de contrôler les aspects de sécurité en se basant sur les critères de QoS et de prendre donc les décisions nécessaires pour des fins correctives ou pour des améliorations.

Pour les aspects protocolaires, nous avons d'abord participé à la proposition de l'extension de la portée du protocole SIP existant, au niveau service de l'architecture pour offrir à l'utilisateur une session adaptée à ses besoins et à son contexte ambiant. Le protocole étendu est nommé SIP+. L'intérêt majeur du protocole SIP+ est de permettre lors des choix de ses services, une négociation de la qualité de service pour chaque service demandé par l'utilisateur pendant la phase d'initiation de la session user-centric. Ce protocole est utilisé bien évidement pour établir, modifier ou bien terminer des sessions user-centric.

- *Face à la mobilité de l'utilisateur (changement de terminal) et le besoin d'assurer son authentification durant cette mobilité,* nous avons défini un jeton de sécurité acheminé par le protocole SIP+ (l'extension du protocole SIP) permettant de préserver la continuité de la sécurité en changeant le terminal. Ce nouveau protocole SIP+ basé sur le jeton permet donc de construire le VPDN de l'utilisateur et de garder l'authentification de l'utilisateur pour l'ensemble de ses terminaux.

- *Dans notre environnement trans-organisationnel qui est caractérisé par la présence de plusieurs fournisseurs de service,* nous avons adopté l'approche de fédération qui permet de passer d'une plate-forme de service à une autre.

- *Face au besoin de l'authentification et de l'autorisation pour tous les composants des services sollicités,* nous avons assuré l'unicité et la continuité de la session avec un accès sécurisé grâce au protocole SIP+ basé sur le jeton qui nous a permis de garantir une authentification unique auprès de tous les services faisant partie du VPSN . Pour pouvoir composer et ajouter des services, le service d'autorisation inclus dans le VPSN permet de valider les droits de l'utilisateur service par service.

- *Pour sécuriser le protocole de signalisation dans la session (dans notre cas, il s'agit du protocole SIP+),* nous avons proposé des mécanismes de sécurité (signature, chiffrement) appliquée aux messages SIP+ permettant d'assurer la sécurité de bout en bout de la session contre les attaques potentielles. Ces mécanismes garantissent principalement l'intégrité et la confidentialité des informations de sécurité encapsulées dans le jeton et les informations de QoS contenues dans les messages SIP+.

## 11.2. Perspectives

Notre travail de thèse rentre dans le cadre du projet UBIS, dans lequel nous avons conçu une nouvelle architecture de sécurité et une approche protocolaire adaptées au contexte

NGN/NGS. Il nous faudrait évaluer les performances de nos propositions, et puis modéliser des services de sécurité à savoir la fédération selon notre modèle de service et de QoS.

Nos propositions peuvent aussi s'intégrer et répondre aux besoins de sécurité dans le contexte du Cloud.

# Liste des Publications

- A.Hammami, N.Simoni, «Mobilité et Sécurité : les nouveaux défis du contexte NGN », GRES, Montréal, Canada, Oct. 2010.

- A.Hammami, N. Simoni , «Secure and Seamless Session Management in Mobile and Heterogeneous Environment », SECRYPT, ROME, Italy, Jul. 2012 .

- A.Hammami, N. Simoni , « Securityware: Towards Architecture for User-Centric Approach» , SECURWARE, ROME, Italy, Aug .2012

- A.Hammami, N. Simoni , « Ubiquity and QoS for Cloud Security » , ICPP, Pittsburgh, USA , Sep 2012

- A.Hammami, N.Simoni, «Security of user-centric session in Mobileand Heterogeneous Environment», 10th Australian Information Security Management Conference , Secau Security Congress , Perth, Western Australia, Dec. 2012 .

# Liste des Délivrables UBIS

- Hammami, N.Simoni, " Lot5.1: Analyse des besoins et de l'existant ", Avril 2010
- Hammami, N.Simoni, " Lot5.3: Architecture de la sécurité des services ", Novembre 2011

# Bibliographie


[1] Booth D., Haas H., McCabe F., Newcomer E., Champion M., Ferris C., Orchard D. W3C working group note: Web Services Architecture. W3C open standards Community, Feb. 2004.

[2] Gudgin M., Hadley M., Mendelsohn N., Moreau J.J., Nielsen H.F., Karmarkar A., Lafon Y. SOAP, version 1.2, part 1 : messaging framework, second edition. W3C Recommendation: SOAP. W3C open standards community, Apr. 2007.

