Université de Limoges

École Doctorale Sciences et Ingénierie des Systèmes, Mathématiques, Informatique

Thèse de doctorat

Spécialité Informatique

Confidentialité et traçabilité dans les systèmes publish/subscribe pour des applications de santé

---

présentée et soutenue le 18 mars 2022 par

Mathieu KLINGLER

Rapporteurs :

Yacine GHAMRI-DOUDANE, Professeur, Université de La Rochelle

Maryline LAURENT, Professeure, Télécom SudParis

Examinateurs :

Olivier BLAZY, Professeur, Ecole Polytechnique de Paris

Sylvain GIROUX, Professeur, Université de Sherbrooke

Thèse dirigée par :

Damien SAUVERON, Maître de conférences HDR, Université de Limoges

Emmanuel CONCHON, Maître de conférences, Université de Limoges

Remerciements

Tout d'abord je tiens à remercier mes directeurs de thèse Damien Sauveron et Emmanuel Conchon. Ils ont toujours été disponibles, à l'écoute de mes nombreuses questions, et se sont toujours intéressés à l'avancée de mes travaux. Leurs nombreux conseils m'ont également permis d'améliorer considérablement la qualité des articles que nous avons publié ainsi que de ce manuscrit, et je les remercie encore une fois pour leur temps et leur patience.

J'aimerais également adresser mes remerciements à mes rapporteurs, Yacine Ghamri-Doudane et Maryline Laurent, pour avoir accepté d'être les rapporteurs de cette thèse.

Je remercie Sylvain Giroux, qui m'a accueilli à l'Université de Sherbrooke pendant un mois et m'a conseillé lors de l'élaboration de plusieurs travaux.

Je remercie également Olivier Blazy, qui a toujours accepté de partager des pauses café, même en sachant que j'en profiterai pour lui poser une infinité de questions sur la cryptographie.

Un grand merci à mes collègues et amis, Hamza, Laura, Loïc, Maxime, Neals, Nicolas et Théo, avec qui j'ai partagé d'excellents moments, pauses café, matchs de tennis, ainsi que de nombreuses réflexions sur le plan scientifique.

Je remercie mes amis Alexandre, Justine, Laura, Léa, Loïc, Maxime et Nadège, avec qui je passe toujours de merveilleux moments.

Je remercie infiniment les membres de ma famille qui ont toujours été à mes côtés pendant cette thèse et qui m'ont toujours encouragé.

Finalement, je remercie chaleureusement Coralie pour son soutien et sa tendresse, qui m'a accompagné pendant toute cette thèse.

Je n'oublie pas mon chaton Orion qui a toujours essayé de m'aider, tout particulièrement en marchant sur mon clavier pendant la rédaction de ce manuscrit.

Table des matières

Table des figures

Liste des tableaux

Glossaire

— **IBE (Identity-based Encryption)** : Chiffrement basé sur l'identité

— **ABE (Attribute-based Encryption)** : Chiffrement basé sur les attributs

— **IBS (Identity-based Signature)** : Signature basée sur l'identité

— **ABS (Attribute-based Signature)** : Signature basée sur les attributs

— **MA-ABE (Multi-Authority Attribute-based Encryption)** : Chiffrement basé sur les attributs géré par plusieurs autorités

— **PIR (Private Information Retrieval)** : Protocole de récupération d'informations privées

— **SE (Searchable Encryption)** : Chiffrement recherchable

— **Trapdoor** : Désigne une requête vers une base de donnée générée par un utilisateur dans un schéma de Searchable Encryption

— **ABKS (Attribute-based Keyword Search)** : Recherche de mots clés chiffrés basé sur les attributs

— **DVS (Designated Verifier Signature)** : Signature à vérifieur désigné

— **ABDVS (Attribute-Based Designated Verifier Signature)** : Signature à vérifieur désigné basée sur les attributs

Chapitre 1

Introduction

Sommaire

1.1 Motivation

Au cours de ces dernières années, on a pu observer une augmentation conséquente de l'intérêt porté à l'Internet des objets, qu'on appelle aussi l'Internet of Things (IoT). Cet intérêt s'illustre par son utilisation massive, que ce soit par les entreprises ou chez des particuliers. Une étude de 2018 effectuée par le McKinsey Global Institute [40] évalue que 127 nouveaux objets sont connectés chaque seconde sur Internet. Cette étude estime que l'impact économique de l'IoT s'élèvera aux alentours des 11,1 milliards de dollars par an en 2025.

L'organisme ITU-T [68] définit l'IoT de la manière suivante : «*Une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution*».

Les objets connectés sont également caractérisés par leur nombre, leur diversité et la quantité d'information qu'ils génèrent. Ils permettent de faire évoluer notre quotidien, au travers de nombreux domaines d'applications. Les objets connectés sont donc utilisés dans plusieurs domaines différents [31],[79] dont les principaux sont : l'industrie, les villes intelligentes, les systèmes de transport intelligent et le domaine de la santé qui nous intéressera plus particulièrement dans le cadre de cette thèse.

Industrie

Dans le secteur industriel, les objets connectés permettent d'améliorer les processus de fabrication, de superviser les installations industrielles et d'offrir une infrastructure optimisée afin de réduire les coûts de fabrications et d'améliorer la sécurité humaine dans les zones industrielles [52]. L'IoT en milieu industriel a donc des effets bénéfiques au niveau économique et au niveau de la sécurité des acteurs.

Les villes intelligentes

L'utilisation des objets connectés dans les villes intelligentes est évidemment importante et contribue à améliorer la gestion de villes où la population a considérablement augmenté au cours de ces dernières décennies. L'optimisation des ressources et de l'énergie est le principal challenge de ces villes intelligentes. On peut par exemple noter la régulation et l'optimisation de la consommation énergétique ou de la consommation d'eau en fonction de la météo, le contrôle de la pollution avec un ensemble de capteurs informant les habitants de la ville, ou même améliorer la sécurité globale de la ville via la télésurveillance [6]. Les objets connectés permettent également de mettre en place dans ces villes de l'agriculture intelligente, avec des capteurs collectant des données en temps réel sur les conditions climatiques et une gestion à distance de robots effectuant les tâches principales telles que le désherbage, l'arrosage intelligent, etc. [62].

Les systèmes de transport intelligent

Une autre application notable des objets connectés porte sur les systèmes de transport intelligent - Intelligent Transportation System (ITS). L'ITS permet d'aider à l'automatisation des chemins de fer, des routes, des voies aériennes et marines, afin d'améliorer l'expérience des usagers ainsi que leur sécurité [88]. Cela permet également un meilleur suivi des marchandises qui sont transportées et livrées.

Les objets connectés sont donc utilisés massivement et s'inscrivent de plus en plus dans notre quotidien. Ils permettent une meilleure gestion des ressources afin d'avoir un impact positif sur l'économie et l'environnement. Ils permettent également d'améliorer le confort des citoyens et des usagers aux travers de projets comme l'ITS. Or, ces usages et la quantité d'information traitée soulèvent plusieurs questions relatives à la protection de la vie privée des utilisateurs. Le domaine de la santé étant le secteur d'activité manipulant les données les plus sensibles, nous allons présenter le rôle de l'Internet des objets dans ce secteur de manière détaillée. Puis, nous allons définir les architectures utilisées dans ce type d'environnement ainsi que les différentes problématiques de sécurité associées.

1.1.1 L'IoT dans le domaine de la santé

L'utilisation des objets connectés dans le domaine de la santé soulève de nombreuses questions, c'est pourquoi son impact au niveau de la protection de la vie privée des patients a été très étudiée au cours de ces dernières années [130],[27],[5].

Les objets connectés permettent en effet de faciliter le travail des professionnels de santé, ainsi que de proposer aux utilisateurs différents services. On peut noter par exemple la supervision de l'état de santé des patients à distance, la vérification de la conformité de traitements médicaux, ainsi qu'un suivi médical plus précis des patients [53]. Plusieurs capteurs positionnés dans un habitat peuvent également faciliter la vie et augmenter la sécurité de personnes âgées ou handicapées comme l'ont montré de nombreux projets d'aide à l'autonomie à domicile - Ambient Assisted Living (AAL) [42],[114],[126].

On retrouve deux catégories d'objets connectés utilisés dans le domaine de la santé : les capteurs portés et les capteurs ambiants.

— Les capteurs portés sont utilisés aussi bien dans les établissements de santé que dans les environnements d'aide à l'autonomie à domicile. Cela peut être des capteurs comme des glucomètres, des oxymètres de pouls, des capteurs de pression artérielle, etc.

— Les capteurs ambiants sont davantage utilisés dans les environnements d'aide à l'autonomie à domicile et sont donc disposés dans l'habitat du patient. Cela peut être des caméras, des détecteurs de chute, des actionneurs connectés aux différentes portes et fenêtres pour

vérifier si elles sont bien fermées.

L'étude [114] recense de nombreux objets connectés ambiants et portables utilisés dans ce type d'habitat connecté.

L'ensemble des informations collectées par ces objets connectés sont envoyées vers des professionnels de santé et peuvent donc être stockées sur des serveurs distants. Ces données doivent être accessibles facilement par ces professionnels mais doivent également être protégées contre les potentiels utilisateurs malveillants.

1.1.2 Problématiques de sécurité de l'IoT dans le domaine de la santé

En pratique, les architectures IoT reposent sur un schéma en étoile au centre duquel se trouve un relais en charge de l'acheminement des messages entre les producteurs de données (les objets connectés) et les consommateurs de données (les utilisateurs, patients ou professionnels de santé). Ce relais est donc en général connecté à Internet. Par conséquent, pour protéger au mieux les données sensibles, on souhaite au maximum limiter la quantité d'informations à lui fournir tout en garantissant son bon fonctionnement. On le considère conforme au modèle de sécurité *honest-but-curious*, c'est-à-dire qu'il suit le protocole et achemine les données. Il ne corrompt pas ces dernières, mais essaye d'obtenir le plus d'informations possible de ce qu'il relaie.

Par ailleurs, le patient doit avoir un contrôle sur les accès à ses données. En effet, au niveau utilisateur, il est très difficile vu le grand nombre d'objets connectés utilisés et fournissant des données de savoir qui accède à ses informations. De plus, le coût du stockage d'information étant en constante baisse depuis plusieurs années, il est d'autant plus difficile de contrôler ces accès lorsque ces informations peuvent rester stockées indéfiniment.

L'utilisation d'objets connectés dans le domaine de la santé présente de sérieux enjeux de sécurité et d'infrastructures réseaux, puisque ces données nécessitent parfois d'être consultées et utilisées en urgence et doivent ainsi être accessibles en temps réel par les personnes qualifiées.

Il faut également prendre en compte que malgré leurs avantages, l'adoption des objets connectés peut rencontrer certaines réticences au niveau des patients. Il est donc d'autant plus important, pour ne pas freiner leurs usages, que ces objets présentent des garanties de sécurité et de

protection de la vie privée.

Les objets connectés collectent et envoient beaucoup d'informations, et ces informations nécessitent souvent un traitement en temps réel. L'ensemble de ces particularités amène à l'utilisation d'infrastructures particulières, spécifiques à l'IoT. Il est nécessaire de connaître en détail ces architectures avant d'élaborer de nouvelles solutions de sécurité.

1.1.3 Infrastructures utilisées dans l'IoT

L'utilisation grandissante de l'IoT a encouragé le développement de diverses architectures de communications. Plusieurs travaux au niveau de ces architectures peuvent être retrouvés dans les états de l'art suivants : [115, 57, 39].

Au cours de ces dernières années, l'architecture orientée services (Service-Oriented Architecture - SOA) et l'intergiciel à messages (Message-Oriented Middleware - MOM) sont les approches qui se démarquent le plus dans des contextes IoT. Ces deux approches restent cependant très différentes, c'est pourquoi nous détaillons dans la suite de cette section leurs caractéristiques principales ainsi que leurs différences. Nous justifierons également le choix des approches de types MoM au sein de ces travaux de thèse.

SOA vs MOM

Une architecture SOA consiste en trois entités qui interagissent directement entre elles : un fournisseur de service (Service Provider), un consommateur de service (Service Consumer), ainsi qu'un serveur qui sert de registre pour les services. Selon [104], un modèle SOA possède les propriétés suivantes :

— Les applications s'exécutent sur des architectures matérielles différentes et peuvent communiquer efficacement les unes avec les autres.

— Le consommateur de service interagit directement avec le fournisseur de service pour la durée de l'utilisation du service.

— Les trois entités de l'architecture (le fournisseur de service, le registre et le consommateur de service) sont liées.

— Les interactions principales de l'architecture sont : la publication, la recherche et la liaison.

— Les consommateurs de services ciblent un service via une interface spécifique. Cette dernière est proposée par le registre de services via les mécanismes de découverte de service.

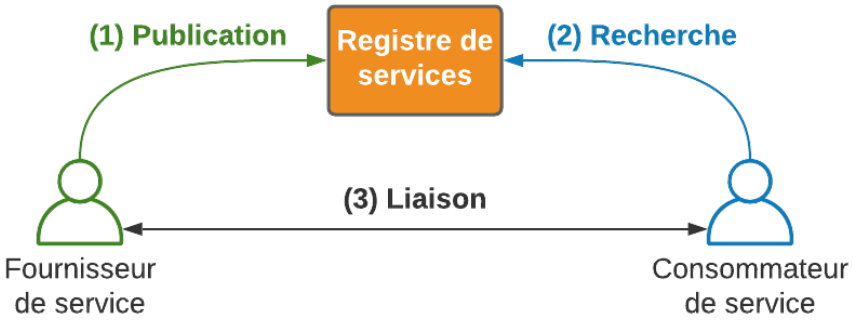


FIGURE 1.1 – Service-Oriented Architecture

L'ensemble des entités et des interactions possibles entre elles sont récapitulées en Figure 1.1. On observe trois phases, avec : (1) la publication d'un service dans le registre, (2) la recherche du service et (3) la liaison entre le fournisseur et le consommateur de service.

Dans un contexte IoT, les capteurs sont des services pour les applications consommatrices et les fournisseurs de services décrivent les caractéristiques des objets connectés au niveau du serveur de registre. Ils interagissent ensuite directement avec les applications.

Leur adaptabilité au fonctionnement de l'IoT a amené la recherche et le développement d'un grand nombre de solutions IoT basées sur l'architecture orientée services, on peut noter : SIRENA [19], COSMOS [76], SOCRADES [124], HYDRA [8], MUSIC [118], SENSEI [132], ubiSOAP [25], Servilla [55], KASOM [35], MSOAH-IoT [85], AUSOM [13], RDS [123], etc.

L'architecture SOA est donc basée sur les services, ce qui la distingue de l'architecture MOM qui est basée sur les événements (ou messages).

Dans une architecture MOM, un événement est caractérisé par un entête et un message, qui peut prendre n'importe quel format. Un serveur intermédiaire sert de lien entre les producteurs de données et les consommateurs de données, de manière à ce que la donnée soit au centre du processus.

Selon [37], l'approche MOM possède les caractéristiques suivantes :

— Les producteurs et consommateurs peuvent s'exécuter sur des architectures matérielles différentes et communiquent via un serveur intermédiaire.

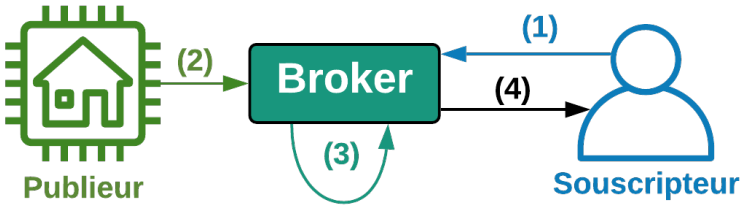


FIGURE 1.2 – Architecture publish/subscribe

— Les applications n'ont pas besoin d'être disponibles en même temps pour échanger. Le serveur intermédiaire entre le producteur et le consommateur peut conserver la donnée si besoin.

— Les producteurs de données et les consommateurs de données n'interagissent pas directement pour échanger de l'information. On parle ici de couplage faible.

Il existe deux modèles de MOM : l'architecture point à point et l'architecture publieur/souscripteur (ou publish/subscribe). Le publish/subscribe est une version plus générale du point à point et s'adapte mieux à un système IoT, c'est pourquoi il nous sert d'exemple ici, mais nous présentons ces deux modèles ainsi que leurs différences plus en détail dans le chapitre 2.

Dans une architecture publish/subscribe, les données envoyées sont appelées des publications, les producteurs de ces données des publieurs et les consommateurs des souscripteurs. Ils s'échangent des messages via un serveur appelé broker de messages que nous appellerons simplement broker dans la suite de ce document.

Lorsque l'on utilise cette architecture dans un contexte IoT, chaque objet connecté peut être un publieur ou un souscripteur, et publier/souscrire à plusieurs informations. L'avantage de ce type d'architecture est que les publieurs et les souscripteurs n'ont pas à dialoguer directement. De plus, l'ajout et la suppression d'un utilisateur se fait de manière transparente pour le système ce qui le rend très flexible.

Le processus général de communication d'une architecture publish/subscribe est illustré en Figure 1.2. On peut le diviser en quatre étapes : (1) les souscripteurs envoient une souscription au broker et le broker la stocke dans sa base d'abonnements ; (2) le publieur envoie une publication ; (3) le broker parcourt sa base d'abonnements et récupère les souscriptions correspondant à la publication reçue ; (4) pour chaque correspondance, la publication est transmise vers le souscripteur

intéressé.

Les solutions publish/subscribe sont également très étudiées et ont été développées au cours de ces dernières années, avec par exemple les solutions : Pogo [23], LooCI [65], Trinity [111], PADRES [71], Dynatops [138] ou MQTT [11], qui est un standard OASIS [125].

Le modèle SOA apporte ainsi un meilleur contrôle sur les données qui circulent dans l'architecture, grâce au contrat établi entre le fournisseur et le consommateur. Cependant, cette propriété pose également des problèmes de flexibilité et de mise à l'échelle, qui sont deux propriétés très importantes dans un contexte IoT.

Il est en effet plus difficile d'ajouter et supprimer des utilisateurs ou des services dans un modèle où le couplage entre les fournisseur et consommateur de données est aussi fort. C'est pourquoi nous avons choisi dans cette thèse de nous intéresser à la sécurité des approches MOM, qui permettent dans un contexte IoT un passage à l'échelle plus simple.

Nous présentons maintenant les objectifs principaux de cette thèse ainsi que ses principales contributions.

1.2 Objectifs, organisation et contributions de la thèse

Les objectifs globaux de cette thèse sont les suivants :

— Étudier différentes architectures MOM dans un contexte IoT.

— Définir un modèle de sécurité pour ces différentes architectures.

— Étudier et adapter des cryptosystèmes afin de les appliquer à ces architectures.

— Proposer des solutions de sécurité pour ces architectures de communication afin d'assurer la sécurité et la confidentialité des données des utilisateurs dans un contexte de santé.

Le plan que nous suivrons dans ce document est le suivant :

— Chapitre 2 : Nous présentons un état de l'art des architectures MOM, de leurs applications dans le domaine de la santé, ainsi que les exigences de sécurité de ce type d'architecture.

Nous commençons par développer le fonctionnement des deux modèles de MOM existants : le point-à-point et le publish/subscribe. Nous détaillons ensuite le fonctionnement du pu-

blish/subscribe, ainsi que ces deux variantes que sont le topic-based publish/subscribe et le content-based publish/subscribe.

Puis, nous présentons deux cas d'application du publish/subscribe dans le domaine de la santé. Le premier est la gestion des données personnelles des patients dans un environnement d'aide à l'autonomie à domicile. Dans ce scénario, toutes les données collectées dans leur vie quotidienne sont externalisées vers des professionnels de santé. Le second scénario implique un Groupement Hospitalier de Territoire, qui est un ensemble d'établissements de santé. Ce scénario présente des challenges au niveau de la quantité d'information à gérer vu le très grand nombre d'acteurs impliqués. C'est une itération à plus grande échelle du scénario précédent.

Enfin, nous définissons les exigences de sécurité adaptées à ces scénarios. Ces dernières servent de point de départ à l'élaboration de nos contributions qui sont présentées dans les chapitres suivants.

— Chapitre 3 : Nous présentons un état de l'art de différents cryptosystèmes qui permettent de garantir la sécurité et la confidentialité des données dans les architectures précédemment détaillées. Nous distinguons trois moyens complémentaires qui contribuent à assurer la sécurité et la confidentialité dans une architecture publish/subscribe : le chiffrement des données, le chiffrement recherchable appliqué aux topics et la signature électronique.

Le chiffrement des données permet de garantir la confidentialité du contenu des publications afin que seul le ou les souscripteurs autorisés puissent déchiffrer la donnée.

Le chiffrement recherchable, ou searchable encryption, permet aux utilisateurs d'effectuer des recherches sur des données chiffrées. Cette propriété permet dans notre cas d'utilisation de masquer les topics attachés aux publications et aux souscriptions.

La signature électronique permet aux publieurs de garantir l'intégrité de leurs publications tout en s'authentifiant à travers ces dernières.

— Chapitre 4 : Nous présentons une contribution, notre solution de sécurité pour une architecture topic-based publish/subscribe à broker-relais, en prenant l'exemple du protocole MQTT. Nous détaillons dans un premier temps les différentes entités, leurs rôles, ainsi que la sécurité recherchée de notre solution et les différentes menaces sur l'architecture

publish/subscribe. Ensuite, nous présentons un état de l'art sur les différentes solutions de sécurité autour de cette architecture. Puis, nous détaillons le fonctionnement de notre solution ainsi que les modifications apportées au cryptosystème de chiffrement recherchable utilisé pour chiffrer les topic. Enfin, nous prouvons l'efficacité de notre solution via différentes expérimentations.

— Chapitre 5 : Nous présentons une autre contribution, notre solution de sécurité pour Apache Kafka, une architecture publish/subscribe avec un broker de stockage. Nous détaillons dans un premier temps le fonctionnement d'Apache Kafka ainsi que celui d'Apache Zookeeper, qui fonctionne de pair avec Apache Kafka. Ensuite, nous définissons les différentes entités, leurs rôles, ainsi que la sécurité recherchée de la solution et les différentes menaces sur l'architecture Apache Kafka. Puis, nous présentons un état de l'art sur les solutions de sécurité pour Apache Zookeeper, ainsi que sur le Private Information Retrieval (PIR) qui permet à un utilisateur d'effectuer des requêtes dans une base de donnée, sans que le serveur gérant cette dernière ne connaisse l'objet de ses requêtes. Enfin, nous détaillons notre solution de sécurité, et nous prouvons son efficacité via différentes expérimentations.

— Chapitre 6 : Nous présentons encore une contribution, une nouvelle construction de la primitive Attribute-Based Designated Verifier Signature (ABDVS). Cette primitive permet à un utilisateur de signer la donnée qu'il envoie, tout en sélectionnant, au travers d'une politique d'attributs, quels utilisateurs peuvent vérifier cette signature. Notre construction a pour but d'être suffisamment efficace pour être utilisable dans un scénario de santé, où un patient cherche à authentifier ses données tout en protégeant la vérification de ces dernières. L'objectif est que seuls les professionnels de santé choisis par le patient puissent effectuer la vérification de la signature.

— Chapitre 7 : Nous concluons cette thèse en résumant les différents travaux accomplis, ainsi qu'en présentant les différentes perspectives et futurs travaux pour poursuivre ces recherches.

Chapitre 2

Message-Oriented Middleware (MOM)

Sommaire

L'émergence des objets connectés dans notre quotidien a considérablement augmenté le nombre d'interlocuteurs sur les réseaux ainsi que les données disponibles [61]. Il est donc plus difficile pour un utilisateur recherchant une donnée, d'identifier et d'interagir avec l'interlocuteur adéquat. Cela amène donc les utilisateurs, qu'on peut qualifier de consommateurs de données, à utiliser d'autres moyens pour obtenir des informations.

L'architecture MOM répond à cette problématique. C'est un modèle de communication où la

donnée est au centre du processus. c'est-à-dire que les clients vont effectuer une recherche sur une donnée au travers d'un ou plusieurs intermédiaires, plutôt que de contacter un serveur spécifique. Ces données peuvent être émises par plusieurs utilisateurs, appelés producteurs de données.

Un des principaux avantages de ce modèle de communication est qu'il permet un couplage faible [37] entre les producteurs de données et les consommateurs. Cela signifie que ces interlocuteurs n'ont pas besoin de se connaître (c'est-à-dire de connaître leurs adresses IP respectives) pour communiquer et échanger des informations. L'intérêt principal de cette propriété est qu'ils n'interagissent pas directement entre eux pour échanger des informations.