[3] Christensen E., Curbera F., Meredith G., Weerawarana S. W3C note: Web Services Description Language (WSDL) 1.1. W3C open standards community, Mar. 2001.

[4] Chinnici R., Moreau J.J., Ryman A., Weerawarana S. Web Services Description Language (WSDL), version 2.0, part 1: CoreLanguage. W3C Recommendation: WSDL. W3C open standards community, Jun. 2007.

[5] Clement L., Hately A., Von Riegen C., Rogers T. UDDI, version 3.0.2, UDDI spec technical committee draft. OASIS open standards consortium, Oct. 2004.

[6] Organisation for the Advancement of Structured Information Standards- http://www.oasis-open.org

[7] XML Signature Syntax and Processing- W3C Recommendation http://www.w3.org/TR/2008/REC-xmldsig-core-20080610

[8] XML Encryption Syntax and Processing- W3C Recommendation http://www.w3.org/TR/2002/REC-xmlenc-core-20021210

[9] WS-Security: SOAP Message Security 1.1-OASIS Standard Specification- http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[10] Web service Reliable Messaging TC WS-Reliability 1.1-OASIS Standard http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf

[11] Assertions and Protocols for the Oasis Security Assertion Markup Language (SAML) V2.0-OASIS Standard- http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[12] eXtensible Access Control Markup Language (XACML) version 2.0- OASIS Standard http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[13] Web Service Policy 1.5 Framework- W3C Recommendation http://www.w3.org/TR/2007/REC-ws-policy-20070904

[14] WS-SecurityPolicy 1.2- OASIS Standard http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf

[15] WS-Trust 1.3 – OASIS Standard http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.pdf

[16] Web Services Federation Language (WS-Federation), version 1.1 http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf

[17] Erl T. Service-Oriented Architecture: concepts, technology, and design, chapter SOA and Service-Orientation, pages 244-313. Prentice Hall PTR, Aug. 2005.

[18] Kanneganti R. and Chodavarapu P. Soa Security, pages 3-33. Manning Publications Co, 2008.

[19] Shah D. and Patel D. Dynamic and Ubiquitous Security Architecture for Global SOA, UBICOMM'08, Mobile Ubiquitous Computing, Systems, Services and Technologies, pages 482-487, IEEE , 2008.

[20] Boehm O. and Caumanns J. and Franke M. and Pfaff O. Federated Authentication and Authorization: A Case Study, EDOC'08. Enterprise Distributed Object Computing Conference, pages 356-362, IEEE, 2008.

[21] Qi-rui P., Cheng W., Jing W., Jun L., Qing L. and Beien S. An authentication and authorization solution supporting SOA-based distributed systems. In Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on, pages 535-538. IEEE, 2010.

[22] Bertino E. and Martino L.D. A service-oriented approach to security–concepts and issues. In Future Trends of Distributed Computing Systems, 2007. FTDCS'07. 11th IEEE International Workshop on, pages 31-40. IEEE, 2007.

[23] Emig C., Brandt F., Kreuzer S., and Abeck S. Identity as a service-towards a service-oriented identity management architecture. In Proceedings of the 13th open European summer school and IFIP TC6. 6th conference on Dependable and adaptable networks and services, pages 1-8. SpringerVerlag, 2007.

[24] Emig C., Brandt F., Abeck F., Biermann J., and Klarl H. An access control metamodel for web service-oriented architecture. In Software Engineering Advances, 2007. ICSEA 2007. International Conference on, pages 57-57. IEEE, 2007.

[25] Pervez Z., Lee S. and Lee Y. Multitenant,secure, load disseminated SaaS architecture. In Advanced Communication Technology (ICACT), The 12th International Conference on, volume 1, pages 214-219. IEEE, 2010.

[26] Chowdhury NM. and Boutaba R. A survey of network virtualization. Computer Networks, 54(5):862-876, 2010.

[27] Takabi H., J BD Joshi J., and Ahn G. Security and privacy challenges in cloud computing environments.Security & Privacy, IEEE, 8(6):24-31, 2010.

[28] Ibrahim A. S., Hamlyn-harris J. H., and Grundy J. Emerging security challenges of cloud virtual infrastructure, 2010.

[29] Ko M., Ahn G.-J., and Shehab M. "Privacy-Enhanced User-Centric Identity Management," Proc. IEEE Int'l Conf. Communications, IEEE Press, pp. 998-1002, 2009.