2.1 Présentation générale

Une architecture MOM est organisée autour de deux types de composants [37] :

— Le **Provider** ou **Broker** : il s'agit du composant central de l'architecture, dont le rôle est de transmettre les messages entre les applications. Il sert de relais entre producteurs et consommateurs. Les producteurs envoient leurs données au broker, ces dernières étant attachées à un label, label que les consommateurs rechercheront pour consommer ces données. Il est courant de mettre en place une fédération de brokers afin de mieux gérer un grand nombre d'utilisateurs en équilibrant la charge de chacun.

— Les **Clients** : ce sont les applications qui échangent les messages via le broker. Une application cliente peut jouer successivement le rôle de producteur et celui de consommateur de données.

— Les **Messages** : composés d'un ou plusieurs éléments les caractérisant, qui peuvent être des champs ou un topic, ainsi que d'un contenu, qui peut être du texte, une image, une vidéo,etc.

Ainsi, ce type d'architecture permet aux consommateurs de rechercher une donnée sur la base de l'information qu'elle contient et non de l'émetteur de celle-ci. Cela permet dans une architecture regroupant de multiples objets connectés, de ne pas à avoir à les contacter individuellement mais plutôt de rechercher les données qu'ils produisent.

Cela permet également de gérer un grand nombre de producteurs de données. Puisque les

communications passent par plusieurs serveurs qui servent d'intermédiaires, le producteur n'a pas à initier une communication avec chaque consommateur intéressé par sa donnée. C'est pour cela que ce type de communication est utilisé dans des systèmes à grande échelle, comme par exemple les réseaux bancaires, les télécommunications, et la gestion de stocks [10].

L'architecture MOM permet également à des applications hétérogènes de pouvoir communiquer grâce au découplage entre les interlocuteurs. En effet, puisque les producteurs et consommateurs échangent des données de manière indirecte, le broker peut lisser et harmoniser ces données. Il peut par exemple collecter des données via différents protocoles réseaux (Bluetooth, Wi-Fi, etc.), et les distribuer via le protocole TCP/IP aux consommateurs. Cela permet à ces derniers de traiter les données plus efficacement, sans avoir à s'adapter aux technologies des producteurs.

Ce découplage permet également à ces interlocuteurs d'échanger des données sans qu'ils n'aient besoin d'être en ligne au même moment. On parle de communication asynchrone [105]. À noter que la plupart des solutions proposent les modes synchrones et asynchrones pour transmettre les messages. Cette flexibilité au niveau des communications facilite l'utilisation d'objets connectés pour transmettre et récupérer des données puisqu'ils sont souvent hors ligne, ou du moins changent d'état régulièrement.

Enfin, le mode de consommation des utilisateurs se focalise plus sur la donnée en elle-même que sur sa source. Cela s'explique en partie par la multiplication de diffuseurs de données, ce qui contribue à augmenter actuellement l'utilisation et l'adoption d'architectures de type MOM.

Comme mentionné dans l'introduction, il existe deux types de MOM : le point à point et le publish/subscribe. Nous présentons dans la section suivante leur fonctionnements ainsi que leurs principales différences.

2.2 Point à Point vs. Publish/Subscribe

Le point à point est un modèle de communication où les producteurs de données envoient des messages au travers d'une file de messages, également appelée « Queue », qui sera le plus souvent une file FIFO (First In, First Out). De l'autre côté, un consommateur pourra consommer le message, et acquittera sa bonne réception une fois qu'il l'aura consommé. Bien qu'il n'y ait aucune restriction sur le nombre de clients pouvant publier dans une file d'attente, il n'y a

généralement qu'un seul client consommateur, puisque chaque message n'est délivré qu'une seule fois à un seul destinataire. Dans le modèle point à point, les messages sont toujours livrés et sont stockés dans la file d'attente jusqu'à ce qu'un consommateur soit prêt à les récupérer.

Ce schéma s'inscrit dans le modèle de communication asynchrone, puisque producteurs et consommateurs n'ont pas besoin d'être en ligne en même temps pour échanger de l'information. Les messages ne sont envoyés qu'une seule fois et sont retirés de la file une fois consommés. Dans ce type d'architecture, une file est symbolisée par un identifiant, et un même serveur peut gérer plusieurs files de messages.

Un scénario utilisant l'architecture point à point est un système de messagerie, où l'adresse mail du consommateur est l'identifiant de la file, et où plusieurs utilisateurs peuvent déposer des messages qui seront consommés par le propriétaire de cette adresse mail. Cette dernière sert ainsi d'intermédiaire entre les producteurs et le consommateur. Le principal inconvénient de ce schéma de communication est qu'il ne permet pas, ou du moins pas facilement, de gérer plusieurs consommateurs sur les mêmes données.

Pour cela, on va préférer l'architecture publish/subscribe, qui est une architecture basée sur les événements, où la donnée transmise est appelée publication. Les producteurs de données sont appelés des publieurs et les consommateurs de données des souscripteurs. Les souscripteurs souscrivent à un abonnement sur le broker, et celui-ci transmettra toutes les nouvelles publications qui correspondront aux abonnements des souscripteurs.

On préfère donc utiliser l'architecture publish/subscribe à l'architecture point à point dans un contexte où plusieurs consommateurs consomment la même donnée. Les travaux présentés dans cette thèse se basent sur l'élaboration de solution de sécurité dans des contextes IoT, qui impliquent un grand nombre d'acteurs interagissant et échangeant des données. Cette thèse se focalise donc sur l'architecture publish/subscribe, qui présente de nombreux avantages dans ce contexte.

2.3 Publish/Subscribe

Présentation générale

Le publish/subscribe est une architecture de communication où des publieurs de données et des souscripteurs échangent des données via un ou plusieurs serveurs intermédiaires, appelés brokers [48]. Ces brokers permettent un découplage entre les publieurs et les souscripteurs. Les publieurs n'ont pas connaissance de qui sont les destinataires des données qu'ils envoient, et les souscripteurs ne connaissent pas la source des messages qu'ils reçoivent. Cela permet un système plus flexible, seulement limité par les capacités des serveurs intermédiaires. Ajouter et supprimer des utilisateurs est transparent pour l'architecture et ne perturbe pas son fonctionnement.

Les données envoyées par les publieurs au broker sont appelées publications. Ces publications sont définies par un identifiant ou un ensemble de champs accompagnés de valeurs, et d'un contenu. Les souscripteurs notifient leurs intérêts au broker avec une souscription. Une souscription illustre les intérêts des souscripteurs au travers d'un identifiant unique ou d'un ensemble de contraintes sur les champs des publications, qu'on appelle des filtres.

Le broker est responsable de transmettre les publications qu'il reçoit aux souscripteurs. Une fois la nouvelle publication reçue, le broker va parcourir sa table de diffusion afin d'effectuer la transmission vers les souscripteurs concernés (c'est-à-dire qui ont envoyé préalablement une souscription correspondant à l'identifiant/ensemble de champs de la publication reçue).

Les publications sont le plus souvent éphémères, et ne sont pas conservées sur le broker, même si des implémentations comme Kafka [95] permettent de conserver les données sur les brokers pendant une durée prédéfinie. Ce mécanisme permet de pallier la réception de très grosses quantités de données qu'il n'est pas possible de traiter en temps réel.

Apache Kafka utilise donc le publish/subscribe pour être une plateforme de streaming d'évènements [64]. Dans ce contexte, les données sont envoyées aux brokers sous formes de séquences de tuples d'évènements. Dans ce type d'architecture, les données sont conservées par les différents brokers, qui attendent les requêtes en lecture des différents consommateurs. Cela permet à ces brokers d'être plus adaptés pour gérer de gros volumes de données. Les caractéristiques d'Apache Kafka sont détaillées en section 5.1.1. On distingue donc les brokers relais des brokers de stockage. Cette thèse propose des solutions de sécurité pour ces deux types de brokers, présentées

respectivement aux chapitres 4 et 5.

Les catégories de publish/subscribe se distinguent par la manière dont sont formées les souscriptions. Ces dernières peuvent être formées en fonction du nom du topic recherché ou du contenu des publications : on parle respectivement de topic-based ou content-based. Le publish/subscribe topic-based définira la donnée par un identifiant unique, qui peut être un arbre hiérarchique et organisé, tandis que le publish/subscribe content-based définira la donnée avec un ensemble de champs et de valeurs qui définissent le contenu. Nous détaillons ci-dessous ces deux types de publish/subscribe afin de déterminer leurs principaux avantages et inconvénients, et de choisir lequel s'adapte le mieux à un scénario de santé avec beaucoup d'utilisateurs.

2.3.1 Content-based

Dans une architecture publish/subscribe content-based, la donnée est définie avec différents champs ayant la forme [attribut,valeur] [69]. Dans un contexte de gestion de stocks, la publication d'une fiche produit peut être caractérisé par les champs "[MARQUE,ABC], [PRIX,30€], [TYPE,ORDINATEUR]".

Ces champs permettent d'effectuer des souscriptions en se basant sur le contenu des publications. Les utilisateurs envoient leurs souscriptions sous la forme de filtres, représentés par plusieurs contraintes sur les différents champs qui caractérisent la donnée. Cette définition de la donnée permet aux souscripteurs d'être plus précis dans leurs demandes, et convient bien aux applications à grande échelle, comme les gestions de stocks [12] ou les villes intelligentes [102], qui réunissent un grand nombre d'acteurs différents.

La principale caractéristique de l'architecture content-based est qu'elle permet aux utilisateurs de soumettre des filtres très expressifs qui sont définis comme des formules booléennes contenant des conjonctions et des disjonctions de prédicats. Un prédicat définit une contrainte sur un attribut d'événement. Les contraintes sont définies à l'aide d'opérateurs tels que $=$, $<$, $>$, $\leq$ et $\geq$. Par exemple, un souscripteur envoyant le filtre "MARQUE=ABC and PRIX$\geq$25€" au broker recevrait toutes les publications concernant les stocks de la marque ABC d'un prix supérieur ou égal à 25€.

Il existe plusieurs solutions de publish/subscribe content-based, dont les plus connues sont

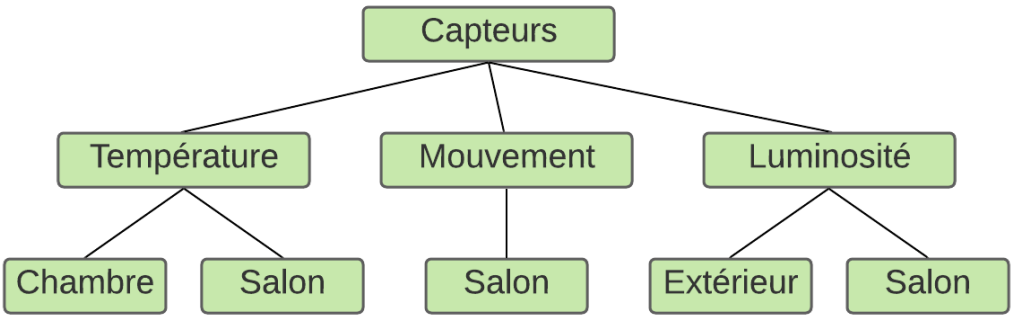


FIGURE 2.1 – Topic-Based Hiérarchique

Gryphon [9], SIENA [26] et PADRES [72]. Ces trois solutions sont basées sur un réseau de brokers qui échangent des données afin d'équilibrer leur charge de travail. Gryphon est une des premières solutions de content-based publish/subscribe, où différents brokers sont dédiés à effectuer la correspondance entre les publications et souscriptions, tout comme SIENA. PADRES se base sur une architecture totalement décentralisée, où chaque broker participe au processus de correspondance.

Le content-based publish/subscribe permet aux utilisateurs de définir leurs publications avec plusieurs champs [attribut,valeur] et leurs souscriptions avec un ensemble de filtres. Il se distingue du topic-based publish/subscribe où les publications et souscriptions sont définies avec un label unique, appelé topic.

2.3.2 Topic-based

L'architecture topic-based publish/subscribe [70] définit une publication par un topic et son contenu. Le topic est en général représenté par une chaîne de caractère servant d'identifiant à la publication. Le contenu est lui de taille variable et peut être une image, un texte, un son, une vidéo. Les souscriptions sont également définies par un topic.

Cette architecture est similaire à l'idée d'un groupe de communication. Ainsi, dès qu'un souscripteur s'abonne à un topic T, il est considéré comme membre du groupe T. Lorsqu'une nouvelle donnée est publiée sur ce topic, elle est diffusée à tous les membres du groupe.

Il existe également des implémentations d'architecture topic-based publish/subscribe qui permettent aux utilisateurs d'organiser les topics de manière hiérarchique, comme illustré en Figure 2.1. Pour reprendre cet exemple, un utilisateur peut ainsi s'abonner au topic "Cap-

teurs/Luminosité/" et recevoir les publications publiées sur les topics "Capteurs/Luminosité/Salon" et "Capteurs/Luminosité/Extérieur".

Une itération de ce mécanisme est le système de wildcards - signes génériques - qui permettent d'effectuer des souscriptions plus expressives, en acceptant des variations dans le nom des topics. Si on reprend l'exemple de la Figure 2.1, ainsi que les wildcards utilisées dans l'implémentation topic-based MQTT [11], un souscripteur peut envoyer au broker la souscription "Capteurs/Température/#" afin de s'abonner aux topics "Capteurs/Température/Chambre" et "Capteurs/Température/Salon", et effectuer la souscription au topic "Capteurs/+/Salon" pour s'abonner aux topics "Capteurs/Température/Salon", "Capteurs/Mouvement/Salon" et "Capteurs/Luminosité/Salon". Ici, le sigle # est utilisé pour indiquer accepter toutes les variations en fin de chaîne, tandis que le sigle + permet une variation au milieu de la chaîne. Ces outils permettent une meilleure expressivité dans les souscriptions et ainsi une gestion plus fine des communications.

Un autre avantage de l'architecture topic-based est que deux publieurs peuvent poster dans le même topic. Cela permet de réduire la taille des tables de diffusion et ainsi réduire la latence lors du routage d'information.

Il existe donc de nombreuses implémentations de topic-based publish/subscribe, comme le protocole MQTT [11], précédemment cité, qui est un standard OASIS [125]. RabbitMQ [97] est également populaire et implémente un mode pour faire du point à point. Une autre implémentation déjà citée, utilisée dans l'industrie dans des applications comme LinkedIn, Twitter, Uber, Netflix [95] est Apache Kafka.

Les architectures publish/subscribe topic-based ont donc l'avantage de pouvoir être déployées rapidement, à grande échelle, et avec une grande flexibilité. Cela en fait le type de publish/subscribe le plus utilisé aujourd'hui dans le monde industriel et chez les particuliers.

2.3.3 Topic-based vs Content-based

Même si les architectures content-based publish/subscribe présentent un certain nombre d'avantages, en particulier au niveau de la sélection de contenu par les souscripteurs, elle ne sont pas autant utilisées que les architectures topic-based. Cela s'explique pour plusieurs rai-

sons ; premièrement, la mise en place de ce type d'architecture demande aux publieurs, qui sont dans notre cas d'utilisation des objets connectés, d'envoyer beaucoup d'informations les concernant, afin que les brokers puissent organiser les attributs associés à ce type de données.

De plus, ces données doivent garder une certaine cohérence pour que leur utilisation ne soit pas trop laborieuse pour les consommateurs. En effet, si les couples [attribut,valeur] sont formés différemment en fonction du fabriquant de l'objet connecté ou de son année de production, ou si les unités de mesures sont différentes, il est difficile d'effectuer des souscriptions renvoyant des résultats satisfaisants. Prenons l'exemple de deux podomètres renvoyant les activités physiques d'un patient. Si l'un définit cette mesure en kilomètre et l'autre en miles, on peut distinguer deux cas : soit le système a défini à priori deux champs distincts pour ces mesures [DISTANCE_KM = 2.2] et [DISTANCE_MI = 1.5], et le souscripteur n'a qu'à former sa souscription de manière à prendre en compte ces deux possibilités. Soit ces valeurs sont indistinctement placées dans l'attribut DISTANCE, et il n'est même pas possible pour le souscripteur de soumettre un filtre de contrainte sur ce champ. Utiliser une architecture content-based dans un scénario réel demande donc une configuration très précise et minutieuse, qui demande souvent à être vérifiée/changée lors de l'ajout de nouveaux publieurs.

Un autre point clé de l'architecture content-based publish/subscribe est le processus de correspondance entre publications et souscriptions. Ce processus doit traiter différentes contraintes pour chaque souscription, et l'efficacité de cette phase représente un point clé de l'infrastructure. De nombreux travaux ont été effectués pour répondre à cette problématique comme [109], [49] et [24]. Même si ces travaux permettent d'obtenir des temps de réponse satisfaisants, ces derniers ne peuvent être aussi efficaces que la vérification de la correspondance entre deux topics.

Avec un très grand nombre d'utilisateurs, le broker va donc mettre de plus en plus de temps à effectuer les transmissions à chaque nouvelle publication. Même si les souscripteurs ont l'avantage de pouvoir effectuer des souscriptions de plus en plus précises, le nombre de souscriptions et de notifications maintenues dans les tableaux de correspondances du broker ne va donc qu'augmenter, et ainsi produire un délai de transmission qui peut, pour certaines mesures, rendre l'information reçue par le souscripteur obsolète.

Enfin, ajouter une couche de sécurité dans une architecture content-based publish/subscribe au niveau des champs caractérisant la donnée s'avère très coûteux. En effet, pour chaque pu-

blication, le broker doit analyser les différentes contraintes des souscriptions qu'il stocke. Or, effectuer ces comparaisons avec des données chiffrées est très coûteux. En effet, il faut utiliser des schémas comme le chiffrement entièrement homomorphe - Fully Homomorphic Encryption (FHE) [58], [134].

Ces schémas présentent encore des problèmes d'efficacité ce qui ne leur permet pas d'être utilisés dans des solutions réalistes [1]. Le topic-based publish/subscribe a l'avantage de ne nécessiter que la vérification d'une égalité, ce qui est moins coûteux [106].

Un tableau récapitulatif de la comparaison entre l'architecture publish/subscribe topic-based et content-based est présenté en Figure 2.1. Les travaux présentés dans cette thèse se baseront donc sur une architecture topic-based publish/subscribe, ainsi que sur l'implémentation de solutions de sécurité dans des applications de santé rassemblant un grand nombre d'utilisateurs.

2.4 Scénarios d'applications Publish/Subscribe

Le publish/subscribe est le modèle de communication le plus adapté aux architectures utilisant un grand nombre d'objets connectés. Pour illustrer son utilisation, et ainsi démontrer les enjeux de sécurité que représente ce modèle de communication, deux scénarios autour du domaine de la santé sont présentés.

Tout d'abord, nous prenons l'exemple de la gestion des données personnelles de patients dans un projet d'assistance à l'autonomie à domicile - Ambient Assisted Living (AAL). Cet environnement utilise plusieurs objets connectés, qui peuvent être par exemple des capteurs portables connectés au patient (mesure de la fréquence cardiaque, température cutanée) [87] ou des capteurs ambiants (capteur de lumière, détecteur de chute) [133]. Les données ainsi collectées sont traitées et analysées par plusieurs organismes et professionnels différents (professionnels de santé, diététiciens, rééducateurs, statisticiens, etc.).

Ensuite, nous prenons un scénario à plus grande échelle, avec une plus grande quantité de données à gérer : la gestion des systèmes d'information de plusieurs établissements de santé. Cet ensemble d'établissements de santé est appelé un Groupement Hospitalier de Territoire (GHT) [89]. Dans un GHT, les établissement de santé partagent les données de leurs systèmes d'information avec les autres établissements. Plus précisément, les données collectées par les

Phase	**Topic-based**	**Content-based**
Publications	Organisées en topics	Organisées par rapport aux propriétés des publications
Souscripteur	Reçoit toutes les publications des topics auxquels il est abonné	Définit un ensemble de filtres, et seules les publications correspondant **exactement** à l'ensemble de ses filtres lui sont transmises
Publieur	Définit sa publication par un topic	Définit sa publication par un ensemble de tuple [attribut,valeur]
Communication	Pattern de routage bien défini d'un publieur aux souscripteurs abonnés	Les publications sont transmises en fonction des correspondances entre les différents filtres et les attributs de la publication
Avantages	Implémentation simple et rapide puisque l'ajout de topic/utilisateurs se fait de manière dynamique sans perturber le système. Le routage d'un publieur vers un groupe de souscripteurs est efficace	Les souscripteurs peuvent être expressifs avec leurs souscriptions de manière à limiter l'envoi superflu de données
Inconvénients	Les souscriptions ne peuvent être expressive qu'avec l'usage de wildcards/topics hiérarchiques. Un souscripteur s'abonnant à un topic reçoit toutes les données de celui-ci, même s'il n'est intéressé que par une partie	Le routage demande des calculs supplémentaires qui peuvent être problématique dans un scénario à grande échelle. Demande une implémentation complexe et rigoureuse pour être utilisé
Implémentations	MQTT [11], RabbitMQ [97], Apache Kafka [95]	Gryphon [9], Siena [26], PADRES [72]

TABLE 2.1 – Topic-based vs content-based publish/subscribe

établissements de santé sont envoyées à un ensemble de serveurs, qui ont pour rôle de les traiter et les transmettre vers différentes bases de données partagées. La principale problématique de cette architecture est la gestion d'un flux important de données, qui ne peut être assurée par une architecture classique comme pour le scénario précédent.

Ces deux scénarios permettront de mettre en lumière plusieurs problématiques de sécurité, et serviront d'appui pour l'élaboration de solutions à celles-ci.

2.4.1 Ambient Assisted Living (AAL)

L'assistance à l'autonomie à domicile - Ambient Assisted Living (AAL) est un domaine multidisciplinaire visant à fournir un écosystème de différents types de capteurs, ordinateurs, appareils mobiles, réseaux sans fil et applications pour maintenir et favoriser l'autonomie des personnes âgées et ainsi augmenter la sécurité dans leur mode de vie et dans leur habitat [42], [83].

L'AAL repose sur un ensemble d'objets connectés, qui sont majoritairement des capteurs ambiants et des capteurs portables [114]. Ils permettent de faciliter les soins médicaux à domicile, avec de la télémédecine par exemple, ainsi qu'un meilleur suivi des traitement prescrits par les médecins [126]. Les données collectées peuvent être utilisées par des organismes différents.

Ces environnements sont souvent destinés aux personnes âgées, qui sont des patients requérant un bon télésuivi des constantes biologiques. Ainsi, les personnes âgées se déplacent moins souvent dans les établissements de santé et une part importante de leur recours aux soins se fait via des visites des professionnels de santé à leur domicile.

De nombreux projets de maisons intelligentes pour l'AAL ont été proposés, et sont d'ailleurs récapitulés dans l'étude [114]. Ces maisons peuvent être adaptées en fonction des besoins de leurs habitants, avec par exemple le projet "Aging in Place" [112] de l'université du Missouri, et le projet CASAS [113] de l'université d'état de Washington. Le premier a pour but d'apporter un soutien aux personnes âgées, tandis que le deuxième permet d'assister les patients atteints de démence à leur domicile.

Nous récapitulons les différents acteurs et leurs interactions en Figure 2.2. Nous définissons ces acteurs de la manière suivante :

— Les capteurs portés par le patient et les capteurs ambiants ; ils produisent un grand nombre

de données relatives à la santé de la personne, à sa sécurité et à ses habitudes.

— La passerelle sur laquelle vont être envoyées toutes les données récoltées dans l'habitat ; elle se situe chez le patient et transmettra toutes les données au broker. Elle représente l'infrastructure informatique, souvent un Raspberry Pi ou un ordinateur [83], qui permet de collecter les données dans la maison.

— Les professionnels intervenant à domicile ; ce sont les professionnels supervisant le patient qui ont besoin d'accéder aux données de ce dernier afin de l'accompagner au mieux. Ces professionnels n'utiliseront souvent pas toutes les données produites mais seulement celles correspondant à leurs interventions. On peut lister les infirmiers, le médecin traitant, les médecins spécialistes, les aides soignants, les professionnels du secteur social, etc.

— Analystes de statistiques de la vie quotidienne : effectuer des analyses sur les habitudes des patients utilisant ces environnements d'aide à l'autonomie à domicile peut s'avérer très utile pour détecter des anomalies et des soucis de santé [60].

— Serveur (Broker) : il sert d'intermédiaire entre les données recueillies chez le patient et les professionnels souhaitant y accéder. Il doit être accessible aussi bien chez le patient qu'à l'extérieur.

Dans ce type d'architecture, il est important de gérer les accès aux données de ces différents organismes, afin que l'utilisateur ait un contrôle sur ses données personnelles de santé. En effet, en fonction des professionnels intervenants, les données nécessaires ne sont pas les mêmes et il faut garantir que la vie privée de la personne n'est pas exposée plus que nécessaire. Cette propriété est d'autant plus importante qu'une personne est plus disposée à utiliser ces technologies si elle sait que ses données seront protégées.

Comme illustré en Figure 2.2, le broker transmet donc les données produites par les différents capteurs vers les professionnels intéressés. Plus précisément, un ensemble de publieurs de données vont centraliser leurs données sur la passerelle disposée dans l'habitat. Cette passerelle relayera ensuite ces informations au broker, qui les transmettra vers l'ensemble des professionnels concernés par ces données.

Les données envoyées sont donc des publications, avec pour chacune un topic qui indique le type de capteur et l'endroit où il est utilisé, ainsi qu'un contenu qui contient la date d'envoi et

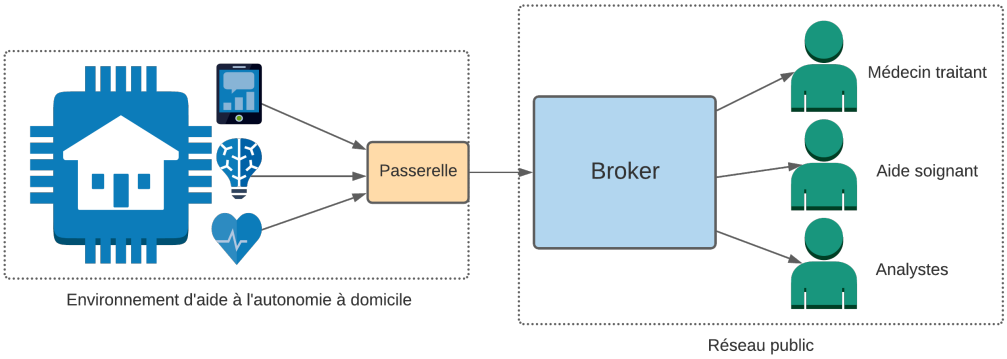


FIGURE 2.2 – Application publish/subscribe dans un environnement d'aide à l'autonomie à domicile

les données recueillies.

Les divers professionnels prenant en charge le patient effectuent des souscriptions qui illustrent aux mieux les données dont ils ont besoin pour la prise en charge du patient. Le médecin traitant peut ainsi s'informer sur les constantes biologiques du patient via des capteurs, et les analystes peuvent utiliser les données relatives à la vie quotidienne du patient pour définir ses habitudes.

Le broker a donc un rôle essentiel étant donné qu'il centralise l'ensemble des données transmises. Cela en fait également un acteur très exposé puisque de nombreux acteurs différents communiquent avec lui via des publications et des souscriptions. Il relaye les informations de l'AAL vers les professionnels de santé.

Nous avons présenté dans ce scénario un cas d'usage où les données collectées dans un environnement AAL sont transmises à des professionnels de santé via un broker relais. Or, lorsque l'on passe à un scénario à plus grande échelle, avec par exemple plusieurs établissements de santé, la quantité de données à transmettre par le broker est bien plus conséquente. C'est pourquoi nous étudions également ce type de scénario avec un grand nombre d'utilisateurs, où un broker relais ne suffit pas.

2.4.2 Groupement Hospitalier de Territoire (GHT)

Les établissements de santé en France connaissent un certain nombre de bouleversements sur leurs systèmes d'information, principalement dû au processus de numérisation globale des données ainsi qu'à l'externalistation de ces dernières [92]. Depuis la loi de modernisation du

système de santé du 26 janvier 2016 qui a pour but d'organiser les centres hospitaliers en groupements hospitaliers de territoire (GHT), 136 GHT se sont formés, chacun regroupant de 2 à 20 établissements pour 100 000 à 2,5 millions d'habitants [96]. Au centre de ce projet s'inscrit le dossier patient unique [89]. Pour un même GHT, chaque dossier patient est partagé entre tous les établissements afin de faciliter l'accès aux examens déjà réalisés, à leurs résultats, et aux antécédents du patient. Ainsi, chaque GHT met en place un Hébergement de Données de Santé (HDS) afin de mutualiser ces dossiers patients.

Ainsi, parmi l'ensemble d'établissements faisant parti d'un GHT, un établissement dénommé « établissement support » se voit responsable de la gestion d'un système d'information hospitalier et d'un département d'information médicale de territoire.

Au niveau de l'architecture réseau, chaque établissement de santé gère un certain nombre de données via son propre système d'information ; capteurs, résultats d'examens et diagnostics des personnels soignants. Ces données, attachées au dossier patient, sont externalisées afin de les mettre à profit des autres membres du GHT. Étant donné le gros volume de données à traiter, on considère qu'un broker de stockage sert de lien entre les différents systèmes d'information et l'hébergement des données de santé.

On peut ainsi définir les acteurs suivants :

— Les patients : usagers des établissements de santé, ce sont leurs données personnelles de santé qui sont utilisées par les professionnels de santé afin d'effectuer des diagnostics et prodiguer des soins.

— Les professionnels de santé : médecins, infirmiers, ambulanciers, qui ont besoin de pouvoir consulter les données des patients, mais ne disposent pas tous du même degré d'accréditation.

— Le système d'information interne de l'établissement de santé : c'est le système qui gère le traitement des données à travers l'établissement. Il centralise ainsi les données produites par les divers capteurs de l'établissement ainsi que les examens et diagnostics qui y ont lieu.

— Le broker : il a pour rôle de faire le lien entre les données qui sont envoyées par les différents établissements de santé et le HDS cloud contenant les bases de données partagées. Il peut

y avoir plusieurs brokers qui coopèrent pour gérer plusieurs systèmes d'information.

— L'hébergeur des données de santé : le HDS Cloud centralise les données de santé produites par les établissements de santé de manière à ce qu'elles soient à disposition de tous les professionnels de santé du GHT concernés. Ces données doivent également rester disponibles pour les patients, qui ont un droit de regard sur leurs données [90].

Dans ce scénario, nous avons donc les publieurs qui sont les différents systèmes d'information des établissements de santé, et indirectement l'ensemble des objets connectés qui y sont liés ainsi que les professionnels de santé. Les souscripteurs sont les différents serveurs de l'hébergement des données de santé, qui sont les bases de données partagées entre les différents établissements.

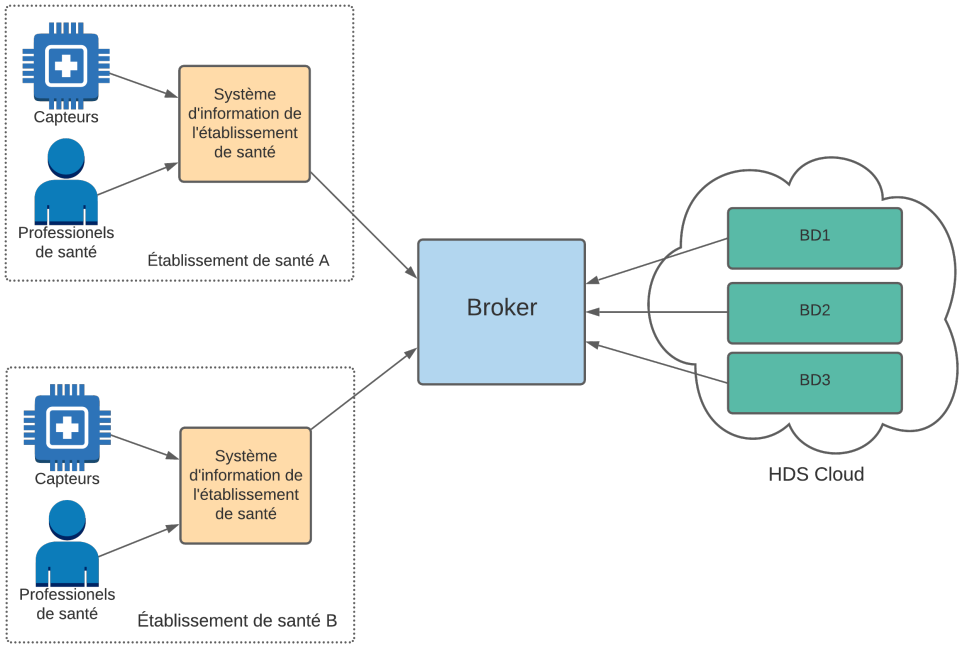


FIGURE 2.3 – Application publish/subscribe pour un GHT

Nous illustrons les principales entités ainsi que leurs interactions en Figure 2.3. Les données sont remontées du système d'information de l'établissement de santé vers le broker. Il a pour rôle de collecter les données transmises par les différents établissements de santé et de les distribuer vers les différentes bases de données de HDS Cloud. La principale différence, par rapport au scénario précédent, est que le broker doit gérer un très gros volume de données ; c'est pourquoi on préférera utiliser un broker de stockage plutôt qu'un broker relais.

Dans les deux scénarios présentés, les données personnelles des patients sont externalisées

et partagées entre différents acteurs. C'est pourquoi ces données doivent être protégées par des mécanismes de sécurité robustes. Nous prenons également en considération que ces données nécessitent parfois d'être consultées en urgence pour des interventions, et que ces mécanismes de sécurité ne doivent pas perturber leurs disponibilités.

2.5 Approche globale de la sécurité des scénarios

Dans ces deux scénarios, les données personnelles des patients sont externalisées hors du réseau vers un ou plusieurs serveurs afin d'être diffusées vers des professionnels de santé distants. Ces serveurs sont donc exposés à de nombreux acteurs potentiellement malveillants, et relayent la totalité des informations personnelles des utilisateurs.

En prenant en compte leur exposition ainsi que les données auxuquelles ils peuvent accéder, on considère ces serveurs suivant le modèle de sécurité *honest-but-curious*, c'est-à-dire qu'ils effectuent honnêtement leur travail sans compromettre les données, mais cherchent à en apprendre le plus possible sur ces dernières. Le but des solutions de sécurité que nous allons élaborer est donc de minimiser les connaissances des serveurs sur les données qu'ils manipulent, tout en réduisant les risques liés aux interceptions de données.

Une première solution est donc de chiffrer le contenu des publications du publieur au souscripteur, ce qui permet d'éviter que le broker ait accès à ces données. Ce n'est cependant pas suffisant, il est également important de rendre confidentiel les intérêts (les souscriptions) des souscripteurs, ainsi que le topic caractérisant la donnée lors des publications. De plus, dans des scénarios de santé, le contenu ainsi que le topic des publications qui transitent via le broker sont extrêmement sensibles. Si une personne mal intentionnée cherche à connaître la situation médicale d'un patient, il est important que toutes ces informations restent chiffrées lorsqu'elles sont sur un réseau public.

Afin d'illustrer ce cas d'usage, prenons l'exemple de la publication d'une mise à jour d'un dossier patient contenant les résultats d'analyse d'une coronarographie, publiée dans le topic :

— "Service/Cardiologie/Patient/Martin_Jean/"

Ici, les résultats de l'analyse sont des données personnelles sensibles qui doivent être chiffrées lorsqu'elles sont hébérgées sur le serveur, pour que seul les personnels soignants autorisés puissent

y accéder. De plus, le nom du topic sur lequel les données sont publiées indique qu'il a pu faire des analyses via le service de cardiologie, il doit donc être chiffré également.

Un autre aspect de la sécurité d'une telle architecture repose sur l'authentification et l'intégrité des données traitées. En effet, ce sont des informations relatives à l'état de santé des patients et celles-ci ne doivent pas être corrompues, falsifiées, ou envoyées par une personne qui n'est pas un professionnel de santé, ou qui n'est pas authentifié comme un des capteurs du patient. Si de telles conditions ne sont pas respectées, le patient ne peut pas être pris en charge de manière satisfaisante, puisque cela peut entraîner des délais voir des traitements médicaux qui ne correspondent pas à sa situation, et donc menacer sa santé.

Puisque la plupart des architectures publish/subscribe n'intègrent pas par défaut de mécanisme d'authentification entre le publieur et le souscripteur, il est nécessaire de mettre en place un schéma permettant l'authentification du publieur et qui vérifie l'intégrité de la donnée envoyée.

Une dernière propriété nécessaire dans ce type d'architecture est la possibilité de révoquer un utilisateur. En effet, dans le domaine de la santé, cela peut représenter un point critique pour la sécurité des données des patients si un professionnel de santé, qui n'est plus autorisé à exercer son métier, conserve les droits d'accès aux données. C'est une propriété à prendre en compte lors de l'élaboration d'une solution de sécurité.

Pour finir, lors de l'élaboration de ces solutions, il est nécessaire de conserver un couplage faible entre les publieurs et les souscripteurs. Il n'est pas réaliste pour un professionnel de santé de devoir contacter chaque capteur pour récupérer les informations qui l'intéresse. Ce serait évidemment long et laborieux et cela pourrait entraîner un délai quand au traitement du patient, et donc des conséquences assez grave s'il doit être traité en urgence.

2.6 Conclusion du chapitre

Dans ce chapitre, nous avons présenté l'approche MOM ainsi que les différentes variations qui la compose, le publish/subscribe et le point à point. Après avoir présenté les avantages et les inconvénients de chacun pour élaborer une solution de sécurité pour des objets connectés dans un environnement de santé, nous avons décidé de travailler sur l'architecture publish/subscribe.

Cette dernière se divise en deux sous-catégories, le topic-based et le content-based. Nous

avons choisi d'utiliser l'architecture topic-based, qui présente des solutions permettant une mise à l'échelle plus simple, ainsi qu'une plus grande flexibilité. De plus, elle permet la mise en place de solutions de sécurité qui ne risquent pas de compromettre l'accessibilité aux données avec des temps de traitement trop longs.

Afin de mieux présenter nos solutions de sécurité, nous avons détaillé deux scénarios de santé ayant des particularités différentes au niveau de l'accès à la donnée. Ces deux scénarios ont été choisis pour différencier deux types de broker différents, les brokers relais (MQTT) et les brokers de stockage (Apache Kafka). MQTT convient à l'environnement AAL, où les données sont routées vers différents organismes qui restent connectés au broker de manière continue. D'un autre côté, Apache Kafka convient mieux à l'utilisation dans un établissement de santé, où les données peuvent être consommées plusieurs fois pendant le séjour du patient par plusieurs utilisateurs différents à des moments différents.

Pour finir, nous avons présenté notre approche sur la sécurité que demande une architecture publish/subscribe dans un scénario de santé. Cette approche sert de point de départ pour l'élaboration de nos solutions, ainsi que pour le choix des schémas cryptographiques utilisés.

Chapitre 3

Définitions

Dans ce chapitre, nous présentons différentes primitives cryptographiques qui permettent d'assurer confidentialité et protection de la vie privée pour les utilisateurs d'une architecture publish/subscribe. Le but de ce chapitre est d'introduire les principaux mécanismes cryptographiques que nous utiliserons dans la suite de nos contributions.

Sommaire

3.1 Confidentialité des données : Chiffrement

Les schémas de chiffrement permettent de protéger les données en les rendant illisibles pour les utilisateurs non autorisés à les consulter. Pour cela, une clé est utilisée pour chiffrer le message et le déchiffrer, et sa possession symbolise le fait de pouvoir accéder aux données.

3.1.1 Cryptographie symétrique

La cryptographie symétrique, également appelée chiffrement à clé secrète, est la première forme de chiffrement de l'histoire. Dans les schémas de cryptographie symétrique, les deux utilisateurs échangent au préalable un secret, appelé clé, via un protocole.

Cette clé leur permet aussi bien d'effectuer l'opération de chiffrement que de déchiffrement. Les premiers schémas de chiffrement symétriques sont apparus dès l'antiquité, le plus populaire étant le chiffrement de César. En 1586, Blaise de Vigenère décrit un schéma de chiffrement plus complexe qu'on appellera a posteriori le chiffrement de Vigenère. Ces schémas font partie de la famille des chiffrements à substitution, qui consistent à substituer chacune des lettres d'un message par une autre.

Les schémas symétriques plus récents sont des chiffrements par blocs. Ces algorithmes prennent une suite de n bits en entrée, pour produire un chiffré de n bits. Ces algorithmes effectuent un ensemble de transformations sur le clair et utilisent pour cela une clé de longueur k. On retient l'algorithme Data Encryption Standard (DES) [107] adopté en tant que standard en 1976, mais devenu obsolète aujourd'hui. Au début des années 2000, le schéma Advanced Encryption Standard (AES) [44] devient le nouveau standard après avoir remporté le concours du NIST débuté en 1997. Il utilise des clés de 128, 192 ou 256 bits, et chiffre par blocs de 128 bits. Le schéma de chiffrement AES est composé de trois algorithmes :

— $\mathsf{KeyGen}(n) \rightarrow k$: Étant donné un nombre de bits n, l'algorithme KeyGen génère une clé secrète AES k de n bits.

— $\mathsf{Encrypt}(m, k) \rightarrow c$: Étant donné un message m et une clé AES k, l'algorithme $\mathsf{Encrypt}$ génère le chiffré c.

— $\mathsf{Decrypt}(c, k) \rightarrow m$: Étant donné un chiffré c et une clé AES k, l'algorithme $\mathsf{Decrypt}$ déchiffre le chiffré c afin de récupérer le message m.

Les schémas de cryptographie symétriques sont souvent associés aux schémas de cryptographie asymétriques, ces derniers permettant d'échanger une clé secrète lors de la première communication entre les interlocuteurs.

3.1.2 Cryptographie asymétrique classique

La cryptographie asymétrique, également appelée cryptographie à clé publique, est un concept énoncé pour la première fois publiquement dans un article de 1976 [41] publié par les cryptologues Diffie et Hellman. Ils y présentent le schéma d'échange de clés Diffie-Hellman, qui permet à deux utilisateurs d'échanger une clé dans un canal publique sans qu'une tierce personne écoutant la totalité de la conversation ne puisse la reconstruire. Pour cela, deux clés sont maintenues par le receveur : une clé publique et une clé privée. La première sert à chiffrer des données et est accessible à tous tandis que la deuxième est conservée secrète par le receveur et lui permet de déchiffrer les données chiffrées avec sa clé publique.

Un schéma de chiffrement à clé publique est composé de quatre algorithmes :

— $\mathsf{Setup}(\mathfrak{K}) \rightarrow param$: Étant donné un paramètre de sécurité $\mathfrak{K}$, l'algorithme Setup génère les paramètres $param$.

— $\mathsf{KeyGen}(param) \rightarrow ek, dk$: Étant donné les paramètres $param$, l'algorithme KeyGen génère une paire de clés (ek,dk) avec ek la clé publique de chiffrement et dk la clé secrète de déchiffrement.

— $\mathsf{Encrypt}(m, ek) \rightarrow c$: Étant donné un message m et une clé publique de chiffrement ek, l'algorithme $\mathsf{Encrypt}$ génère le chiffré c.

— $\mathsf{Decrypt}(c, dk) \rightarrow m$: Étant donné un chiffré c et une clé de déchiffrement dk, l'algorithme $\mathsf{Decrypt}$ déchiffre le chiffré c afin de récupérer le message m.

Comme illustré en Figure 3.1, Bob met à disposition sa clé publique, ce qui permet à Alice de chiffrer les données qu'elle souhaite lui envoyer. Une fois le message chiffré reçu, Bob utilise sa clé privée pour déchiffrer le message.

Il faut attendre 1978 pour que les cryptologues Rivest, Shamir et Adleman proposent le premier schéma de chiffrement asymétrique, baptisé RSA [116]. Encore utilisé aujourd'hui, la

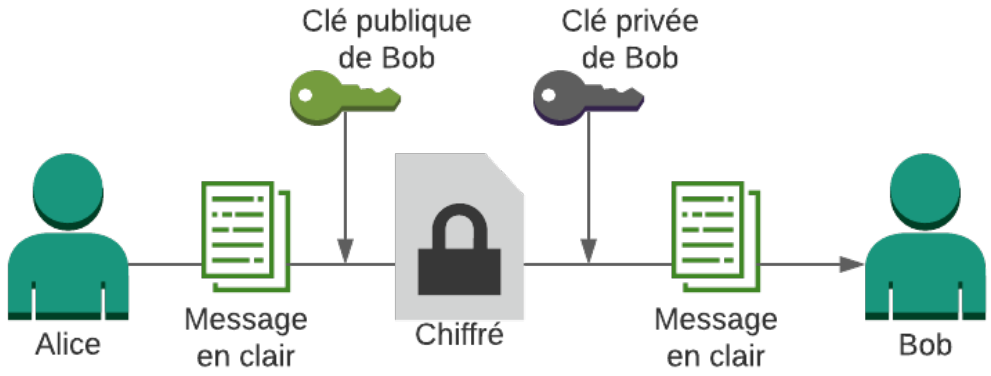


FIGURE 3.1 – Chiffrement asymétrique

sécurité de ce schéma est basé sur le problème difficile de la factorisation du produit de deux grands nombres premiers.

De nombreux autres cryptosystèmes sont apparus ensuite, comme le chiffrement El Gamal en 1984 [45], sa sécurité étant basée sur le problème du logarithme discret.

D'autres schémas ont apporté des propriétés différentes, afin de faciliter les communications et la gestion des clés dans des scénarios mettant en œuvre plus de deux interlocuteurs.

C'est le cas du chiffrement basé sur l'identité, Identity-Based Encryption (IBE), introduit par Shamir en 1984 [120]. Cette primitive cryptographique permet aux utilisateurs de ne pas avoir à partager leur clé publique et à conserver la clé publique des autres utilisateurs du système. En effet, ce schéma utilise un couple de clés maîtresses (c'est-à-dire une clé publique et une clé secrète), conservées par une autorité tierce. La clé maîtresse publique permet à n'importe quel utilisateur de chiffrer un message en fonction d'une identité qu'il va choisir au moment du chiffrement, sous la forme par exemple d'une adresse mail. La clé maîtresse secrète permet à l'autorité de générer les clés secrètes des utilisateurs en fonction de leur identité. Le fonctionnement de ce schéma de chiffrement est résumé en Figure 3.2.

Dans un schéma IBE, l'avantage est qu'il n'est pas nécessaire d'avoir une infrastructure gérant les clés publiques des utilisateurs, il suffit d'une autorité de confiance qui maintienne les clés maîtresses.

Une généralisation de ce schéma, le schéma Attribute-Based Encryption (ABE), permet de chiffrer un message pour plusieurs utilisateurs, qui sont caractérisés non pas par une identité mais par des attributs.

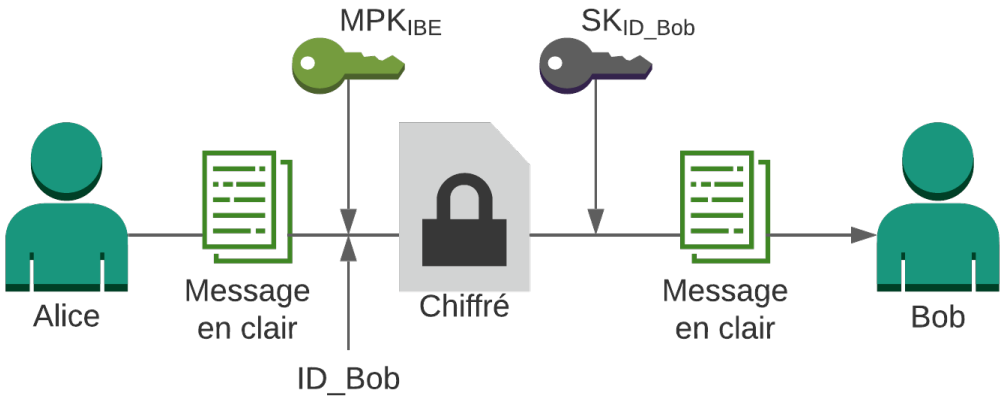


FIGURE 3.2 – Chiffrement IBE

3.1.3 Attribute-Based Encryption (ABE)

Le schéma ABE a été introduit par Sahai et Waters dans [119]. C'est un schéma de chiffrement asymétrique qui permet de chiffrer un message avec des attributs, ces derniers pouvant cibler plusieurs utilisateurs. Le déchiffrement de ce message n'est possible qu'en possédant une clé correspondant à ces attributs. ABE est donc un schéma de chiffrement à diffusion, c'est-à-dire que la donnée chiffrée n'est pas destinée à un seul utilisateur en particulier mais plutôt à un groupe d'utilisateurs ayant un ensemble d'attributs en commun. Cela permet par exemple, plutôt que de chiffrer un message pour n docteurs et ainsi effectuer n chiffrements, de chiffrer le message une seule fois avec l'attribut "**DOCTEUR**" de manière à ce que toute personne possédant cet attribut puisse accéder à cette donnée. Ainsi, un même chiffré peut être déchiffré avec plusieurs clés différentes.

Dans ce schéma, les interlocuteurs n'échangent pas de clé au préalable pour communiquer. Le chiffrement s'effectue avec la clé maîtresse publique tandis que le déchiffrement est fait avec une clé secrète délivrée par l'autorité de confiance.