[30] Wayne J. and Timothy G., "NIST Guidelines on Security and Privacy in Public Cloud Computing," Draft Special Publication 800-144, 2011.

[31] OpenID Authentication 2.0, OpenID Foundation, http://openid.net/specs/openid-attribute-exchange-2 0.html, 2007.

[32] Joshi J. and al., “Access Control Language for Multi-domain Environments,” IEEE Internet Computing, vol. 8, no. 6, pp. 40–50, 2004.

[33] Blaze M. and al., “Dynamic Trust Management,” Computer, vol. 42, no. 2, pp. 44–52, 2009.

[34] Zhang Y. and Joshi J., “Access Control and Trust Management for Emerging Multidomain Environments,” Annals of Emerging Research in Information Assurance, Security and Privacy Services, S. Upadhyaya and R.O. Rao, eds., Emerald Group Publishing, pp. 421–452, 2009.

[35] Shin D. and Ahn G.-J., “Role-Based Privilege and Trust Management,” Computer Systems Science & Eng. J., vol. 20, no. 6, pp. 401–410, 2005.

[36] ISO. ISO 20000-1 IT Service management standard: specification for service Management. 2005.

[37] ISO/IEC 27004:2009. Technologies de l'information -- Techniques de sécurité -- Management de la sécurité de l'information – Mesurage. Organisation internationale de normalisation (ISO) et Commission électrotechnique internationale (CEI). 2004.

[38] Lloyd V. ITIL Continual Service Improvement. The Stationery Office (TSO). 23 Aug 2011.

[39] Alsaleh M. and Adams C. “Enhancing Consumer Privacy in the Liberty Alliance Identity Federation and Web Services Frameworks”. In: *Workshop on Privacy Enhancing Technologies*, pages. 59–77, LNCS 4258, Springer-Verlag, Cambridge, UK, 2006.

[40] Recordon D. and Reed D. “OpenID 2.0: a platform for user-centric identity management”. In: *Second ACM workshop on Digital identity management*, pages.11–16, ACM Press, Alexandria, Virginia, USA, 2006.

[41] RFC 3261: SIP (Session Initiation Protocol) (2002-06)

[42] RFC 4566 : SDP (Session Description Protocol) (2006)

[43] M.Poikselkä, G.Mayer, H.Khartabil, A.Niemi - The IMS: IP Multimedia Concepts and Services, 2nd edition, Wiley, 2006.

[44] ETSI TS 123 279 V10.0.0 (2011-03): “Technical Specification Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Combining Circuit Switched (CS) and IP Multimedia Subsystem (IMS) services.

[45] El Sawda S. and Urien P., «SIP Security Attacks and Solutions: A state-of the-art review», In proceedings of the Information and Communication Technologies, ICTTA'06, April 2006.

[46] Werapun W., El kalam A., Paillassa B., Fasson J.Solution Analysis for SIP Security Threats, ICMCS'09, Ouarzazate. (2009)

[47] Arkko J., Torvinen V., Camarillo G., Niemi A. and Haukka T. Security Mechanism Agreement for the Session Initiation Protocol RFC 3329 IETF.2003.

[48] Geneiatakis D., Kambourakis G., Dagiuklas T., Lambrinoudakis T., and Gritzalis S., «SIP Security Mechanisms: A State-of-the-art Review», In the Proceedings of the Fifth

International Network Conference (INC 2005), pages 147-155 ,Samos, Grèce, juillet 2005.

[49] Franks J., Hallam-Baker P., Hostetler J., Lawrence S., Leach P., Luotonen A. and L. Stewart L., “HTTP Authentication: Basic and Digest Access Authentication”, RFC 2617, juin 1999.

[50] Alaoui Soulimani H., Simoni N., Coude P. “User-centric and QoS-based service session, conférence IEEE APSCC” APSCC 2011, Jéju, Corée du sud, Décembre 2011.

[51] Dierks T. and Allen C. The TLS Protocol Version 1.0, IETF RFC 2246.1999.

[52] VIRTUOR. Réseaux Virtuels VirtuOR. http://www.virtuor.fr/.

[53] FOKUS. Open IMS Core. www.open-ims.org/core/.

[54] SUN Microsystems. GlassFish Server. http ://glassfish.java.net/.