Il existe essentiellement deux catégories d'ABE, qui se distinguent par la manière dont sont chiffrées les données ainsi que par la manière dont sont définies les clés secrètes :

— Si une politique d'accès est attachée au chiffré et que la clé secrète de l'utilisateur est définie par un ensemble d'attributs ; on parle de Ciphertext-Policy ABE (CP-ABE)

— Si un ensemble d'attributs est attaché au chiffré et que la clé secrète de l'utilisateur est définie par une politique ; on parle de Key-Policy ABE (KP-ABE)

Key-Policy Attribute-Based Encryption (KP-ABE)

Le schéma KP-ABE [63] est une forme d'ABE où la donnée va être définie par un ensemble d'attributs (par exemple un objet connecté sera associé aux attributs [**CAPTEUR**, **LIMOGES**]) et la clé secrète d'un utilisateur va être associée à une politique d'accès, cette dernière définissant les données auxquelles il peut accéder (par exemple "(**CAPTEUR** AND **LIMOGES**)"). Ainsi, un utilisateur pourra déchiffrer toutes les données présentant des attributs que sa politique valide. Cette approche permet de gérer l'accès aux données à partir des consommateurs, mais implique un renouvellement des clés lorsqu'un nouveau type de donnée est intégré au système.

Ciphertext Policy Attribute-Based Encryption (CP-ABE)

Le schéma CP-ABE [15], est une forme d'ABE où la donnée va être chiffrée avec une politique (par exemple "**DOCTEUR** AND **LIMOGES**"), et les clés secrètes vont être associées à un ensemble d'attributs (par exemple [**DOCTEUR**, **LIMOGES**, **CHU**]), comme illustré sur la Figure 3.3. Le déchiffrement est possible si la politique d'accès associée au message est satisfaite par les attributs de la clé secrète de l'utilisateur.

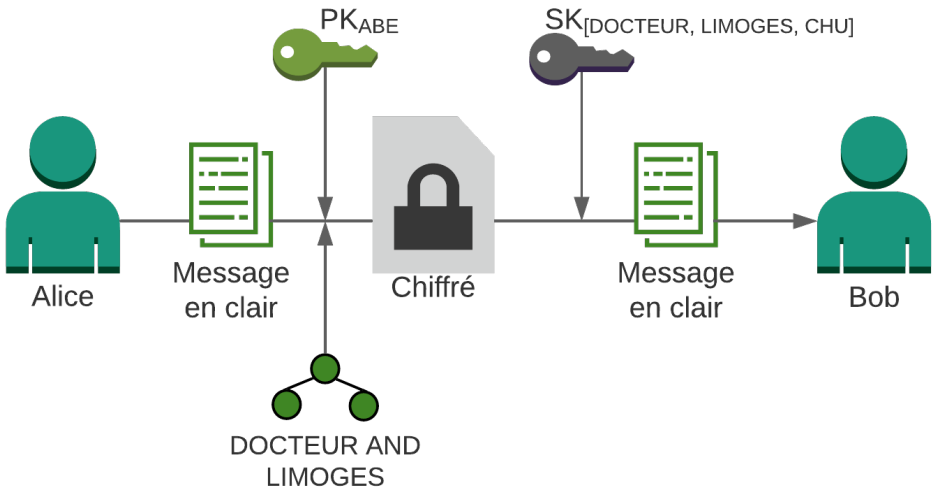


FIGURE 3.3 – Chiffrement CP-ABE

Même si une transformation directe est possible d'une forme vers l'autre, le schéma CP-ABE permet une gestion plus intuitive des accès à la donnée puisque ce sont les producteurs de données qui définissent la politique.

Le schéma CP-ABE est défini par les quatre algorithmes suivants :

— $\mathsf{Setup}(\mathfrak{K}) \rightarrow PK, MK$: Étant donné un paramètre de sécurité $\mathfrak{K}$, l'algorithme Setup génère une clé maîtresse publique PK ainsi qu'une clé maîtresse secrète MK.

— $\mathsf{KeyGen}(MK, \mathcal{A}) \rightarrow SK$: Étant donné un ensemble d'attributs $\mathcal{A}$ et la clé maîtresse secrète MK, l'algorithme KeyGen génère une clé secrète SK pour un utilisateur.

— $\mathsf{Encrypt}(PK, GT, m) \rightarrow c$: Lorsqu'un utilisateur veut chiffrer un message m, il choisit dans un premier temps une politique d'accès GT. Ensuite, avec l'algorithme Encrypt, la clé maîtresse publique PK, la politique d'accès GT et le message m, il génère le chiffré c.

— $\mathsf{Decrypt}(SK, c) \rightarrow m$: Lorsqu'un utilisateur veut déchiffrer un message, il utilise l'algorithme Decrypt avec sa clé secrète SK ainsi que le chiffré c. Le déchiffrement est possible si et seulement si l'ensemble d'attributs $\mathcal{A}$ correspond à la politique d'accès GT associée au chiffré. Dans ce cas seulement, l'utilisateur retrouve le message m après déchiffrement.

Le principal inconvénient des schémas IBE et ABE est qu'ils nécessitent une confiance totale en une autorité qui centralise l'ensemble de la distribution de clés. Pour pallier ce problème, des schémas réduisant le pouvoir de l'autorité ont été proposés.

Multi-Authority Attribute Based Encryption

Le schéma Multi-Authority Attribute Based Encryption (MA-ABE) est proposé par Chase et al. [28] en 2007. Alors que les schémas CP-ABE et KP-ABE sont gérés par deux clés maîtresses uniques, le schéma Multi Authority ABE sépare et distribue les clés maîtresses entre plusieurs entités. Chase et al. définissent dans leur première proposition un ABE géré par une autorité centrale qui va générer les clés maîtresses des autres autorités lors du démarrage du système.

Lorsqu'un utilisateur cherche à obtenir sa clé secrète, il effectue une requête à chaque autorité afin de récupérer une clé partielle qui correspond à la fois aux attributs qu'elle peut gérer ainsi qu'aux attributs qui correspondent à l'utilisateur. L'inconvénient de ce type d'architecture est que l'autorité centrale concentre encore trop de pouvoirs.

Lin et al. [82] présentent en 2008 un MA-ABE sans autorité centrale. Dans leur schéma, toutes les autorités collaborent au démarrage du système pour générer leurs clés. De la même

manière, chaque mise à jour de ces clés ne peut s'effectuer qu'avec une coopération de toutes les autorités.

Plus récemment, Qian et al. [108] proposent un Multi-Authority ABE avec révocation d'utilisateurs en temps réel. Ce schéma est basé sur les travaux de Li et al. [81] de 2012 et se formalise par les algorithmes suivants :

— Global Setup($\mathfrak{K}$) $\rightarrow param$: Étant donné un paramètre de sécurité $\mathfrak{K}$, l'algorithme Global Setup génère un ensemble de paramètres $param$ de sécurité.

— Authority Setup($param$) $\rightarrow SK_k, PK_k$: Étant donné un ensemble de paramètres $param$, l'algorithme Authority Setup genère la clé maîtresse secrète SK_k et la clé maîtresse publique PK_k pour une autorité A_k qui gère un ensemble d'attributs $\widetilde{A}_k$.

— KeyGen($SK_k, GID, \widetilde{A}_U^k$) $\rightarrow SK_U$: Pour une autorité A_k, étant donné un ensemble d'attributs $\widetilde{A}_U^k = \widetilde{A}_U \cap \widetilde{A}_k$ et la clé maîtresse secrète de l'autorité SK_k, l'algorithme KeyGen génère une clé secrète SK_U^k pour un utilisateur. L'ensemble des $\{SK_U^k\}$ fourni par chaque autorité est la clé secrète complète de l'utilisateur U, $\{SK_U\}$.

— Encrypt($\{PK_k\}, m, \mathcal{T}$) $\rightarrow CT$: Avec un message m, l'ensemble de clés maîtresses publiques de chaque autorité $\{PK_k\}$ et un arbre d'accès (représentant une politique d'accès) $\mathcal{T}$ l'algorithme Encrypt produit un chiffré CT.

— Decrypt(SK_U, CT) $\rightarrow m$: Avec une clé secrète SK_U ainsi qu'un chiffré CT, l'algorithme Decrypt permet de récupérer le message m, si et seulement si l'ensemble d'attributs $\widetilde{A}_U$ de l'utilisateur valide l'arbre d'accès $\mathcal{T}$ du chiffré.

— ReKeyGen($L, param$) $\rightarrow rk_k$: Pour une autorité A_k, étant donné un ensemble d'attributs L devant être mis à jour, l'algorithme ReKeyGen algorithme génère une clé de re-chiffrement rk_k. L'ensemble de clés $\{rk_k\}$ est la clé de re-chiffrement global rk.

— ReEnc(CT, rk, W) $\rightarrow CT'$: Étant donné un chiffré CT, la clé globale de re-chiffrement rk et un ensemble d'attributs W, qui contient tous les attributs de la politique d'accès original, l'algorithme ReEnc permet de re-chiffrer le message CT vers CT'.

— KeyUpdate(SK_U^k, rk, S) $\rightarrow SK_U'^k$: Étant donné une clé secrète SK_U^k, la clé globale de re-chiffrement rk et un ensemble d'attributs S, qui contient tous les attributs faisant partie

de la clé secrète devant être mis à jour, l'algorithme KeyUpdate lancé par l'autorité A_k, met à jour la clé secrète SK_U^k vers $SK_U'^k$.

— PolicyUpdate$(\{PK_{k,i}\}, CT, P') \rightarrow CT'$: Étant donné une clé publique maîtresse $\{PK_k\}$, le chiffré CT et la politique d'accès mise à jour P', l'algorithme PolicyUpdate permet de re-chiffrer le message CT vers CT' avec la nouvelle politique P'.

L'ensemble de ces solutions de chiffrement permettent de protéger les données lorsqu'elles transitent sur des serveurs.

3.2 Chiffrement recherchable (Searchable Encryption)

Le chiffrement recherchable ou searchable encryption est une primitive cryptographique qui permet aux utilisateurs de rechercher des données dans une base de données chiffrée. Ainsi, les requêtes mais aussi l'indexation de ces données restent chiffrées aussi bien pour le serveur hébergeant les données (ou les faisant transiter) que pour un attaquant écoutant les communications. C'est une propriété essentielle dans un modèle de sécurité où l'on veut garantir la protection de la vie privée de l'utilisateur.

Les schémas de searchable encryption se divisent en deux principales catégories : Symmetric Searchable Encryption (SSE) et Public key Encryption with Keyword Search (PEKS).

Les schémas de PEKS se divisent eux-mêmes en plusieurs catégories :

— Single Writer/Single Reader (S/S)

— Single Writer/Multiple Readers (S/M)

— Multiple Writers/Single Reader (M/S)

— Multiple Writers/Multiple Readers (M/M)

Cette thèse portant sur l'application de schémas de sécurité pour une architecture distribuée avec plusieurs utilisateurs, seuls les schémas PEKS (M/M) sont détaillés dans cette section. Un exemple de construction PEKS (M/M), est le schéma Searchable Data Encryption (SDE).

3.2.1 Searchable Data Encryption (SDE)

Le Searchable Data Encryption (SDE) est un schéma PEKS de type (M/M) proposé par Dong et al. [43]. Il est basé sur du Proxy Re-Encryption (PRE). Nous décrivons ici son fonctionnement général.

Lors de la mise en place du système, une autorité de confiance, notée TA, génère une clé publique PK_{SDE} ainsi qu'une clé maîtresse MK_{SDE}. La clé publique est alors envoyée à tous les utilisateurs ainsi qu'au proxy, tandis que la clé maîtresse est conservée par la TA.

Pour chaque nouvel utilisateur i, la TA génère deux clés : x_{i1} et x_{i2}. Les clés générées sont envoyées de la manière suivante : x_{i1} pour l'utilisateur et (i,x_{i2}) pour le proxy.

Lorsqu'un utilisateur va chercher à publier une donnée, l'index de la donnée (ce qui la caractérise) est chiffré une première fois avec la clé x_{i1}, puis re-chiffré par le serveur avec la clé x_{i2}.

Pour effectuer une recherche dans la base de données, un utilisateur devra générer une porte dérobée (ou trapdoor) avec sa clé x_{j1}. Cette dernière sera envoyée par l'utilisateur j, en tant que requête. Le proxy re-chiffrera la trapdoor avec la clé x_{j2}.

L'algorithme de correspondance pourra donc, à partir de la donnée chiffrée et de la trapdoor, vérifier si la requête correspond à la donnée.

Cette méthode permet de pouvoir vérifier la correspondance entre l'index chiffré et une trapdoor, tout en conservant la confidentialité de la requête. Elle présente néanmoins l'inconvénient d'être vulnérable à une coopération entre un utilisateur et le proxy, puisqu'ils peuvent combiner leurs clés afin de récupérer la clé maîtresse.

3.2.2 Attribute-Based Keyword Search (ABKS)

Un schéma Attribute-Based Keyword Search (ABKS) va permettre à un utilisateur de chiffrer un index avec une politique, et à un autre de générer une trapdoor avec sa clé secrète, celle-ci étant définie par un ensemble d'attributs. Ainsi, seul un utilisateur effectuant une recherche avec les bons mots clés et qui possède les attributs correspondant à la politique de l'index chiffré pourra récupérer la donnée qu'il recherche. L'ensemble de ce processus est présenté en Figure 3.4.

Sun et al. ont proposé le schéma Attribute-Based Keyword Search with User Revocation

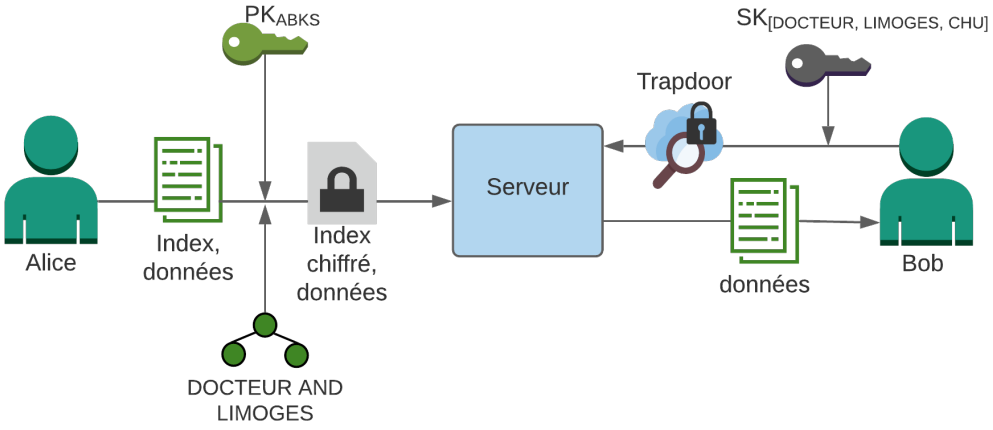


FIGURE 3.4 – Schéma ABKS

(ABKS-UR) [127], qui permet d'intégrer la propriété de searchable encryption tout en contrôlant l'accès à la donnée. Ce schéma permet également la révocation d'utilisateur en temps réel. Le scénario d'utilisation de ce schéma est un serveur distant qui reçoit les données de plusieurs producteurs ainsi que les requêtes sur ces données de plusieurs consommateurs. Le schéma s'opère de la manière suivante :

Dans un premier temps, un producteur de données définit une politique d'accès. Ensuite, il choisit un ensemble de mots clés qui décrivent la donnée qu'il cherche à envoyer. Enfin, avec sa politique, son ensemble de mots clés et la clé maîtresse publique ABKS-UR du système, il génère un index chiffré qui est envoyé au serveur accompagné de la donnée.

D'un autre côté, un consommateur va générer une trapdoor avec sa clé secrète ainsi qu'un ensemble de mots clés qui définissent sa recherche. Sa clé secrète est délivrée par la TA et représente l'ensemble des attributs qui le caractérise (c'est-à-dire son niveau d'accréditation). La trapdoor ainsi générée est envoyée au serveur, qui vérifie la correspondance entre cette trapdoor et l'ensemble d'index qu'il possède. Pour qu'une trapdoor corresponde à un index chiffré, il faut que les mots clés correspondent et que les attributs de l'utilisateur ayant généré la trapdoor correspondent à la politique d'accès attachée à l'index chiffré.

Le schéma ABKS-UR de Sun et al. est composé de neuf algorithmes. Le schéma se base sur l'espace de mots clés $\mathcal{W}$ et l'espace de politiques d'accès $\mathcal{G}$. On note $\mathcal{N}$ l'univers d'attributs $\{1, ..., n\}$ avec $n \in \mathbb{N}$. L'ensemble d'attributs d'une clé secrète est noté $\mathcal{A}$ avec $\mathcal{A} \subseteq \mathcal{N}$.

— $\mathsf{Setup}(\mathfrak{K}, \mathcal{N}) \rightarrow PK, MK$: Cet algorithme prend en entrée un paramètre de sécurité $\mathfrak{K}$

ainsi que l'univers d'attributs $\mathcal{N}$. Il définit le groupe bilinéaire $\mathbb{G}$ de base p et de générateur g, ainsi que l'opération de couplage $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. Il génère également la clé maîtresse publique PK et la clé maîtresse secrète MK. Le numéro de version *ver* de ces clés maîtresses est initialisé à 1.

— $\mathsf{CreateUL}(PK, ID) \to UL$: Cet algorithme prend en entrée la clé maîtresse publique PK ainsi que l'identité d'un utilisateur ID. Il crée UL, une liste de consommateurs gérée par un producteur de données.

— $\mathsf{EncIndex}(PK, GT, W) \to D$: Cet algorithme prend en entrée la clé maîtresse publique PK, la politique d'accès $GT \in \mathcal{G}$ et un ensemble de mots clés $W \subseteq \mathcal{W}$. Il génère l'index chiffré D.

— $\mathsf{KeyGen}(PK, MK, \mathcal{A}) \to SK$: Cet algorithme prend en entrée la clé maîtresse publique PK, la clé maîtresse secrète MK et un ensemble d'attributs $\mathcal{A}$. Il génère ainsi la clé secrète SK.

— $\mathsf{ReKeyGen}(\Phi, MK) \to rk, MK', PK'$: Cet algorithme est utilisé lors de la phase de révocation. Il prend en entrée un ensemble d'attributs Φ qui contient les attributs à mettre à jour et la clé maîtresse secrète MK. Il génère la clé de re-chiffrement rk qui sera utilisée pour re-calculer les clés pour chaque attribut dans $\mathcal{N}$, ainsi que les clé maîtresse mises à jour MK' et PK'. Les numéros de version *ver* de ces clés sont également incrémentées de 1. Les rk_i correspondants aux attributs ne faisant pas partie de l'ensemble Φ ont pour valeur 1.

— $\mathsf{ReEncIndex}(rk, D, \Delta) \to D'$: Cet algorithme prend en entrée la clé de re-chiffrement rk, l'index chiffré D et l'ensemble d'attributs Δ qui contient tous les attributs de la politique d'accès GT associée à D où les valeurs correspondantes des rk_i ne sont pas de 1. Il génère le nouvel index chiffré D'.

— $\mathsf{ReKey}(\Omega, rk, PSK) \to PSK'$: Cet algorithme prend en entrée l'ensemble d'attributs Ω qui contient les attributs à mettre à jour ainsi que la clé secrète partielle PSK qui correspond à la partie de la clé utilisateur dédiée aux attributs. Il utilise également la clé de re-chiffrement rk et génère la clé partielle mise à jour PSK' pour l'utilisateur.

— $\mathsf{GenTrapdoor}(PK, SK, W') \to Q$: Cet algorithme permet de générer une trapdoor à partir

d'un ensemble de mots clés et de la clé secrète de l'utilisateur. Il prend en entrée la clé maîtresse publique PK, la clé secrète de l'utilisateur SK et un ensemble de mots clés $W' \subseteq \mathcal{W}$. Il génère la trapdoor Q associée à cet ensemble de mots clés.

— $\mathsf{Search}(UL, D, Q) \rightarrow \{\mathsf{search\ result}, \perp\}$: Cet algorithme prend en entrée une liste d'utilisateurs UL, l'index chiffré D et la trapdoor Q. Il renvoie le résultat de la correspondance, ou $\perp$ si échec.

Pouvoir effectuer des requêtes chiffrées sur des données chiffrées de manière à ce qu'un serveur distant n'ait pas connaissance des intérêts de l'utilisateur protège la vie privée de ce dernier. Dans certains cas, il peut également être utile pour l'utilisateur de pouvoir s'authentifier au travers des données qu'il produit. Pour cela, on utilise des schémas de signature.

3.3 Intégrité et authentification : signatures

Lorsqu'un utilisateur veut s'authentifier au travers d'une donnée qu'il envoie, et assurer son intégrité, il peut utiliser le mécanisme de signature numérique, également appelé digital signature.

3.3.1 Signature électronique

La notion de digital signature a été définie pour la première fois par Diffie et Hellman en 1976. De la même manière qu'une signature manuscrite, elle lie le signataire avec le document qu'il signe. De plus, elle garantit l'intégrité d'un document. La signature électronique possède également la propriété de non répudiation, puisqu'on suppose que dans un schéma de signature sécurisé, il est impossible de forger une signature en se faisant passer pour quelqu'un d'autre.

Un schéma classique de signature, présenté en Figure 3.5, possède les algorithmes suivants :

— $\mathsf{Setup}(\mathfrak{K}) \rightarrow param$: Étant donné un paramètre de sécurité $\mathfrak{K}$, l'algorithme Setup génère les paramètres globaux du système.

— $\mathsf{KeyGen}(param) \rightarrow vk, sk$: L'algorithme KeyGen génère une clé publique de vérification vk ainsi qu'une clé secrète de signature sk.

— $\mathsf{Sign}(m, sk) \rightarrow \sigma$: L'algorithme Sign génère une signature σ d'un message m avec la clé sk.

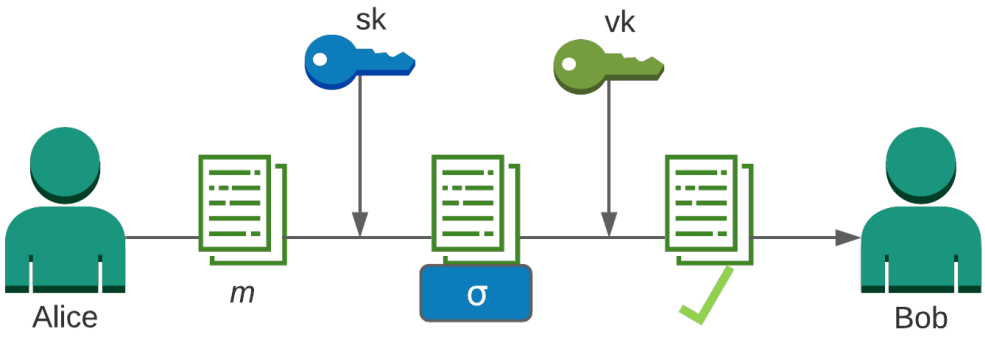


FIGURE 3.5 – Digital Signature

— $\mathsf{Verify}(vk, \sigma) \rightarrow \{0, 1\}$: L'algorithme Verify vérifie la signature σ avec la clé de vérification vk.

De la même manière que les schémas de chiffrements à clé publique, les signatures numériques demandent une gestion assez lourde des clés des utilisateurs. Ainsi, des travaux portant sur des schémas de signature plus flexibles ont été réalisés, afin de gérer avec plus de facilité un grand nombre d'utilisateurs dans un même système.

3.3.2 Attribute-Based Signature (ABS)

Le schéma Attribute-Based Signature (ABS) est une extension du schéma Identity-Based Signature (IBS) [121], où les utilisateurs peuvent s'authentifier au travers d'attributs les définissant plutôt que de leur identité. C'est-à-dire qu'un utilisateur peut signer un message avec n'importe quel prédicat obtenu à partir des attributs qui composent sa clé, cette dernière étant fournie par la TA. Cela permet à un utilisateur de s'authentifier anonymement puisque sa signature sera indistinguable d'un autre utilisateur signant avec le même prédicat. Le schéma *ABS with flexible threshold predicate support* proposé par Li et al. [80] permet aux utilisateurs de signer en fonction d'un prédicat seuil, relatif aux attributs distribués par la TA. Par exemple, un utilisateur peut signer avec les attributs [A,B,C] avec un seuil de 2. Cela signifie qu'il signe en assurant posséder au moins 2 de ces 3 attributs, c'est-à-dire soit [A,B], soit [A,C], soit [B,C], soit [A,B,C].

Une clé de signature est composée d'un ensemble d'attributs $\alpha \subseteq \mathcal{N}$. Pour signer un message, l'utilisateur choisira un sous ensemble $\alpha^* \subseteq \mathcal{N}$. Ce schéma consiste en ces quatre algorithmes :

— $\mathsf{Setup}(\mathfrak{K}, \mathcal{N}) \rightarrow PK, MK$: Cet algorithme génère une clé maîtresse publique PK ainsi

qu'une clé maîtresse secrète MK relativement au paramètre de sécurité $\mathfrak{K}$ et l'univers d'attributs $\mathcal{N}$.

— $\mathsf{Extract}(PK, MK, \alpha) \rightarrow SK$: Cet algorithme génère une clé secrète SK en fonction d'un ensemble d'attributs α pour un utilisateur.

— $\mathsf{Sign}(SK, \alpha^*, m) \rightarrow \sigma$: Cet algorithme permet à un utilisateur possédant une clé secrète SK relative à un ensemble d'attributs α de signer un message m en prouvant posséder k attributs d'un ensemble α^*. Pour cela il sélectionne un sous ensemble $\alpha*$, qui sera attaché à sa signature, ainsi qu'un sous ensemble α' de taille k tel que $\alpha' \subseteq \alpha \cap \alpha^*$.

— $\mathsf{Verify}(m, \alpha^*, PK, \sigma) \rightarrow \{0, 1\}$: Pour vérifier une signature σ associée à un message m, le vérifieur utilise la clé maîtresse publique PK. Il peut ainsi vérifier si le signeur possède k attributs de l'ensemble α^*.

Dans certains scénarios, la vérification d'une signature peut être un point sensible au niveau de la sécurité. En effet, vérifier une signature permet de lier une identité à un message, et permet au vérifieur de divulguer ensuite cette signature. De plus, dans le mécanisme classique, le signeur n'a pas de contrôle sur les personnes qui pourront vérifier la signature qu'il a généré dans la mesure ou la simple possession de sa clé publique permet de réaliser cette opération. Dans certains cas, il est préférable pour préserver la confidentialité des échanges tout en assurant une authentification d'avoir des signatures non divulgables et donc non opposables.

3.3.3 Designated Verifier Signature (DVS)

En 1989, Chaum et van Antwerpen introduisent la notion de undeniable signature [29], qui permet à un signeur de contrôler l'accès à ses signatures. Pour cela, le signeur doit participer au processus de vérification de manière à empêcher les utilisateurs non autorisés de connaître la validité de la signature. Cependant ce premier schéma est sensible à la collusion entre utilisateurs, ce qui provoque des fuites d'informations pour des utilisateurs non autorisés lors de la vérification.

Pour pallier ce problème, Jakobsson et al. [73] introduisent le premier schéma de signature à vérifieur désigné. Dans ce schéma, seul un vérifieur spécifique peut vérifier la validité d'une signature. Cela implique que le vérifieur possède une clé secrète qui permet la vérification. Une autre propriété de ce schéma est qu'il est impossible pour le vérifieur de prouver à un autre

utilisateur la validité de la signature. Cette propriété résulte du fait que le vérifieur à la possibilité de générer des signatures valides.

Un schéma de Designated Verifier Signature est défini par les algorithmes suivants :

— $\mathsf{Setup}(\mathfrak{K}) \rightarrow param$: Génère les paramètres globaux du système $param$.

— $\mathsf{KGS}(\mathsf{param}) \rightarrow (\mathsf{sk}_1, \mathsf{pk}_1)$: Génère une paire de clés de signature $(\mathsf{sk}_1, \mathsf{pk}_1)$.

— $\mathsf{KGV}(\mathsf{param}) \rightarrow (\mathsf{sk}_2, \mathsf{pk}_2)$: Génère une paire de clés pour un vérifieur $\mathsf{sk}_2, \mathsf{pk}_2$.

— $\mathsf{Sign}(\mathsf{sk}_1, m) \rightarrow \sigma$: Génère une signature σ vérifiable avec pk_1.

— $\mathsf{Verify}(\mathsf{pk}_1, m, \sigma) \rightarrow \{0, 1\}$: Vérifie la validité de la signature σ.

— $\mathsf{Des}(\mathsf{pk}_1, \mathsf{pk}_2, m, \sigma) \rightarrow \hat{\sigma}$: Génère une signature $\hat{\sigma}$ vérifiable avec sk_2.

— $\mathsf{DesV}(\mathsf{sk}_2, \mathsf{pk}_1, m, \hat{\sigma}) \rightarrow \{0, 1\}$: Vérifie la validité de la signature $\hat{\sigma}$.

L'utilisation d'un tel schéma dans des scénarios pratiques demande une gestion de clés assez contraignante, qui ralentit le système lors des signatures/vérifications. Ces problèmes étant de plus en plus présents avec un grand nombre d'utilisateurs, afin de mieux gérer cette situation, des travaux ont été effectués autour de signature à vérifieur désignés basés sur l'identité [128]. Cette primitive permet de désigner des vérifieurs avec leur identité (par exemple une adresse mail) plutôt que leur clé publique. L'itération de cette solution est le schéma Attribute-Based Designated Verifier Signature (ABDVS), une signature à vérifieur désigné basée sur les attributs.

3.3.4 Attribute-Based Designated Verifier Signature (ABDVS)

Le schéma ABDVS combine les propriétés d'un DVS et d'un ABS. Il permet à un utilisateur d'assurer l'authenticité et l'intégrité des données qu'il envoie tout en désignant plusieurs vérifieurs au travers d'une politique. En d'autres termes, cela permet à l'utilisateur de protéger sa signature, tout en permettant à des vérifieurs ayant des clés secrètes différentes de la vérifier.

Un schéma Attribute-Based Designated Verifier Signature est défini par les algorithmes suivants :

— $\mathsf{Setup}(\mathfrak{K}) \rightarrow (\mathsf{param}, \mathsf{mpk}, \mathsf{msk})$: Génère les paramètres globaux du système param ainsi que les clés maîtresses mpk et msk.

— $\mathsf{KGS}(\mathsf{param}, \mathsf{msk}, \mathsf{id}) \rightarrow (\mathsf{sk}, \mathsf{ek})$: Génère une clé de signature et d'encapsulation $(\mathsf{sk}, \mathsf{ek})$ pour l'identité id.

— $\mathsf{KGV}(\mathsf{param}, \mathsf{mpk}, \mathsf{msk}, \mathsf{id}) \rightarrow (\mathsf{usk}, \mathsf{dk})$: Génère une paire de clés de vérification $(\mathsf{usk}, \mathsf{dk})$.

— $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, \mathsf{ek}, \mathsf{id}, m, \mathbb{F}) \rightarrow \sigma$: Génère une signature σ avec la politique $\mathbb{F}$ et l'identité id.

— $\mathsf{Verify}(\mathsf{mpk}, \mathsf{usk}, \sigma, m) \rightarrow \{0, 1\}$: Vérifie la validité de la signature σ.

À notre connaissance, le seul schéma ABDVS est celui proposé par Fan et al. [51]. C'est pourquoi nous avons travaillé sur une nouvelle version de ce schéma, reposant sur des primitives différentes. Ces travaux sont présentés et détaillés dans le chapitre 6.

3.4 Conclusion du chapitre

Nous avons présentés dans ce chapitre plusieurs primitives cryptographiques qui recouvrent différents aspects de la protection des données.

Nous avons dans un premier temps détaillé les différents mécanismes permettant de chiffrer des données. Ce processus va la plupart du temps s'effectuer en deux étapes. Dans un premier temps, les utilisateurs souhaitant communiquer vont utiliser un schéma asymétrique, d'échange de clé ou de chiffrement, pour chiffrer une clé secrète pour un schéma de chiffrement symétrique. Cette clé secrète leur permettra ensuite de chiffrer l'ensemble de la communication à l'aide d'un algorithme de chiffrement symétrique puisque ce sont en général des schémas beaucoup plus efficaces en termes de temps de chiffrement/déchiffrement. Les schémas asymétriques que nous utilisons dans cette thèse sont les schémas ABE et MA-ABE. Pour le chiffrement symétrique, nous utilisons le standard AES.

Ensuite, nous avons présenté le Searchable Encryption, qui permet d'effectuer des recherches dans une base de données chiffrée. Tout comme pour un chiffrement classique, il existe des schémas de searchable encryption symétriques et asymétriques. Or, nos cas d'utilisation présentant de nombreux utilisateurs, il n'est pas envisageable de partager beaucoup de clés secrètes entre eux. C'est pourquoi nous nous sommes concentrés sur les schémas de searchable encryption asymétriques. Nous avons choisi de détailler les deux schémas qui correspondent le mieux à des scénarios avec plusieurs producteurs et plusieurs consommateurs : le SDE et l'ABKS.

Pour finir, nous avons abordé la problématique d'authentifier et d'assurer l'intégrité des données envoyées. Même s'il existe des schémas de signature asymétriques classiques, ils ne correspondent pas à un scénario avec de nombreux utilisateurs. En effet, cela impose une infrastructure de gestion de clés conséquente. Pour nos scénarios dans des environnements de santé, nous avons choisi de retenir des schémas basés sur les attributs. Nous avons détaillé le fonctionnement d'un ABS, ainsi que celui d'un ABDVS.

L'ensemble de ces solutions permettent donc d'assurer la confidentialité et la protection de la vie privée des utilisateurs, au niveau de leurs données et de leurs requêtes. De plus, elles leurs permettent de s'authentifier avec leurs données. La plupart de ces schémas nécessitent certaines adaptations afin de les utiliser dans des architectures distribuées et cette intégration constitue une partie importante des travaux effectués pendant cette thèse.

Chapitre 4

Solution de sécurité basée sur les attributs pour une architecture topic-based publish/subscribe

Dans ce chapitre nous présentons notre solution de sécurité pour l'architecture topic-based publish/subscribe. Pour illustrer cette solution, nous nous basons sur le scénario d'environnement d'aide à l'autonomie à domicile présenté en Section 2.4.1. L'objectif est de garantir la protection de la vie privée des patients dont les données sont utilisées dans le cadre de cet environnement.

Les objets connectés utilisés dans ce type de scénario ne disposent pas de la puissance de calcul nécessaire pour chiffrer leurs propres données. On considère donc que pour faire le lien entre les objets connectés et le serveur distant (broker), une ou plusieurs passerelles (type Raspberry Pi) rassemblent toutes les données produites par les objets connectés avant de les chiffrer.

Cette solution utilise les cryptosystèmes basés sur les attributs : ABE, ABKS-UR et ABS, pour respectivement protéger le contenu des publications, le topic des publications et assurer l'intégrité des données via une authentification anonyme. Ces primitives ainsi que certains cryptosystèmes sont présentés dans le Chapitre 3.

Sommaire

4.1 Présentation de l'architecture de notre solution

Afin de mieux appréhender l'ensemble de l'architecture ainsi que les enjeux de sécurité de celle-ci, nous détaillons le fonctionnement et les caractéristiques de chaque entité de l'architecture topic-based publish/subscribe. Ensuite, nous présentons l'ensemble des exigences de sécurité que nous fixons pour notre solution de sécurité. Enfin, nous définissons l'ensemble des menaces auxquelles doit faire face cette solution.

4.1.1 Entités

Les entités de notre architecture sont détaillées dans cette section. À noter que l'autorité de confiance présentée ici a été ajoutée afin de distribuer les clés des cryptosystèmes que l'on utilise. Son utilisation est détaillée en section 4.4.

— Le **Broker** est responsable de la transmission des données entre publieurs et souscripteurs. Les données envoyées par les publieurs sont appelées des publications et sont associées à un topic. Les publications ne sont pas stockées sur le broker et sont directement transmises aux souscripteurs concernés. Le format des publications est (topic,contenu).

— Les **Publieurs** peuvent envoyer des publications au broker. Lorsqu'un broker reçoit une nouvelle publication, il parcourt sa base de données pour trouver les souscriptions qui correspondent au topic de la publication. Si le topic de la publication existe déjà, le broker transmet directement la publication aux souscripteurs.

— Les **Souscripteurs** peuvent souscrire à un ou plusieurs topics différents. Pour effectuer une souscription, le souscripteur sélectionne un topic et le transmet au broker. Il est ensuite dans un état passif où il attend de recevoir de nouvelles publications qui correspondent à ses intérêts.

— L'**Autorité de confiance** (TA) est en charge de la distribution de clés aux utilisateurs. Les utilisateurs s'authentifient aux autorités pour récupérer leurs différentes clés secrètes : ABE, ABKS-UR et ABS.

Lorsque l'on applique notre scénario présenté en Section 2.4.1 à cette architecture, on obtient les correspondances suivantes : les publieurs sont les capteurs portables et ambiants de l'environnement AAL, avec les données qui sont envoyées à travers la passerelle de type Raspberry Pi située dans la maison. Les souscripteurs sont l'ensemble des professionnels étant en charge du patient (médecin traitant, spécialistes, etc.). Enfin, le broker est responsable de transmettre les données de la passerelle vers les professionnels. L'autorité de confiance est le prestataire (public ou privé) de la solution d'aide à l'autonomie à domicile.

Les composants de l'architecture topic-based publish/subscribe ainsi que leurs rôles étant définis, nous présentons maintenant les exigences de sécurité ainsi que le modèle de menaces de notre solution.

4.1.2 Exigences de sécurité

Pour répondre au mieux aux menaces sur la sécurité de l'architecture, nous établissons les exigences de sécurité suivantes :

— E1 : Le contenu d'une publication doit être chiffré de manière à être masqué pour le broker ainsi que pour toute partie non autorisée.

— E2 : Le broker doit être capable d'associer des publications à des souscriptions sans avoir connaissance ni du topic attaché à la publication ou la souscription, ni du contenu de la publication.

— E3 : Si un utilisateur est exclu du système, il ne doit plus avoir accès aux futures publications.

— E4 : Le publieur doit avoir la possibilité d'assurer l'intégrité de ses données ainsi que de s'authentifier au travers de ses publications.

4.1.3 Modèle de menaces

Dans ce scénario d'un environnement AAL envoyant des informations via une architecture topic-based publish/subscribe, avec les rôles précédemment établis pour les entités du système, nous considérons les menaces suivantes sur la sécurité :

— Les opérations de publication et de souscription sur le broker ne sont pas limitées/contrôlées (il ne rejette pas les communications).

— Des souscripteurs malhonnêtes peuvent collaborer pour essayer d'accéder à des données auxquelles ils n'auraient normalement pas accès.

— Un utilisateur malhonnête peut corrompre des données.

— Un souscripteur révoqué essayera toujours d'accéder au broker.

En plus de ces menaces, il est important de prendre en compte le modèle de sécurité du broker : *honest-but curious*, c'est-à-dire que l'on considère qu'il suit la procédure et qu'il ne corrompt pas les données, mais essaye de s'informer le plus possible sur le topic et le contenu des publications, ainsi que sur les souscriptions des souscripteurs.

Dans ce modèle, une autorité de confiance, notée TA, est chargée de générer les clés pour les utilisateurs. Elle est également en mesure de révoquer un utilisateur. Tous les utilisateurs la considère comme étant de confiance.

4.2 État de l'art

Dans cette section, nous présentons un certain nombre de travaux sur la sécurité de l'architecture publish/subscribe. Ces solutions répondent plus ou moins partiellement aux exigences de sécurité présentées en Section 4.1.2.

4.2.1 Contrôle d'accès utilisateurs

De nombreuses solutions basées sur le contrôle d'accès au niveau utilisateur avec un portail d'authentification ont été proposées dans la littérature. Ces dernières utilisent des technologies comme Role-Based Access Control (RBAC) [14, 122], Attribute-Based Access Control (ABAC) [54, 56] et Policy-Based Access Control (PBAC) [86, 67]. Dans ces solution, l'authentification doit être effectuée par une entité de confiance (par exemple un proxy). Cette entité doit également être en ligne de manière permanente.

Ces schémas permettent de gérer de manière fluide les accès aux données, néanmoins ces dernières ne sont pas chiffrées lorsqu'elles transitent via le broker. Ainsi, ce dernier a accès à

toutes les informations échangées par les utilisateurs.

De plus, dans un scénario avec un grand nombre d'utilisateurs, le portail qui assure l'authentification des utilisateurs risque d'être surchargé puisqu'il gère l'intégralité des communications. C'est pour cela que plusieurs recherches récentes se sont orientées vers le contrôle d'accès au travers du chiffrement des données, qui ne nécessite pas une telle entité de confiance.

4.2.2 Chiffrement point à point - SSL/TLS

Utilisé dans de nombreux scénario, SSL/TLS peut également être utilisé dans une architecture publish/subscribe, comme le protocole MQTT [11]. Dans ce protocole, ce mécanisme est néanmoins optionnel. En effet, dans une configuration basique, les données sont envoyées vers le port 1883 du broker par défaut, qui correspond au canal en clair, tandis que le canal chiffré est sur le port 8883. Il est donc nécessaire de re-configurer le broker ainsi que les clients pour utiliser cette fonctionnalité.

Le protocole **TLS** (Transport Layer Security) permet :

— L'authentification du serveur

— La confidentialité des données échangées

— L'intégrité des données échangées

— De manière optionnelle, l'authentification du client (mais dans la réalité celle-ci est souvent assurée par le serveur).

Ce protocole (et, de manière plus générale le chiffrement point à point) permet d'établir une connexion sécurisée entre les entités. Il permet par exemple d'empêcher un utilisateur extérieur d'intercepter les messages transitant du publieur vers le broker ou du broker vers le souscripteur. Or, dans une architecture publish/subscribe, le broker n'est pas considéré totalement de confiance, et utiliser uniquement TLS lui permet d'accéder aussi bien au au label qu'au contenu des publications.

De plus ce protocole de sécurité ne correspond pas à un scénario avec un grand nombre de publieurs et de souscripteurs. En effet, chaque publieur doit gérer sa propre bi-clé par souscripteur, ce qui fait que pour n communications, il est nécessaire de maintenir n paires de clés.

4.2.3 Chiffrement de bout en bout

Une approche différente de TLS consiste à chiffrer les données de l'émetteur au récepteur, de manière à ce qu'elles restent confidentielles pour tous les intermédiaires. Ce chiffrement est également appelé chiffrement de bout en bout. Avec cette approche, on peut distinguer deux types de chiffrement : symétrique ou asymétrique.

Certaines solutions utilisent des schémas de chiffrement symétrique, avec une clé par souscripteur [32] ou une clé par groupe de souscripteurs [101, 136, 110, 103]. Ces schémas nécessitent donc un échange direct entre les publieurs et les souscripteurs, ce qui renforce le couplage entre eux. Ce type de solution présente également des problèmes de mise à l'échelle puisque les données doivent être chiffrées plusieurs fois, une fois par souscripteur/groupe de souscripteurs.

Ce même problème apparaît lorsqu'on utilise un schéma de chiffrement asymétrique. En effet, avec des schémas comme RSA [116] ou El Gamal [45], le publieur doit chiffrer chaque publication (ou au moins une clé symétrique) autant de fois qu'il y a de souscripteurs différents avec des clés publiques différentes.

Afin de pallier les manques des schémas de chiffrement symétriques et asymétriques classiques, certaines solutions utilisent du chiffrement basé sur les attributs (défini en section 3.1.3), comme présenté dans [47]. L'avantage d'utiliser ce type de chiffrement est que les publieurs n'ont besoin de chiffrer qu'une seule fois la donnée pour l'ensemble des souscripteurs. Yuen et al. [137], définissent donc un nouveau modèle de sécurité pour l'architecture content-based publish/subscribe et décrivent leur solution de sécurité. Cette solution utilise le chiffrement ABE afin de garantir la confidentialité du contenu de la publication, ainsi que le schéma de *sanitizable signature* développé par Suzuki et al. [129] pour permettre l'authentification et l'intégrité de la publication du publieur (E1, E4).

Plus récemment, Thatmann et al. [131] ont utilisé un schéma ABE pour chiffrer des clés AES servant au chiffrement des données, et ainsi permettre une révocation d'utilisateur avec une mise à jour globale des clés AES utilisées (E1, E3). Cette solution appliquée à une architecture topic-based publish/subscribe ne gère néanmoins pas la confidentialité des topics et des souscriptions.

Ion et al. [66] appliquent une solution de sécurité dans une architecture content-based publish/subscribe en utilisant un ABE sur le contenu des données (E1). Ils proposent également une

solution pour sécuriser les topics en utilisant un schéma de chiffrement recherchable, Searchable Data Encryption (SDE) proposé par Dong et al. [43] (E2). Cependant, ce schéma nécessite la mise en place d'un proxy en ligne de manière permanente qui effectue le chiffrement recherchable, ce qui augmente de manière conséquente la complexité globale de la solution, et transfère la problématique de confidentialité du broker vers un proxy.

4.3 Sécurité de notre solution

Pour élaborer notre solution, nous avons donc cherché à pallier les manques des solutions déjà existantes en se basant sur nos exigences de sécurité. Pour commencer, nous avons chiffré le contenu des publications sans que ce chiffrement n'implique un échange de clé direct entre publieurs et souscripteurs.

Puisque l'on considère le serveur relayant les informations *honest-but-curious*, il n'est pas suffisant de seulement chiffrer la donnée, il faut également rendre confidentielles les recherches (souscriptions) effectuées, ainsi que le topic des publications. Pour cela, il est donc nécessaire d'utiliser un schéma de chiffrement recherchable qui ne nécessite pas de proxy et n'implique pas un échange de clé direct entre publieurs et souscripteurs.

Pour finir, un publieur doit pouvoir s'authentifier lorsqu'il envoie une publication dans une telle architecture. Dans notre scénario, les données transmises sont des données médicales, et si ces dernières sont falsifiées ou incomplètes, cela peut entraîner une mauvaise prise en charge ou un mauvais traitement du patient. Par défaut, un souscripteur ne peut pas savoir si c'est un publieur légitime qui a envoyé des données sur un topic. C'est pourquoi nous mettons en place un mécanisme d'authentification via une signature.

Pour finir, la révocation d'utilisateurs peut représenter un point critique pour la sécurité d'un système. Il faut pouvoir empêcher un utilisateur ne possédant plus les droits nécessaires d'accéder aux nouvelles données publiées. C'est pourquoi nous incluons un mécanisme permettant de révoquer un souscripteur d'un topic, ou même de le révoquer totalement du système.

4.3.1 Respect des exigences de sécurité

Dans cette section nous présentons les mesures permettant de répondre aux exigences de sécurité définies en Section 4.1.2.

Tout d'abord, pour permettre la confidentialité du contenu des publications (E1), nous utilisons le schéma Ciphertext-Policy Attribute-Based Encryption (CP-ABE) proposé par Waters et al. [135]. Ce schéma, présenté en Section 3.1.3, permet au publieur de choisir une politique d'accès P qui définit, sous la forme d'un ensemble d'attributs et de conditions, quelles sont les contraintes auxquelles doit répondre un souscripteur pour accéder à la donnée. Ensuite, le publieur utilise la clé maîtresse ABE (PK_{ABE}) pour chiffrer le contenu de la publication avec cette politique. Seul un utilisateur possédant une clé secrète (SK_{ABE}) associée à des attributs répondant à cette politique d'accès peut déchiffrer le message.

Ensuite, nous répondons au problème de confidentialité du topic publié et souscrit à l'aide du schéma Attribute-Based Keyword Search with a User Revocation (ABKS-UR) proposé par Sun et al. [127]. Ce schéma ayant été conçu pour fonctionner dans une architecture centralisée, nous avons dû le modifier pour qu'il s'intègre à notre architecture publish/subscribe. Ces modifications sont détaillées en Section 4.5.

Cette version modifiée du schéma permet au broker d'associer des publications chiffrées à des souscriptions chiffrées (E2). Pour cela, le publieur chiffre le topic associé à sa publication avec la clé maîtresse publique ($PK_{ABKS-UR}$) tandis que le souscripteur génère une porte dérobée, ou trapdoor (i.e. une souscription chiffrée), avec sa clé secrète ($SK_{ABKS-UR}$). Cette trapdoor définit les intérêts du souscripteur. Lorsque le broker reçoit une nouvelle publication, il utilise l'algorithme de vérification du schéma ABKS-UR pour vérifier la correspondance entre le topic de la publication ainsi que les topics des souscriptions présentes dans sa base de données. Le topic de la publication est chiffré avec une politique tandis que la trapdoor du souscripteur est générée avec une clé secrète associée à un ensemble d'attributs. Dans notre solution, nous utilisons la même politique pour chiffrer le topic et le contenu de la publication, puisque nous considérons qu'une personne pouvant souscrire à un topic doit pouvoir consulter les données qui y transitent.

La possibilité de révoquer un utilisateur (E3) est assurée par le protocole de révocation du schéma ABKS-UR. Ce protocole permet d'empêcher la correspondance entre n'importe quel topic

chiffré et une trapdoor générée par une clé révoquée.

Enfin, les publieurs peuvent assurer l'intégrité de leurs données ainsi que s'authentifier anonymement avec leurs publications (R4) en utilisant le schéma de signature par attributs - Attribute-Based Signature (ABS) proposé par Li et al. [80]. Le publieur signe sa publication avec sa clé secrète (SK_{ABS}) qui correspond à son statut sous la forme d'un ensemble d'attributs. Le souscripteur peut vérifier l'intégrité du message ainsi que le statut du publieur en utilisant la clé maîtresse (PK_{ABS}).