[55] Ornelas N., Simoni N., Chen K., and Boutignon A., “VPIN: User-session knowledge base for self-management of ambient networks”, UBICOMM’08, Spain, Oct. 2008,

[56] Simoni N., Yin C., Du Chene G., « An intelligent user centric middleware for NGN: Infosphere and AmbientGrid”, In Proc. of COMSWARE, pages 599–606, Bangalore, India, Jan. 2008.

[57] OpenSSO Project. https://opensso.dev.java.net/.

[58] SUN Microsystems. Sailfin. http ://sailfin.java.net/.

[59] SIPP “ Portal for SIPP” www.sipp.sourceforge.net

# Acronymes

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| ACK | Acknowledgement |
| B2B | Business to Business |
| CORBA | Common Object Request Broker Architecture |
| DoS | Denial of Service |
| EC2 | Elastic Compute Cloud |
| ES | Elément de Service |
| ESB | Enterprise Service Bus |
| FTP | File Transfer Protocol |
| GPRS | General Packet Radio Service |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| IaaS | Infrastructure as a Service |
| ID | Identifier |
| IDM | Identity Management |
| IEC | International Electrotechnical Commission |
| IETF | Internet Engineering Task Force |
| IIOP | Internet Inter-ORB Protocol |
| IMS | IP Multimedia Subsystem |
| IP | Internet Protocol |
| IPsec | Internet Protocol Security |
| IPTV | Internet Protocol Television |
| ISV | |
| ISO | International Organization for Standardization |
| IT | Information Technology |
| J2EE | Java Enterprise Edition |
| Java-RMI | Java-Remote method invocation |
| KDC | Key Distribution Center |
| LDAP | Lightweight Directory Access Protocol |
| M2M | Machine to Machine |
| MD5 | Message Digest 5 |
| MIT | Massachusetts Institute of Technology |

| | |
|---|---|
| MTU | Maximum Transmission Unit |
| NGN | Next Generation Network |
| NGS | Next Generation Services |
| NIST | National Institute of Standards Technology |
| NLR | Nœud, Lien, Réseau |
| NOK | Not OK |
| OASIS | Organization for the Advancement of Structured Information Standards |
| OSI | Open System Interconnection |
| PaaS | Platform as a Service |
| PAN | Personal Area Network |
| PDA | Personal Digital Assistant |
| PDP | Packet Data Protocol |
| PDP | Policy Decision Point |
| POP | Post Office Protocol |
| PS | Packet Switched |
| QoS | Quality of Service |
| RBAC | Role-Based Access Control |
| RFC | Request for Comments |
| RMI-IIOP | Java Remote Method Invocation over Internet Inter-ORB Protocol |
| ROI | Return On Investment |
| RTP | Real-Time Transport Protocol |
| S/MIME | Secure / Multipurpose Internet Mail Extensions |
| SaaS | Software as a Service |
| SAML | Security assertion markup language |
| SCA | |
| SDP | Session Description Protocol |
| SE | Service Element |
| SIP | Session Initiation Protocol |
| SLA | Service Level Agreement |
| SMS | Short Message Service |
| SMSI | Système de Management de la Sécurité de l'Information |
| SMTP | Simple Mail Transport Protocol |
| SOA | Service-Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SP | Service Provider |
| SQL | Structured Query Language |
| SRTP | Secure Real-time Transport Protocol |
| SSL | Secure Sockets Layer |
| SSO | Single Sign On |
| TCP | Transmission Control Protocol |
| TGT | Ticket Granting Ticket |
| TINA | Telecommunications Information Networking Architecture |
| TLS | Transport Layer Security |
| TTM | Time To Market |
| UA | User Agent |
| UBIS | “User-centric”: uBiquity and Integration of Services |
| UDDI | Universal Description Discovery and Integration |

| | |
|---|---|
| UDP | User Datagram Protocol |
| URL | Uniform Resource Locator |
| URI | Uniform Resource Identifier |
| | |
| VoD | Video on Demand |
| VoIP | Voice over Internet Protocol |
| VPDN | Virtual Private Device Network |
| VPN | Virtual Private Network |
| VPSN | Virtual Private Service Network |
| | |
| W3C | World Wide Web Consortium |
| WS | Web Service |
| WSDL | Web Services Description Language |
| WSFL | Web Services Flow Language |
| WSRP | |
| | |
| XACML | eXtensible Access Control Markup Language |
| XKMS | XML Key Management Specification |
| XML | Extensible Markup Language |
| XML-RPC | XML-Remote Procedure Call |