Notre solution utilise donc trois cryptosystèmes basés sur les attributs : ABE, ABKS-UR et ABS. Ils partagent le même univers d'attributs afin de faciliter le déploiement de la solution. Utiliser des attributs pour authentifier les utilisateurs nous permet d'assurer la confidentialité des données sans compromettre le découplage publieur/souscripteur de l'architecture publish/subscribe.

Une autorité de confiance (TA) génère les clés maîtresses de ces trois cryptosystèmes et assure la distribution des clés à tous les utilisateurs. Cette distribution est illustrée dans la Figure 4.1.

4.3.2 Modèle de sécurité de l'ABKS-UR

Nous définissons dans cette section le modèle de sécurité du schéma ABKS-UR avec ses modifications.

Indiscernabilité des souscripteurs

Dans cette expérience, nous voulons démontrer que des souscripteurs collaborant ne peuvent pas générer une clé ABKS-UR qui leurs permettrait d'accéder à des données auxquelles ils n'auraient pas accès seuls. En pratique, l'adversaire choisit un ensemble de souscripteurs corrompus, une politique à laquelle aucun d'eux ne peut répondre et deux messages M_0, M_1. Il reçoit, avec cette politique, le chiffré d'un de ces deux messages. Il gagne s'il réussit à deviner quel message a été chiffré avec un avantage non négligeable.

Indistinguabilité des trapdoors

Dans cette expérience, nous voulons démontrer que le broker ne peut pas distinguer que deux trapdoors Q_0, Q_1 ont été générées indépendamment l'une de l'autre.

Lorsque la trapdoor est générée, l'utilisateur choisit un nombre aléatoire u pour brouiller

la trapdoor. Cela signifie qu'un observateur est incapable de distinguer deux trapdoors mêmes si elles ont été générées avec les mêmes mots clés et le même utilisateur. Cela permet une indistinguabilité des trapdoors par sa construction.

Avec un couplage faible entre publieurs et souscripteurs, il semble impossible d'obtenir une confidentialité maximale de la trapdoor puisque cela demanderait l'utilisation de trapdoors à usage unique pour éviter que le broker n'effectue des corrélations entre elles. Or, l'utilisation de telles trapdoors implique une interaction entre publieurs et souscripteurs ce qui compromet la propriété de couplage faible dans une architecture publish/subscribe.

Sécurité sémantique des mots clés

Dans cette expérience, nous voulons démontrer que le broker ne peut pas savoir quel topic est visé par un chiffré. En pratique, l'adversaire choisit un message M ainsi que deux topics T_0 and T_1. Il reçoit le chiffrement de M associé à un des deux topics. Il gagne s'il réussit à deviner à quel topic correspond ce message avec un avantage non négligeable. L'adversaire peut également demander des trapdoors visant des topics différents de T_0, T_1. Cela signifie que la construction est sémantiquement sécurisée contre l'attaque par mot clé choisi sous le modèle de politique de texte chiffré sélectif (IND-sCP-CKA).

4.4 Solution détaillée

Dans cette section, nous présentons de manière détaillée notre solution de sécurité pour une architecture topic-based publish/subscribe appliquée au scénario d'environnement d'aide à domicile. Notre solution présente donc les propriétés suivantes :

— Chiffrement des publications avec un CP-ABE.

— Chiffrement du topic des publications et souscriptions avec une version modifiée d'ABKS-UR.

— Authentification des publieurs et intégrité des données avec un ABS.

Notre solution se compose de sept phases qui permettent son bon fonctionnement : Mise en place du système, Ajout d'un nouveau publieur, Ajout d'un nouveau souscripteur, Publication, Souscription, Processus de correspondance et Révocation d'utilisateur. Les modifications effectuées

sur le schéma ABKS-UR de Sun et al. sont détaillées en Section 4.5. Ces modifications n'entraînent aucune baisse du niveau de sécurité par rapport au schéma d'origine. Les algorithmes non mentionnés conservent leur forme originale.

Afin d'illustrer au mieux le fonctionnement de notre solution, nous présentons un exemple pour chaque phase basée sur le scénario AAL présenté en Section 2.4.1.

4.4.1 Mise en place du système

Lorsque le système est mis en place, la TA utilise l'algorithme Setup de chaque schéma (CP-ABE, ABKS-UR et ABS) et génère les clés suivantes : PK_{ABE}, MK_{ABE}, $PK_{ABKS-UR}$, $MK_{ABKS-UR}$, PK_{ABS} et MK_{ABS}. Les clés maîtresses publiques sont accessibles à tous tandis que les clés maîtresses secrètes sont conservées par la TA.

Cette autorité de confiance est donc responsable de distribuer les clés secrètes aux différents professionnels impliqués dans le projet ainsi qu'aux patients. Les attributs utilisés dans ces schémas permettent de symboliser le statut ainsi que l'organisme duquel dépend le professionnel.

4.4.2 Ajout d'un nouveau publieur

Lorsqu'un nouveau publieur (c'est-à-dire un patient) est intégré au projet, il fait une requête à la TA pour recevoir sa clé secrète ABS. Cette dernière est associée à un ensemble d'attributs α qui correspond à son statut. Elle est générée par l'algorithme ABS Extract.

Chaque patient possède sa propre clé de signature qui lui permet d'assurer l'intégrité des données qu'il transmet ainsi que de prouver son appartenance au groupe "Patient". Cela permet d'éviter qu'un utilisateur ne faisant pas partie du système publie des informations à sa place. Les attributs attachés à un patient regrouperont par exemple son statut ainsi que la ville dans laquelle il se situe : [PATIENT,LIMOGES]. Le fait que cette authentification soit anonyme permet à des organismes d'effectuer des statistiques sur ces patients sans pouvoir faire le lien avec leur identité.

4.4.3 Ajout d'un nouveau souscripteur

Lorsqu'un professionnel intègre le projet, il (ou son organisme) effectue une requête à la TA pour recevoir les clés SK_{ABE} et $SK_{ABKS-UR}$, associées à un ensemble d'attributs $\mathcal{A}$ qui

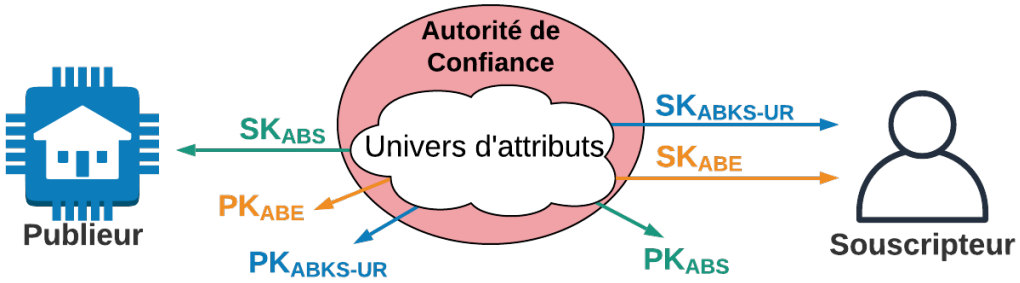


FIGURE 4.1 – Distribution des clés par l'autorité de confiance

correspond à son statut. À noter que $\mathcal{A}$ correspond aux attributs spécifiquement attachés aux professionnels de santé et que α corresponds aux attributs attachés aux patients. Ce sont donc des univers d'attributs différents. Ces clés sont générées avec les algorithmes ABE et ABKS-UR KeyGen.

Chaque professionnel possède sa propre paire de clés ABE et ABKS-UR lui permettant de déchiffrer le contenu des publications ainsi que de générer une trapdoor. Les attributs associés à ses clés regroupent par exemple son statut, l'organisme duquel il dépend et la ville dans laquelle il travaille : [AIDE SOIGNANT,CHU,LIMOGES].

Un récapitulatif de l'ensemble des clés générés par la TA est illustré dans la Figure 4.1.

4.4.4 Publication

Lors de l'envoi d'une nouvelle publication, la passerelle mise en place chez le patient doit tout d'abord chiffrer le topic avec l'algorithme ABKS-UR EncIndex. Ensuite, elle chiffre le contenu de la publication avec l'algorithme ABE Encrypt. Ces deux schémas utilisent la même politique d'accès pour restreindre l'accès aux données.

La publication (D, c), avec D le topic chiffré et c le chiffré ABE, est donc envoyée au broker éventuellement accompagnée de la signature σ générée par l'algorithme ABS Sign. À noter que même si les algorithmes ABE et ABKS ne sont pas liés lors du chiffrement, les publications chiffrées sont protégées et ne sont pas vulnérables aux attaques visant à remplacer le chiffré afin de corrompre l'intégrité des données. En effet, la signature est effectuée sur le topic ainsi que sur le contenu pour résister à cette attaque.

En fonction de la donnée publiée par la passerelle, la politique choisie par la passerelle est

différente puisque la donnée est destinée à une catégorie de professionnel spécifique. Par exemple, si la publication contient des données transmises par un capteur cardiaque porté par le patient, la politique pourrait être définie telle que : "(MEDECIN ET LIMOGES) OU CARDIOLOGUE".

4.4.5 Souscription

Lorsqu'un professionnel souhaite souscrire à un topic, il doit générer une trapdoor avec sa clé secrète ABKS-UR en utilisant l'algorithme GenTrapdoor. Le topic auquel il souhaite souscrire s'exprime sous la forme d'un ensemble de mots clés noté W. La trapdoor est ensuite envoyée au broker en tant que souscription. Si sa souscription correspond à un topic auquel il peut accéder, il reçoit la donnée chiffrée en ABE. Ensuite, en utilisant l'algorithme ABE Decrypt et sa clé secrète SK_{ABE}, il peut déchiffrer le message. Enfin, si la publication possède une signature, il peut la vérifier avec l'algorithme ABS Verify et la clé maîtresse publique PK_{ABS}.

4.4.6 Processus de correspondance

Lorsque le broker lance le processus de correspondance entre une nouvelle publication et l'ensemble de souscriptions qu'il a reçu auparavant, l'algorithme ABKS-UR Search est utilisé pour comparer le topic chiffré de la publication avec chaque trapdoor stockée. Si une correspondance est trouvée, le broker transmet la publication vers le professionnel concerné.

4.4.7 Révocation d'un souscripteur

Pour révoquer un professionnel du système, la TA utilise les algorithmes ABKS-UR ReKeyGen, ReEncIndex et ReKey pour mettre à jour toutes les clés ABKS-UR concernées. C'est-à-dire que les clés maîtresses $PK_{ABKS-UR}$ et $MK_{ABKS-UR}$ sont re-générées, et les topics sont chiffrés de nouveau. Enfin, les clés secrètes des utilisateurs légitimes possédant les mêmes attributs que ceux de l'utilisateur révoqué sont mises à jour par la TA.

4.4.8 Discussion

Pour résumé, notre solution de sécurité pour une architecture topic-based publish/subscribe appliquée à un scénario d'environnement d'aide à domicile utilise le schéma CP-ABE pour chiffrer

le contenu des publications, le schéma ABKS-UR pour révoquer des souscripteurs et chiffrer le topic des publications/souscriptions et enfin le schéma ABS pour permettre aux publieurs de s'authentifier via leurs publications et assurer l'intégrité des données qu'ils envoient. Ces schémas assurent l'ensemble de ces propriétés de sécurité sans dépendre ni d'un proxy ni d'une liste de révocation.

Les principales étapes qui permettent le bon fonctionnement de notre modèle de sécurité sont présentées dans la Figure 4.2. On se base sur le scénario présenté en Section 2.4.1 pour illustrer son fonctionnement.

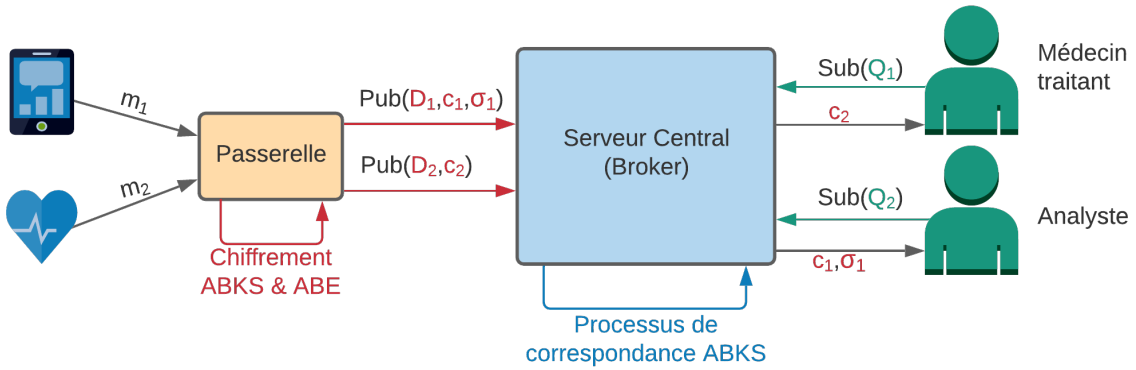


FIGURE 4.2 – Solution de sécurité pour environnement AAL

Dans l'habitat du patient, de nombreux capteurs sont connectés à la passerelle de l'habitat et lui envoient des données. Nous prenons l'exemple ici de deux capteurs, un application podomètre sur le téléphone du patient envoyant la donnée m_1 et un capteur cardiaque connecté au patient envoyant la donnée m_2.

De nombreux professionnels de santé peuvent souscrire à des données sur le broker. Nous prenons l'exemple ici de deux souscripteurs, le médecin traitant ainsi qu'un analyste. Ils disposent chacun d'une clé secrète ABKS-UR et d'une clé secrète ABE. Ils utilisent dans un premier temps leur clé secrète ABKS-UR pour générer les trapdoors Q_1 pour le médecin et Q_2 pour l'analyste. Ils effectuent ensuite une souscription au broker. Dans notre exemple, on imagine que le médecin est intéressé par le topic concernant le capteur cardiaque et l'analyste par le podomètre. Le médecin traitant a pour rôle de détecter une anomalie cardiaque tandis que l'analyste utilise les données relatives à l'activité quotidienne du patient pour y détecter des anomalies. Ces souscripteurs sont en attente de nouvelles publications correspondant à leurs intérêts.

À chaque nouveau message reçu par d'un capteur, la passerelle génère le chiffré du topic, qu'on note D, ainsi que le chiffré du contenu du message, noté c. Pour effectuer ces chiffrements, il utilise les clés maîtresses publiques ABKS-UR et ABE qui sont stockées sur la passerelle. La publication (D, c) est envoyée au broker.

Lorsque le broker reçoit une nouvelle publication, il utilise le processus de correspondance ABKS-UR entre le nouveau topic chiffré D et chaque trapdoor Q qu'il conserve dans sa base de données. Lorsqu'une publication correspond à une trapdoor, il transmet le contenu chiffré de la publication c au souscripteur correspondant. Dans notre exemple, la donnée chiffrée c_1 du podomètre est bien transmise à l'analyste tandis que la donnée chiffrée du capteur cardiaque c_2 est transmise au médecin. Ce processus de correspondance n'a lieu que lors de la première publication dans un topic, il n'est pas répété lors de la énième publication d'un message dans un topic déjà existant. Une fois le chiffré reçu, chaque souscripteur utilise sa clé SK_{ABE} pour retrouver le message m correspondant.

À noter que si un utilisateur ne possède pas les droits pour accéder aux données d'un topic, le broker lui répond de la même manière que si ce topic n'existait pas. Cela renforce la confidentialité sur le nom du topic.

Pour permettre aux patients de s'authentifier anonymement au travers de ses publications, il peut configurer sa passerelle pour ajouter une signature ABS à ses publications. Dans notre exemple, la passerelle envoie également la signature σ_1 attaché à la publication (D_1, c_1). Cette signature permet au patient de s'authentifier avec ses attributs sans révéler son identité, ce qui permet à l'analyste de traiter les données anonymement, tout en étant certain de leur intégrité.

Dans cette solution, les schémas ABE et ABKS-UR définissent les utilisateurs avec le même univers d'attributs pour renforcer l'homogénéité de notre modèle.

4.5 Modification du schéma ABKS-UR

Dans notre solution de sécurité, le schéma ABKS-UR proposé par Sun et al. [127] est utilisé pour chiffrer le topic des publications ainsi que le topic des souscriptions. Il permet également la révocation de souscripteurs en temps réel. Cependant le schéma original est conçu pour fonctionner avec une architecture où le producteur de données doit conserver la liste des consommateurs

accédant à ses donnés.

Or, dans notre solution de sécurité pour architecture topic-based publish/subscribe, nous avons pour objectif de conserver un couplage faible entre publieurs et souscripteurs. Nous avons donc effectué plusieurs changements au schéma ABKS-UR proposé par Sun et al. Ces changements sont présentés dans cette section et mis en parallèle avec la construction originale. À noter que l'ensemble des algorithmes de [127] concernant la révocation d'utilisateurs n'ont pas été modifiés.

4.5.1 Application bilinéaire

Soit $\mathbb{G}$, $\mathbb{G}_T$ deux groupes cycliques d'ordre un grand nombre premier p. Avec g un générateur de $\mathbb{G}$. L'opération de couplage $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ désigne l'application bilinéaire. Cette application bilinéaire possède les propriétés suivantes :

— Bilinéarité : Pour tout $g, h \in G, a, b \in \mathbb{Z}_p$, on a $e(g^a, h^b) = e(g, h)^{ab}$.

— Non-dégénérescence : $e(g, g) \neq 1 \in \mathbb{G}_T$.

— Calculabilité : L'opération $e(g, h)$ est facilement calculable, avec $g, h \in \mathbb{G}$.

4.5.2 Construction de Sun et al.

Dans la construction de Sun et al., les utilisateurs sont mentionnés en tant que producteurs et consommateurs de données. Chaque producteur conserve une liste des consommateurs qui ont accès à leurs données. Le schéma est organisé en 5 étapes : **System setup**, **New user enrollment**, **Secure index generation**, **Trapdoor generation** et **Search**. Le nom de ces étapes sont conservées en anglais puisqu'elles sont issues du schéma original.

System setup : Lors de la mise en place du système, la TA définit un ensemble d'attributs, appelé également univers d'attributs, composé de n éléments : $\mathcal{N} = \{1, ..., n\}$. Elle effectue ensuite les opérations suivantes :

— Génère un ensemble $\{t_1, ..., t_{3n}\}$ de $3n$ éléments aléatoires $\in \mathbb{Z}_p$.

— Définie une fonction de hachage résistante à la collision H avec une clé secrète associée.

— Calcule $T_k = g^{t_k}$ pour $k \in \{1, ..., 3n\}$ avec $g \in \mathbb{G}$.

— Calcule $Y = e(g, g)^y$ avec $y \xleftarrow{\$} \mathbb{Z}_p^*$.

La TA publie la clé maîtresse publique $PK = (ver, g, Y, T_1, ..., T_{3n})$ et conserve la clé maîtresse secrète $MK = (ver, y, t_1, ..., t_{3n})$ avec ver étant la version de la clé, initialisée à 1 (cette version servira de pré-contrôle pour éviter les calculs inutiles avec des clés non mises à jour).

New user enrollment : Lorsqu'un consommateur de données ID_f veut rejoindre le système, il effectue une requête à la TA pour une génération de clé secrète (SK) associée à un ensemble d'attributs $\mathcal{A}$. Pour générer cette clé, la TA effectue les opérations suivantes :

— Sélectionne un élément aléatoire $x_f \in \mathbb{Z}_p$ en tant que composant de la clé MK et calcule son équivalent publique $X_f = Y^{x_f}$.

— Sélectionne un élément aléatoire r_i pour chaque $i \in \mathcal{N}$ et calcule $r = \sum_{i=1}^{n} r_i$.

— Calcule $\widehat{K} = g^{y-r}$ et si $i \in \mathcal{A}$ calcule $K_i = g^{\frac{r_i}{t_i}}$, sinon $K_i = g^{\frac{r_i}{t_{n+i}}}$.

— Calcule $F_i = g^{\frac{r_i}{t_{2n+i}}}$.

La clé secrète $SK = (ver, x_f, \widehat{K}, \{K_i, F_i\}_{i \in \mathcal{N}})$ est envoyée au consommateur par la TA.

Le producteur de données calcule $D_f = {X_f}^{-s}$ avec $s \xleftarrow{\$} \mathbb{Z}_p^*$ et ajoute le tuple (ID_f, D_f) à sa liste de consommateurs.

Secure index generation : Lorsqu'un producteur veut envoyer des données sur le serveur distant, il définit tout d'abord un ensemble de mots clés W, puis :

— Définit une structure d'accès GT qui contient des attributs dit « positifs » s'ils doivent être présents dans la clé générant la trapdoor et « négatifs » s'ils doivent être absents. Ils sont séparés par des clauses *AND*.

— Calcule $\widehat{D} = g^s$, $\widetilde{D} = Y^s$.

— Pour chaque $i \in GT$, calcule $D_i = T_i^s$ pour les attributs positifs et $D_i = T_{n+i}^s$ pour les négatifs.

— Calcule $D_i = T_{2n+i}^s$ pour chaque $i \notin GT$.

— Choisit un attribut positif i' et calcule $D_{i'}$ tel que $D_{i'} = T_{i'}^{\overline{\prod_{w_j \in W}^{s} H(w_j)}}$, où H est une fonction de hachage résistante à la collision $\{0,1\}^* \to \mathbb{Z}_p$.

L'index chiffré est donc défini tel que D :

$$D := (ver, GT, \widehat{D}, \widetilde{D}, \{D_i\}_{i \in \mathcal{N}}). \tag{4.1}$$

Trapdoor generation : Lorsqu'un consommateur de données veut effectuer une requête au serveur, il doit générer une trapdoor Q avec sa clé secrète SK et un ensemble de mots clés W. Ensuite, il effectue les opérations suivantes :

— Génère un élément aléatoire $u \in \mathbb{Z}_p$, calcule $\widehat{Q} = \widehat{K}^u$ et $\widetilde{Q} = u + x_f$.

— Pour chaque $i \in \mathcal{N}$, calcule $Q_i = K_i^u$ et $Qf_i = F_i^u$.

— Pour l'indice i', calcule $Q_{i'} = (K_{i'}^{\prod_{w_j \in W} H(w_j)})^u$ et $Qf_{i'} = (F_{i'}^{\prod_{w_j \in \mathcal{W}} H(w_j)})^u$.

La trapdoor générée est donc définie telle que $Q = (ver, \widehat{Q}, \widetilde{Q}, \{Q_i, Qf_i\}_{i \in \mathcal{N}})$.

Search : Cet algorithme permet au serveur de comparer un index chiffré D avec une trapdoor Q.

Plus spécifiquement, il vérifie tout d'abord si l'identité ID_f du consommateur est présente dans la liste d'utilisateurs. Si elle ne l'est pas, l'opération échoue.

Sinon, le serveur vérifie la version de la trapdoor :

— Si la version de Q est inférieure à la version de D, renvoie $\perp$ car la clé servant à générer la trapdoor n'était pas à jour.

— Si la version de Q est supérieure à la version de D, le mécanisme de lazy re-encryption est appliqué, c'est-à-dire dès que le chiffrement s'effectue lorsque l'index est consulté pour la première fois depuis la mise à jour des clés. Ainsi, le serveur le re-chiffre avec la nouvelle clé publique ABKS-UR mise à jour.

— Si la version de Q est égale à la version de D, le processus de correspondance peut continuer.

Pour chaque attribut i, si $i \in GT$, calcule $e(D_i, Q_i)$; sinon calcule $e(D_i, Qf_i)$. Enfin, on vérifie :

$$\widetilde{D}^{\widetilde{Q}} \cdot D_f \stackrel{?}{=} e(\widehat{D}, \widehat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*) \tag{4.2}$$

Où $Q_i^* = Q_i$ si $i \in GT$ et $Q_i^* = Qf_i$ sinon.

Si l'équation 4.2 est vérifiée, la correspondance entre l'index D et la trapdoor Q est valide.

4.5.3 Modifications pour un environnement distribué

Ces modifications ont pour objectif d'éviter que les publieurs conservent une liste de souscripteurs qui s'abonnent à leurs données, afin de conserver un couplage faible entre eux. Pour

modifier le schéma ABKS-UR afin qu'il corresponde à cet environnement distribué, nous avons modifié les quatre étapes suivantes : **System setup**, **Secure index generation**, **New user enrollment** and **Search**.

System setup : La quatrième étape a été modifiée afin de supprimer l'opération de couplage. Y a été modifié tel que $Y = g^y$.

Secure index generation : Nous introduisons un identifiant cryptographique basé sur une opération de couplage DT tel que $DT = e(g, \widetilde{D})$. DT est utilisé dans l'algorithme **Search**. L'équation 4.1 est modifiée tel que D contienne DT :

$$D := (ver, GT, \widehat{D}, \widetilde{D}, DT, \{D_i\}_{i \in \mathcal{N}}) \tag{4.3}$$

New user enrollment : La première étape est modifiée, X_f se calcule désormais : $X_f = g^{-x_f}$. D_f a également été modifié, tel que : $D_f = e(\widetilde{D}, X_f)$. Le tuple (ID_f, D_f) est envoyé au broker et ajouté à sa liste d'utilisateurs. Dans notre schéma modifié, c'est le broker qui conserve la liste d'utilisateurs souscrivant aux topics qu'il maintient.

Search : Cette étape est modifiée telle que l'équation 4.2 s'exprime maintenant : $DT^{\widetilde{Q}} \cdot D_f \stackrel{?}{=} e(\widehat{D}, \widehat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*)$

4.5.4 Exactitude du schéma modifié

Si le souscripteur possède bien les attributs qui correspondent à la politique d'accès de l'index, et que les mots clés correspondent également, nous avons :

— Le souscripteur possédant : $x_f, \widehat{K} = g^{y-r}, K_i = g^{\frac{r_i}{t_i}}, F_i = g^{\frac{r_i}{t_{2n+i}}}$

— Pour ce souscripteur, la TA génère $D_f = e(g, g)^{yx_f}$, qui est envoyée au broker

— L'index chiffré généré par le souscripteur contient : $\widehat{D} = g^s, \widetilde{D} = Y^s, D_i = T_i^s, D_{i'} = T_{i'}^{\frac{s}{\prod_{w_j \in W} H(w_j)}}$

Le processus de recherche vérifie la correspondance :

— Premièrement, avec l'index $\widetilde{D}^{\widetilde{Q}}$ et D_f :

$$\begin{aligned} DT^{\widetilde{Q}} \cdot D_f &= e(g, \widetilde{D})^{u+x_f} \cdot e(\widetilde{D}, X_f) \\ &= e(g,g)^{y \cdot s \cdot (u+x_f)} \cdot e(g,g)^{-y \cdot s \cdot x_f} \\ &= e(g,g)^{s \cdot u \cdot y} \end{aligned}$$

— Puis, avec les trapdoors $\widehat{Q}, Q_i^*$ générées par le souscripteur

$$\begin{aligned} e(\widehat{D}, \widehat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*) &= e(g^s, g^{u \cdot y - u \cdot r}) \cdot \prod_{i=1}^{n} e(g,g)^{s \cdot u \cdot r_i} \\ &= e(g,g)^{s \cdot u \cdot y - s \cdot u \cdot r} \cdot e(g,g)^{s \cdot u \cdot r} \\ &= e(g,g)^{s \cdot u \cdot y} \\ &= DT^{\widetilde{Q}} \cdot D_f \end{aligned}$$

Ces équations démontrent que la correspondance entre un chiffré et une trapdoor se vérifie si l'ensemble d'attributs attachés à la trapdoor valide la politique d'accès du chiffré et que les mots clés correspondent. Cela prouve l'exactitude de notre ABKS-UR modifié.

4.5.5 Preuve de la sécurité sémantique des mots clés

Le simulateur reçoit un challenge Twin Diffie–Hellman (TDH) $A = g^a, B = g^b, C = g^c, Z \in \mathbb{G}$.

L'adversaire envoie au simulateur une politique d'accès challenge $GT = \bigwedge_{i \in I} i'$ et un ensemble d'attributs $\mathcal{A}$.

Mise en place

Le simulateur définit $PK = e(A, B)$, $MK = ab$.

Le simulateur choisit un nombre aléatoire $x \in \mathbb{Z}_p$ et calcule $Y' = PK^x$.

Pour chaque $i \in \mathcal{N}$, il choisit les éléments aléatoires $\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_p$ et pour chaque $i \in I$:

Si $i' = i, T_i = g^{\alpha_i}, T_{n+i} = B^{\beta_i}, T_{2n+i} = B^{\gamma_i}$; $T_i = B^{\alpha_i}, T_{n+i} = g^{\beta_i}, T_{2n+i} = B^{\gamma_i}$ sinon.

Pour chaque $i \notin I, T_i = B^{\alpha_i}, T_{n+i} = B^{\beta_i}, T_{2n+i} = g^{\gamma_i}$;

Pour chaque ensemble d'attributs Φ^j dans $\mathcal{A}$ fourni par l'adversaire, le simulateur génère $rk^{(j)}$, ou lui donne la valeur 1 sinon.

Phase 1

Si un adversaire envoie une requête pour un mot clé w, ainsi qu'un ensemble d'attributs S différent de la politique GT, sous réserve de la propriété de non collision de la fonction de hachage, il existe un index k tel que $k \in S\Delta\mathcal{A}$ (où Δ est la différence symétrique). Tout comme [127], on peut se servir de cet index pour introduire un élément parfaitement aléatoire dans la partie Qf_k de la trapdoor.

Challenge

Après avoir reçu les mots clés challenge w_0, w_1 de l'adversaire, le simulateur choisit un bit aléatoire b, et chiffre w_b avec la politique d'accès GT. Le simulateur peut ensuite définir $\widehat{D} = C, \widetilde{D} = Z$ and $D_f = e(Z^{-x}, g)$.

C'est la différence principale par rapport à la preuve originale, puisque à cette étape la valeur Z est un élément de $\mathbb{G}_1$ dans le schéma original, tandis que dans notre schéma modifié, il est dans $\mathbb{G}$.

Phase 2

Le simulateur répond aux requêtes post-challenge exactement comme avant (l'opération est abandonnée si la requête cible la politique GT).

Wrapping Up L'adversaire donne sa réponse par rapport à la valeur b.

Si Z est généré honnêtement, l'adversaire voit le vrai jeu.

Si Z est aléatoire, c'est pour l'adversaire indistinguable par rapport à de l'aléatoire.

Donc l'avantage de l'adversaire contre la sécurité sémantique des mots clés est plus petit que l'avantage du simulateur contre TDH ($\mathsf{Adv}^{kss} \leq \mathsf{Adv}^{\mathsf{TDH}}$).

4.6 Expérimentations

Dans cette section, la complexité théorique ainsi que les résultats expérimentaux de notre solution de sécurité sont décrits et illustrés. Puisqu'une des principales particularités de notre

solution est l'ensemble de modifications apportées au schéma ABKS-UR, une grande partie de nos expérimentations se concentrent sur ce point.

4.6.1 Complexité théorique

La Table 4.1 présente la complexité théorique du schéma original ABKS-UR par rapport à notre version modifiée. Les notations sont les suivantes : n correspond au nombre d'attributs, P désigne les opérations de couplage, les exponentiations modulaires sont notés E dans le groupe $\mathbb{G}$ et E_1 dans le groupe $\mathbb{G}_1$. Pour finir, les opérations de multiplication sont notées M dans $\mathbb{G}$ et M_1 dans $\mathbb{G}_1$.

Nous avons donc ajouté une opération de couplage aux étapes **New user enrollment** et **Secure index generation** d'une part, et supprimé l'opération de couplage de **System setup** d'autre part. Ces étapes s'effectuant un nombre de fois limité lors de l'utilisation pratique de la solution, l'impact d'un couplage supplémentaire sur l'efficacité globale du schéma reste négligeable.

Étape	**ABKS-UR** [127]	**Version modifiée**
System setup	$3nE + E_1 + P$	$(3n+1)E$
New user enrollment	$(2n+1)E + 2E_1$	$(2n+2)E + P$
Secure index generation	$(n+1)E + E_1$	$(n+1)E + E_1 + P$
Trapdoor generation	$(2n+1)E$	$(2n+1)E$
Search	$(n+1)P + (n+2)M_1 + E_1$	$(n+1)P + (n+2)M_1 + E_1$

TABLE 4.1 – Complexité théorique ABKS-UR

4.6.2 Résultats expérimentaux

L'objectif de cette section est de mettre en valeur l'efficacité et la praticabilité de notre solution de sécurité dans un scénario pratique : l'environnement d'aide à l'autonomie à domicile. Ces expérimentations ont été mises en places sur des machines virtuelles disposant de 4 Go de RAM et d'un processeur Intel Core i7 de 2.6 GHz pour les souscripteurs, le broker MQTT et la TA. Les publieurs ont été simulés à l'aide d'un Raspberry Pi 3 model B, représentant la passerelle entre les objets connectés du patient et le serveur distant (broker). Puisque nous supposons que cette passerelle est alimentée sans difficulté et de manière continue, la consommation d'énergie

n'est pas évaluée ici.

Le broker MQTT est implémenté avec le broker open source HBMQTT [99] qui a été modifié au niveau du processus de correspondance entre une publication et une souscription pour y ajouter la partie ABKS-UR de notre modèle.

Pour les publieurs et les souscripteurs, nous utilisons la librairie Eclipse Paho MQTT Python [100]. Cette dernière permet les opérations de publications/souscriptions vers le broker depuis un programme Python.

Enfin, les cryptosystèmes utilisés sont implémentés à l'aide de la librairie Charm-crypto Python [4]. Nous utilisons les fonctions déjà implémentés permettant l'usage du schéma CP-ABE proposé par Waters [135] : Charm Waters'09. Les schémas ABKS-UR et ABS n'étant pas présents de base dans la librairie, nous les avons implémentés en Python 3 en utilisant la courbe elliptique *SS512* disponible sur Charm. Cette dernière représente une courbe elliptique symétrique de 512 bits [117].

Pour des questions de performances, le chiffrement ABE est utilisé pour chiffrer une clé AES 256, qui est ensuite utilisée pour chiffrer la donnée (c'est-à-dire le contenu de la publication). Nous effectuons nos mesures en faisant varier différents paramètres qui sont présentés sur les figures relatives aux évaluations de performances.

Comme illustré dans la Figure 4.3a, lors de la mise en place du système initié par la TA, le temps d'exécution pour générer les clés maîtresses ABKS-UR augmente de façon linéaire par rapport au nombre d'attributs présents dans l'univers, jusqu'à atteindre 390 ms pour 100 attributs. Ce résultat est raisonnable dans un scénario où le nombre total d'attributs sera en général moins élevé. On peut également observer que la génération de clés maîtresses pour les schémas de CP-ABE et ABKS sont de 10 ms, ce qui est négligeable. Ces opérations ne sont effectuées qu'une seule fois durant la mise en place du système.

Les performances des étapes d'ajout d'un nouveau souscripteur et d'ajout d'un nouveau publieur sont présentées en Figure 4.3b. Lancé sur la TA, la génération d'une clé secrète ABKS-UR contenant 100 attributs prend 275 ms et celle d'une clé secrète CP-ABE 55 ms. Pour l'ajout d'un nouveau publieur, la génération de la clé secrète du schéma ABS prend 780 ms. Ces opérations ne sont effectuées qu'une seule fois par utilisateur.

Lors de l'étape de Publication, nous évaluons le temps de calcul pour : (1) chiffrement du

contenu de la publication avec CP-ABE, (2) chiffrement du topic avec ABKS-UR et (3) signature de la publication avec ABS. Comme illustré sur la Figure 4.3c, le temps d'exécution de (1) est constant, aux environs de 1100 ms. On observe également que les temps d'exécution de (2) et (3) sont linéaires en fonction du nombre d'attributs dans l'univers. Par exemple, pour 100 attributs dans l'univers, (2) est aux environs de 1900 ms et (3) 12000 ms, cette dernière opération étant optionnelle.

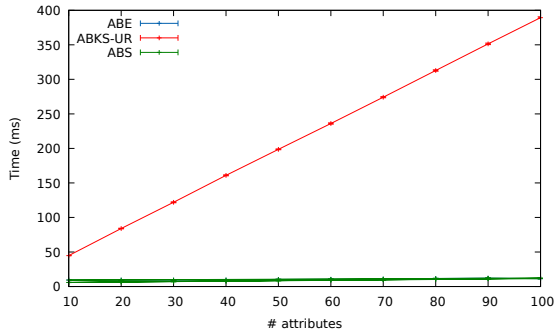
Comme illustré sur la Figure 4.3d, la taille totale de la publication est de 37 ko pour un univers de 100 attributs (4 ko pour le chiffré ABE, 15 ko pour le topic chiffré en ABKS-UR et 18 ko pour la signature ABS). À noter que le chiffré ABKS-UR est de taille proportionnelle au nombre d'attributs dans l'univers.

Nous avons également évalué les performances en conservant le nombre d'attributs dans l'univers à 100 et en faisant varier le nombre d'attributs dans la politique d'accès. Comme illustré sur la Figure 4.3e, le chiffrement ABE prend 9360 ms, 1950 ms pour le chiffrement du topic avec ABKS-UR et 12000 ms pour la signature ABS, avec une politique d'accès contenant 100 attributs. Ces valeurs n'ont pas pour objectif de montrer le temps d'exécution du schéma dans un cadre réaliste mais plutôt d'explorer les limites de ces schémas.

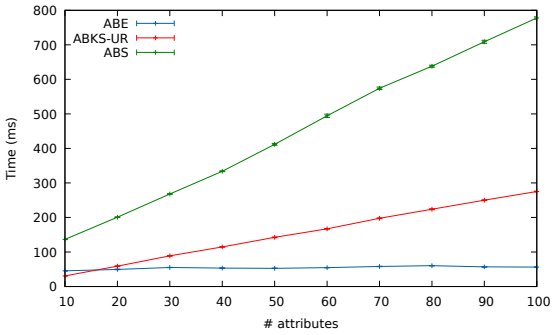
La taille totale des parties de la publication est illustrée dans la Figure 4.3f. Elle est de 60 ko, avec 30 ko pour le chiffré ABE, 15 ko pour le topic chiffré en ABKS-UR et 15 ko pour la signature ABS. Le topic chiffré ABKS-UR a une taille constante quel que soit le nombre d'attributs, tandis que la taille de la signature est dégressive. Cela s'explique parce qu'une partie de la signature contient des éléments relatifs aux attributs ne faisant pas partie de la politique associée à la signature. Ainsi, plus la politique possède d'attributs, moins la signature contient d'éléments.

Les performances relatives à la souscription sont illustrées sur la Figure 4.3g. Le temps de génération de la trapdoor augmente de manière linéaire en fonction du nombre d'attributs dans l'univers. Cette opération est effectuée une seule fois par topic et prend environ 280 ms pour 100 attributs.

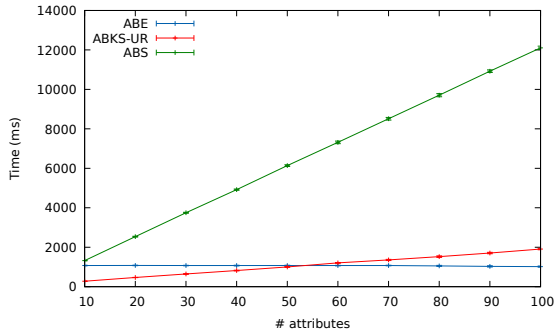
Enfin, au niveau du broker, le temps pour effectuer le processus de correspondance de la solution de sécurité augmente linéairement par rapport au nombre d'attributs dans l'univers, comme présenté sur la Figure 4.3g. Cette opération prend seulement 85 ms pour 100 attributs, ce qui est très satisfaisant pour un scénario réel.



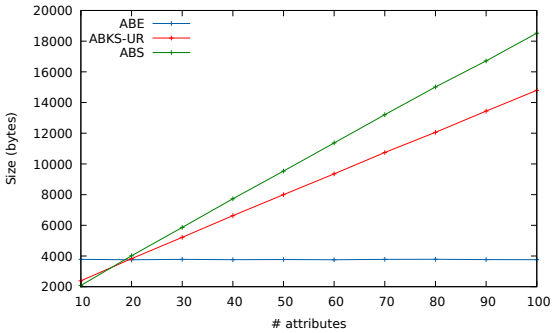
(a) Mise en place du système



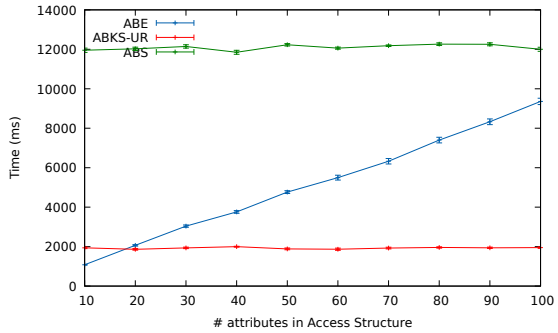
(b) Ajout d'un nouveau utilisateur (publieur/souscripteur)



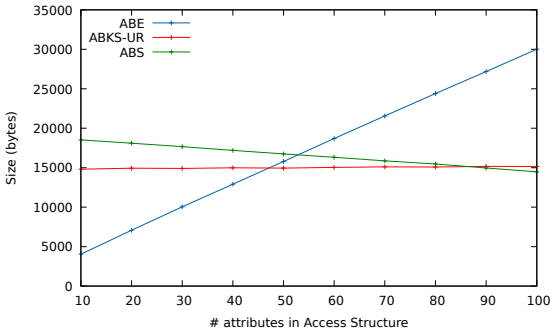
(c) Évaluation du temps de publication avec une variation du nombre d'attributs dans l'univers et une politique d'accès contenant 10 attributs



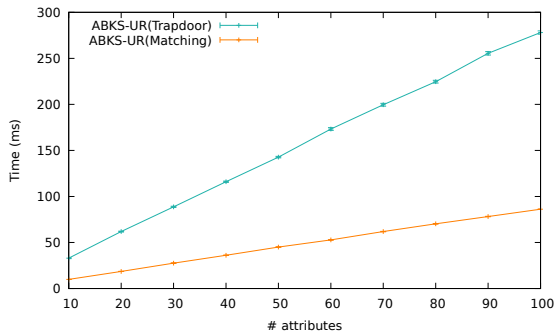
(d) Évaluation de la taille d'une publication avec une variation du nombre d'attributs dans l'univers et une politique d'accès contenant 10 attributs



(e) Évaluation du temps de publication avec une variation du nombre d'attributs dans la politique d'accès et un univers de 100 attributs



(f) Évaluation de la taille d'une publication avec une variation du nombre d'attributs dans la politique d'accès et un univers de 100 attributs



(g) Génération de trapdoor ABKS-UR et processus de correspondance

FIGURE 4.3 – Évaluation des performances

4.7 Conclusion du chapitre

Dans ce chapitre, nous avons présenté une solution de sécurité pour une architecture topic-based publish/subscribe, instancié avec MQTT. Notre objectif était de garantir une protection de la vie privée des utilisateurs ainsi la confidentialité de leurs données.

Pour cela, nous avons pris en compte les différentes spécificités de l'architecture topic-based publish/subscribe. Nous avons défini chacune des différentes entités ainsi que leurs rôles. Après cela, nous avons appliqué le scénario décrit en Section 2.4.2 à cette architecture afin d'associer chaque entité avec un acteur du scénario.

Une fois l'architecture et le scénario bien définis, nous avons établi un modèle de sécurité pour cette architecture. Nous avons ensuite établis différentes exigences de sécurité de notre solution ainsi qu'un modèle de menaces.

En se basant sur l'ensemble des prédicats définis précédemment, nous avons décrit chaque phase de notre solution de sécurité en détaillant l'ensemble du protocole ainsi que les mécanismes de sécurité utilisés.

Nous avons également modifié la construction du schéma ABKS-UR de Sun et al. pour le rendre fonctionnel dans une architecture publish/subscribe. Nous prouvons la sécurité ainsi que l'exactitude du schéma modifié.

Pour finir, nous présentons un ensemble de résultat expérimentaux de cette solution, qui démontrent la praticabilité de ces mesures de sécurité dans un scénario réaliste.

Nous avons donc proposé une solution de sécurité efficace dans un scénario AAL, avec des données qui sont transmises depuis plusieurs capteurs positionnés dans un habitat vers plusieurs professionnels de santé. Notre objectif est maintenant de considérer un scénario à plus grande échelle, avec beaucoup plus d'utilisateurs.

Chapitre 5

Solution de sécurité pour une architecture Apache Kafka

Dans ce chapitre, nous présentons notre solution de sécurité pour l'architecture de streaming d'évènements Apache Kafka. Cette architecture est une implémentation du modèle publish/subscribe, présenté en Section 2.3, qui permet aux utilisateurs de conserver les données sur le broker pendant une période prédéfinie. Cette fonctionnalité permet une communication asynchrone entre les utilisateurs ainsi qu'une re-consommation de la donnée. Elle permet également de gérer un plus gros volume de données. Apache Kafka est donc une architecture de broker de stockage, qui se distingue de MQTT qui est une architecture de broker relais, où les données ne sont pas conservées sur le broker. Cette solution permet de répondre notamment au scénario présenté en Section 2.4.2, pour protéger les données des patients stockées et transitant par le broker chez l'hebergeur des données de santé.

Sommaire

5.1 Apache Kafka et Apache Zookeeper

Avant de présenter la solution en elle-même, nous présentons dans cette section les spécificités de l'architecture Apache Kafka et ainsi que le rôle d'Apache Zookeeper dans cette dernière.

5.1.1 Apache Kafka

Apache Kafka [95] est un projet open source développé par Apache Software Foundation. Il met en place une architecture publish/subscribe qui permet aux utilisateurs d'effectuer des publications/souscriptions au travers d'un ensemble de brokers où les données sont répliquées et distribuées.

Apache Kafka fonctionne avec Apache Zookeeper [98], qui est un service composé d'un ou plusieurs serveurs gérant des métadonnées pour Apache Kafka, notamment la gestion des topics

ainsi que les souscriptions des utilisateurs. Il permet d'externaliser, entre autre, le processus de correspondances entre publications et souscriptions hors d'Apache Kafka, afin de réduire sa charge de travail. Le fonctionnement d'Apache Zookeeper est détaillé dans la Section 5.1.2. À noter que certains projets cherchent à supprimer cette dépendance mais sont encore à l'état de test [91], c'est pourquoi nous considérons encore Apache Kafka fortement lié à Apache Zookeeper.

Apache Kafka est organisé en un cluster, qui regroupe plusieurs serveurs. Ces serveurs, appelés brokers, permettent la transmission des données envoyées par les publieurs vers les souscripteurs. Les données sont stockées sur les brokers (pendant une période de temps prédéfinie) et sont organisées en topics, eux-mêmes divisés en partitions qui sont répliquées à travers les différents brokers de manière à assurer la disponibilité permanente des données, même en cas de défaillance d'un broker. On parle de tolérance aux fautes. Un exemple de partitions est présenté en Figure 5.1 pour un topic qui possède une tolérance aux fautes de 1, c'est-à-dire qu'elle supporte la perte d'un broker. En effet, dans le cas d'une défaillance du broker C, la partition P3 est indisponible mais le réplica RP3 du broker B prend le relais et garantit l'accessibilité des données.

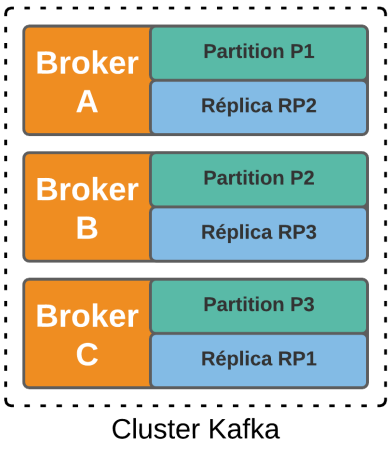


FIGURE 5.1 – Exemple de réplication de partition dans un cluster Kafka

Chaque partition peut donc posséder plusieurs réplicas, et l'un d'eux est sélectionné comme leader. Cela signifie que lorsqu'un publieur publie une donnée sur une partition, il le fait sur la partition leader. Les autres partitions, qu'on appelle suiveurs, suivent le statut de la partition leader et la réplique dés qu'ils sont notifiés. Le serveur Zookeeper gère et conserve dans sa base de donnée la statut de chaque partition. Zookeeper gère donc les suiveurs ainsi que les In-sync-Replicas (ISR), c'est-à-dire l'ensemble des partitions réplicas qui sont à jour par rapport au

leader. Dans notre exemple, si un publieur cherche à ajouter une donnée sur la partition P1, il devra contacter le broker A pour écrire sur P1. Ensuite, le réplica suiveur RP1 situé sur le broker B sera notifié par Zookeeper des changements de données sur la partition P1 et appliquera les modifications en conséquence. Une fois ces modifications appliquées, il est considéré comme un ISR puisqu'il est synchronisé avec le leader.

Les données ne restent pas stockées sur les brokers indéfiniment. En effet, les données disparaissent après une certaine période de temps, définie à l'avance, ou les topics sont limités par une taille plafond. Dans ce cas, les données les plus anciennes sont effacées au profit des données les plus récentes lorsque ce plafond est atteint. Puisque les données sont conservées un certain temps sur les brokers, il est possible pour les souscripteurs de requérir plusieurs fois les données. De plus, ce stockage des données permet aux souscripteur de consulter les données même s'ils étaient hors ligne lors de la publication de celles-ci.

Pour finir, chaque donnée publiée ou lue sur les brokers est appelée un log. Chaque log correspond à une nouvelle publication envoyée par un publieur. Les brokers organisent les logs dans l'ordre dans lequel ils arrivent : premier entré, premier sorti (PEPS). Chaque log possède donc un index qui permet aux utilisateurs de connaître son statut en lecture ou en écriture lorsqu'ils communiquent avec le broker. Cela leurs permet de savoir quel log ils ont consulté ou ajouté lors de leur dernier accès au broker.

Le processus de communication, présenté en Figure 5.2, peut se résumer comme suit : le publieur notifie un serveur Zookeeper de la création d'un nouveau topic (1). Ensuite, le publieur envoie une publication sur le cluster de brokers, celle-ci étant désignée par le topic précédemment notifié (2). Lorsqu'un souscripteur cherche à récupérer des données sur un broker Kafka, il effectue tout d'abord une souscription au serveur Zookeeper, qui lui confirme que le topic existe et éventuellement quel broker contacter en premier (3). Enfin, le souscripteur requête le broker en lecture sur le topic de son choix (4) et reçoit ainsi les données correspondantes (5).

Les publieurs ainsi que les souscripteurs utilisent donc Zookeeper pour rechercher les topics existants et savoir quel(s) broker(s) contacter en fonction de leurs intérêts. Ils peuvent également interroger Zookeeper pour connaître leur historique de consommation de données (c'est-à-dire retrouver le dernier index lu).

Pour résumé, Zookeeper conserve les métadonnées du système et gère également en partie

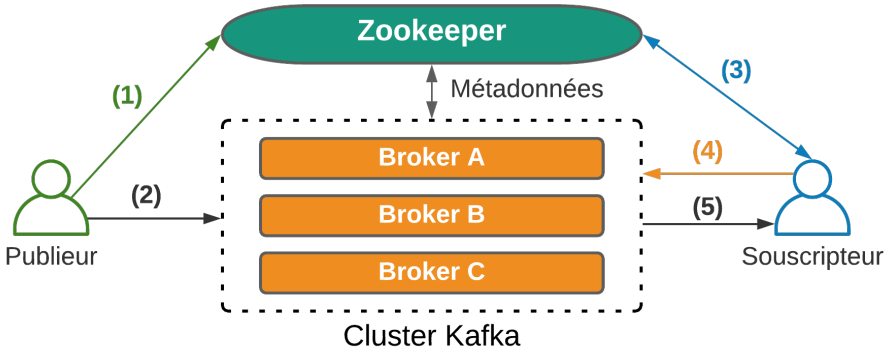


FIGURE 5.2 – Architecture Kafka - Zookeeper

la transmission d'informations au niveau des correspondances entre topics et souscriptions. Les serveurs Kafka ne conservent au final que les publications. C'est pourquoi nous détaillons dans la section suivante le fonctionnement d'Apache Zookeeper, qui sera un des composant clés à sécuriser dans notre solution de sécurité, au vu des informations sensibles auxquelles il a accès.

5.1.2 Apache Zookeeper

Apache Zookeeper est un projet open source qui permet de coordonner la configuration, la synchronisation, la gestion de groupe et l'élection de leader dans Apache Kafka. Tout comme ce dernier, il peut garantir la disponibilité permanente des données, même en cas de défaillance d'un serveur, puisque les données sont répliquées à travers plusieurs serveurs Zookeeper. À noter que Apache Zookeeper est également utilisé avec d'autres projets comme Apache Hadoop [93] ou Apache Hive [94]. Nous nous concentrons néanmoins ici sur son rôle lorsqu'il est utilisé avec Apache Kafka.

Les données stockées par les serveurs Zookeeper sont organisées sous forme d'arbre hiérarchique, composé de différents nœuds, appelés Znodes. Chaque Znode se compose :

— D'un nom : symbolisé par son identifiant ainsi que ses nœuds parents.

— D'un contenu : optionnel, qui peut être de taille variable.

Un Znode peut également avoir plusieurs nœuds fils. Un exemple de cette structure de données est illustré en Figure 5.3. Dans cet exemple, le Znode "/topics" possède comme parent le Znode root "/" et les fils "/topics/A" et "/topics/B". Il ne possède pas de contenu, contrairement à ses nœuds fils où l'adresse IP du broker attaché à ces topics est donnée.

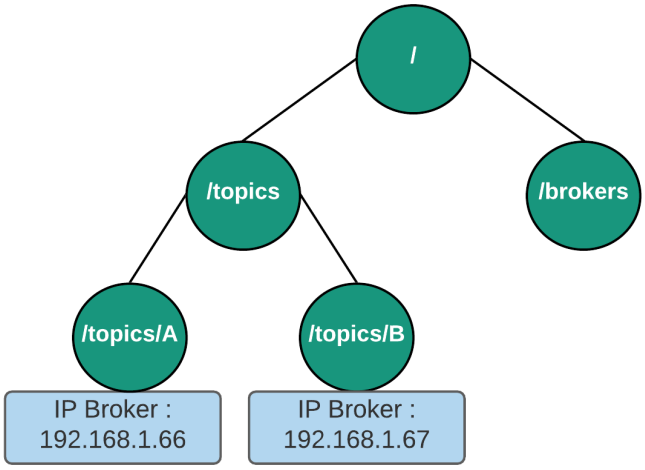


FIGURE 5.3 – Exemple de Znodes dans Apache Zookeeper

Zookeeper permet un certain nombre d'opérations sur ces Znodes, parmi lesquels : Get, Set, Create et Delete, pour respectivement lire, modifier, créer et supprimer un Znode. Il est également possible de récupérer tous les nœuds fils d'un Znode avec la commande GetChildren. Lorsqu'il est combiné avec Apache Kafka, Zookeeper gère un certain nombre d'informations :

— Le statut des brokers : Zookeeper maintient une liste de tous les brokers Kafka du système.

— Les topics : Zookeeper conserve la configuration des topics ainsi que l'emplacement de leurs réplicas.

— L'élection des leaders : Zookeeper maintient un service permettant de changer de leader de partition si un serveur est mis hors ligne de manière inattendue. Il conserve également la localisation de chaque réplica de cette partition. Cela assure qu'à chaque instant, il n'y a qu'une seule partition leader.

Pour résumé, Zookeeper conserve les métadonnées du système et gère également en partie la transmission d'informations au niveau des correspondances entre topics et souscriptions. Les serveurs Kafka ne conservent au final que les publications.

5.2 Présentation de l'architecture de notre solution

Afin de mieux appréhender l'ensemble de l'architecture ainsi que les enjeux de sécurité de celle-ci, nous détaillons le fonctionnement et les caractéristiques de chaque entité ainsi que la

sécurité par défaut de l'architecture Apache Kafka. Ensuite, nous présentons l'ensemble des exigences de sécurité que nous fixons pour notre solution de sécurité. Enfin, nous définissons l'ensemble des menaces auxquelles doit faire face cette solution.

5.2.1 Entités

Les entités de notre architecture sont détaillées dans cette section. À noter que les autorités de confiance présentées ici permettent de distribuer les clés des cryptosystèmes que l'on utilise dans la Section 5.4.

— Les **Brokers Kafka** sont responsables du maintien et de la transmission des données entre publieurs et souscripteurs. Les brokers ont pour rôle de stocker les publications des publieurs pendant une certaine période et de répondre aux requêtes de lecture des souscripteurs. Ils sont dans un état passif où ils attendent les opérations de lecture/écriture des différents utilisateurs.

— Les **Serveurs Zookeeper** gèrent l'ensemble des métadonnées du système ainsi que les fichiers relatifs à la configuration des entités (c'est-à-dire topics, souscriptions, dernière publication consultée, etc.). Dans le cas d'un problème au niveau d'un broker Kafka, Zookeeper est responsable de la continuité du service en redirigeant les utilisateurs vers un broker disponible.

— Les **Publieurs** peuvent créer de nouveaux topics en envoyant une notification aux serveurs Zookeeper. Si le topic existe déjà, les publieurs peuvent demander aux serveurs Zookeeper sur quel broker ils peuvent publier (c'est-à-dire quelle est la partition leader de ce topic). Ils peuvent ensuite envoyer de nouvelles publications sur le broker Kafka.

— Les **Souscripteurs** peuvent souscrire à un ou plusieurs topics différents. Pour effectuer une souscription, le souscripteur va tout d'abord contacter un serveur Zookeeper pour savoir si le topic existe. S'il existe, il récupère la liste de brokers où il est possible de consulter ce topic. Lorsqu'un souscripteur consomme des données sur le broker, c'est à lui de maintenir quelle publication il a consommé en dernier. Cela signifie qu'à chaque fois qu'il se reconnecte, c'est à lui d'initier la connexion aux brokers Kafka. En cas de doute, il peut toujours contacter les serveurs Zookeeper.

— Les **Autorités de confiance** (TA) se partagent la distribution de clés aux utilisateurs. Les utilisateurs s'authentifient aux autorités afin de récupérer leur clé secrète.

Lorsque l'on applique notre scénario présenté en Section 2.4.2 à cette architecture, on obtient les correspondances suivantes : les publieurs sont les capteurs ainsi que les professionnels de santé effectuant des diagnostics et des examens situés dans les établissements de santé, au travers de leurs systèmes d'information. Les souscripteurs sont les multiples bases de données situées au niveau de l'hébergement des données de santé. Ces dernières souscrivent aux brokers Kafka afin d'organiser leurs données. Enfin, les brokers Kafka ainsi que les serveurs Zookeeper font le lien entre les établissements de santé et les bases de données de l'hébergement des données de santé. Les autorités de confiance sont l'ensemble des organismes publics responsables de ces établissements de santé, comme par exemple l'Agence du Numérique en Santé (ANS) [92] (qui est une agence gouvernementale française chargée de la santé numérique) ou l'établissement de santé où officie le professionnel.

Les composants de l'architecture ainsi que leurs rôles étant définis, nous définissons maintenant les mécanismes de sécurité existants déjà par défaut dans Apache Kafka ainsi que leurs limites.

5.2.2 Sécurité élémentaire de l'architecture

Par défaut, l'architecture Apache Kafka incorpore la possibilité de mettre en place un chiffrement point à point avec TLS entre les entités de l'architecture : clients, brokers Kafka et serveurs Zookeeper. Les communications qu'ils effectuent entre eux sont donc chiffrées mais les publications conservées par les brokers Kafka ne le sont pas, tout comme les informations dans les Znodes situées sur les serveurs Zookeeper.

Cela implique que malgré un mécanisme d'authentification existant entre les brokers Kafka et les serveurs Zookeeper, qui permet d'assurer que seuls les brokers Kafka sont autorisés à modifier certaines métadonnées stockées sur les serveurs Zookeeper, ces métadonnées sont accessibles publiquement par tous les utilisateurs.

De plus, Zookeeper est informé des centres d'intérêts de chaque souscripteur lorsqu'ils effectuent une souscription à travers ce dernier. Cette souscription s'effectue sous la forme d'une

requête de lecture sur un Znode pour connaître l'existence d'un topic, et ainsi récupérer l'adresse du broker où est située la partition leader (ou un ISR) de ce dernier.

L'architecture d'Apache Kafka ne propose donc pas par défaut de mécanisme de confidentialité et de protection de la vie privée au niveau des brokers ou des serveurs Zookeeper, et laisse ces derniers avoir accès à la totalité des données des utilisateurs. Ces données sont également exposées à l'extérieur. Notre objectif est de proposer une solution qui garantit la protection de la vie privée des utilisateurs d'Apache Kafka tout en conservant l'ensemble de ses fonctionnalités.

Maintenant que les rôles des entités sont établis ainsi que la sécurité de base du système a été présenté, nous définissons l'ensemble des exigences de sécurité de notre solution ainsi que les menaces auxquelles celle-ci est exposée.

5.2.3 Exigences de sécurité

Pour répondre au mieux aux menaces sur la sécurité de l'architecture, nous établissons les exigences de sécurité suivantes :

— E1 : Le contenu d'une publication doit être chiffré de manière à être masqué aux brokers ainsi qu'à toute partie non autorisée. Il doit être impossible pour un utilisateur non autorisé, ainsi qu'aux serveurs Kafka et aux brokers Zookeeper d'accéder aux informations protégées.

— E2 : Le topic associé à une publication peut contenir des informations sensibles, il doit également être masqué aux utilisateurs non autorisés ainsi qu'aux brokers Kafka et aux serveurs Zookeeper.

— E3 : Les serveurs Zookeeper doivent être capables de répondre aux souscripteurs sans connaître leurs intérêts, tout en ne divulguant pas d'informations supplémentaires à des souscripteurs non autorisés.

— E4 : La gestion des clés doit être répartie entre plusieurs autorités différentes de manière à gérer la compromission d'une ou plusieurs autorités.

— E5 : Dans une architecture distribuée, le système doit être capable de révoquer un utilisateur sans maintenir une liste de contrôle d'accès (ACL). Chaque souscripteur révoqué ne doit plus être capable de lire de nouvelles publications. Il doit être également possible de rendre indisponible les anciennes publications.

5.2.4 Modèle de menaces

Dans ce scénario d'un ensemble d'établissements de santé partageant des informations via une architecture Apache Kafka, avec les rôles précédemment établis pour les entités du système, nous considérons les menaces sur la sécurité suivantes :

— Les opérations de lecture et d'écriture sur le broker ne sont pas limitées/contrôlées (il ne rejette pas les communications).

— Des souscripteurs malhonnêtes peuvent collaborer de manière à accéder à des données auxquelles ils n'ont pas accès seuls.

— Les clés maîtresses d'une autorité peuvent être compromises.

— Un souscripteur révoqué essayera d'accéder aux brokers après avoir été révoqué du système.

En plus de ces menaces, il est important de prendre en compte le modèle de sécurité des brokers Kafka et des serveurs Zookeeper, qu'on considère *honest-but curious*. C'est-à-dire qu'ils suivent la procédure et ne corrompent pas les données, mais essayent de s'informer le plus possible sur le topic et le contenu des publications, ainsi que sur les souscriptions des souscripteurs.

Nous allons maintenant faire une présentation globale de notre solutions de sécurité, qui a pour objectif de répondre à nos exigences de sécurité tout en étant résistante à notre modèle de menaces.

5.2.5 Sécurité de notre solution

Nous présentons dans cette section notre solution de sécurité qui permet de répondre aux exigences de sécurité définies dans la section 5.2.3.

Dans notre première solution, présentée dans le chapitre 4, nous utilisons un schéma de CP-ABE pour garantir la confidentialité du contenu des publications (E1). Puisque nous considérons ici un scénario à plus grande échelle, nous avons choisi d'utiliser une itération de ce schéma qui décentralise les pouvoirs de l'autorité. En effet, puisque notre scénario implique un grand nombre d'acteurs, cela augmente aussi bien les risques que les conséquences de la compromission des clés maîtresses de l'autorité. Une solution avec des autorités décentralisées a l'avantage de permettre au système de résister à la compromission d'une ou plusieurs autorités, tout en distribuant les pouvoirs à différents organismes. Un exemple avec trois autorités est présenté en Figure 5.4.

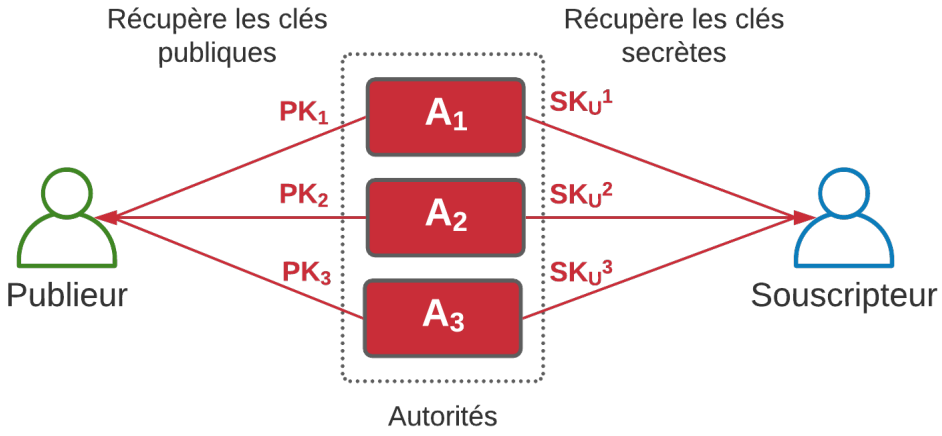


FIGURE 5.4 – CP-ABE avec plusieurs autorités

Nous utilisons donc, pour permettre la confidentialité du contenu des publications (E1), le schéma Multi-Authority Ciphertext-Policy Attribute-Based Encryption (MA-CP-ABE) de Qian et al. [108]. Il permet, tout comme un schéma CP-ABE, à un publieur de définir une politique d'accès pour chiffrer ses publications et ainsi contrôler quel groupe d'utilisateur peut les déchiffrer. Les souscripteurs s'authentifient au niveau des autorités de confiance afin de récupérer leur clé secrète MA-CP-ABE. Cette clé correspond aux attributs de l'utilisateur et permet à ce dernier de déchiffrer le contenu des publications.

Afin de rendre plus efficace le processus de chiffrement, nous utilisons le schéma MA-CP-ABE pour chiffrer une clé AES, qui permet de chiffrer le contenu des publications d'un même topic. Ce procédé nous permet de répondre également à la problématique de la confidentialité des topics (E2). Plus spécifiquement, un identifiant aléatoire est générée pour chaque topic, qui est chiffré avec MA-CP-ABE dans le Znode Zookeeper. Une clé AES est associée à cet identifiant aléatoire. Cet identifiant rend impossible pour les serveurs Zookeeper et pour les serveurs Kafka d'associer un topic avec son identifiant, tandis que les utilisateurs peuvent l'utiliser pour publier/souscrire. Lorsqu'un publieur envoie une publication à un broker Kafka, le contenu de cette publication est chiffré avec une clé AES tandis que le topic est l'identifiant aléatoire. Zookeeper conserve le Znode contenant ce chiffré à l'index du topic de la publication. L'identifiant aléatoire n'est pas connu de Zookeeper.

Au niveau de la confidentialité des souscriptions vis-à-vis des serveurs Zookeeper (E3), il n'était pas possible d'implémenter un schéma de searchable encryption comme appliqué à notre

solution précédente. En effet, le mécanisme de publication/souscription dans Apache Kafka est différent d'un modèle de publish/subscribe classique. Ici, les utilisateurs consultent les serveurs Zookeeper pour s'informer sur le statut des topics dans l'architecture. Zookeeper a donc un fonctionnement similaire à une base de données, et il faudrait changer son fonctionnement en profondeur pour y implémenter un schéma de searchable encryption.

À partir de ce constat, nous avons considéré deux options : le Private Information Retrieval (PIR) [34] et l'Oblivious Transfer (OT) [75]. Ces deux primitives permettent à un utilisateur d'effectuer une recherche dans la base de données d'un serveur sans que ce dernier n'ait connaissance, à aucun moment de l'échange, des intérêts de l'utilisateur. Leur principale différence se situe au niveau des données connues par l'utilisateur pendant l'échange. Avec un PIR, l'utilisateur peut prendre connaissance des données contenues dans la base de données, même si ce ne sont pas les données associées à sa requête. Avec un OT, l'utilisateur ne prend connaissance que de la donnée associée à sa requête. Un protocole de PIR naïf consiste donc à envoyer toute la base de données à l'utilisateur.

Dans notre cas d'utilisation, où nous cherchons à protéger la confidentialité des topics contenus dans Zookeeper, nous considérons chaque Znode fils du nœud parent "/topics" comme faisant parti d'une base de données à part entière. Notre objectif n'est pas de protégé le nom de tous les topics du système mais de protéger leur lien avec les publieurs et les souscripteurs. En d'autres termes, protéger les intérêts des utilisateurs vis-à-vis du serveur Zookeeper. Savoir qu'un topic « patients-cardiologie » existe dans un établissement de santé n'est pas une problématique de sécurité, mais savoir que cette donnée a été recherchée par un certain docteur ou un certain patient représente un problème de confidentialité. C'est pourquoi nous choisissons d'utiliser un PIR dans notre solution afin de protéger les intérêts des souscripteurs. Même si l'utilisation d'un PIR naïf (où l'on envoie la totalité de la base de données à l'utilisateur) est envisageable puisque nous traitons un nombre de topic assez réduit, nous avons préféré utiliser un cPIR, qui représente une amélioration au niveau de la quantité de données transmises en contrepartie de temps de calcul. Ce temps de calcul restant proportionnel à la taille de la base de données, il reste négligeable dans notre scénario. Les détails de ces protocoles sont explicités en Section 5.3.

Le schéma MA-CP-ABE permet également de réduire les risques relatifs à la compromission des clés maîtresses (E4), puisque les autorités de confiance se partagent la répartition des clés à

travers le système.

Pour finir, nous résolvons le problème de révocation d'utilisateur (E5) sans maintenir d'ACL en utilisant un chiffrement MA-CP-ABE qui intègre des mécanismes de révocation d'utilisateur et en mettant à jour les clés AES concernées. Ainsi, un utilisateur révoqué ne pourra pas accéder à de futures publications.

L'ensemble de ces solutions permettent également d'assurer la confidentialité des données sans compromettre la propriété de découplage de l'architecture publish/subscribe.

5.3 Private Information Retrieval (PIR)

Le Private Information Retrieval (PIR) permet d'effectuer une requête sur un serveur sans que celui-ci ne puisse déterminer les intérêts de l'utilisateur ayant effectué cette requête. La manière la plus triviale d'effectuer un PIR consiste à ce que l'utilisateur récupère la totalité de la base de données lors de sa requête et n'utilise que les données dont il a besoin. Évidemment, ce type de solution présente des problèmes de mise à l'échelle avec une base de données trop grande.

Le premier protocole PIR a été introduit par Chor et al. en 1995 [34]. Ces travaux se basent sur un système avec plusieurs serveurs où la base de données est répliquée. Depuis, d'autres schémas multi-serveurs plus efficaces ont été proposés, comme [38].

Les schémas PIR avec plusieurs serveurs ont de très bonnes performances, que ce soit au niveau des communications que des temps de calcul. Le principal inconvénient de ce type de PIR est qu'il ne reste sûr que tant que les serveurs ne collaborent pas pour récupérer des informations. On place ce type de protocole dans la catégorie Information-Theoretic PIR.

Il existe également des schémas qui utilisent un seul serveur. Ils sont dans la catégorie Computationally-PIR (cPIR). La sécurité de ces schémas repose sur des problèmes mathématiques difficiles, reposant sur la théorie des nombres [59] ou sur les réseaux euclidiens [3, 2]. Les schémas cPIR sont globalement moins efficaces au niveau du temps de calcul que les schémas Information-Theoretic PIR mais ont l'avantage de pouvoir s'appliquer lors de scénarios avec un seul serveur.

Choisir quel type de protocole PIR utiliser dans une architecture dépend donc du nombre

de serveurs utilisés ainsi que de la taille de la base de données. Dans tous les cas, les serveurs doivent au moins suivre le protocole correctement et donc être considérés comme semi-trusted (honest-but-curious).

Dans une architecture Apache Kafka, les serveurs Zookeeper communiquent de manière permanente pour rester synchronisés au niveau de leurs bases de données. Pour cette raison, nous considérons qu'il n'est pas réaliste d'appliquer un protocole d'Information-Theoretic PIR, qui irait à l'encontre de la nature du service Zookeeper. De plus, comme énoncé précédemment, utiliser un protocole cPIR dans un scénario avec une base de données de petite taille n'impacte que faiblement le temps de réponse des serveurs. La taille de notre base de données correspond bien à ce cas d'utilisation puisque nous appliquons le protocole cPIR lors des souscriptions, et donc uniquement sur le nom des topics du système.

Même si notre solution est adaptable à n'importe quel schéma de cPIR, nous choisissons de baser nos travaux avec le protocole présenté dans [2], qui est efficace sur des bases de données de petite taille.

Un protocole cPIR est composé des algorithmes suivants :

— $\mathsf{Setup}(\mathfrak{K}) \rightarrow params$: Génère les paramètres du système en fonction d'un paramètre de sécurité $\mathfrak{K}$.

— $\mathsf{Query}(i, params) \rightarrow Q$: Prend en entrée les paramètres du système ainsi qu'un index i pour générer une requête cPIR Q.

— $\mathsf{Reply}(db, Q) \rightarrow R$: À partir d'une base de données db et d'une requête cPIR Q, génère une réponse cPIR R pour l'utilisateur.

— $\mathsf{Extraction}(R) \rightarrow m$: Récupère le message m correspondant à l'index i ayant fait l'objet d'une requête à partir de la réponse R.

Dans la littérature, il n'y a à notre connaissance que peu de travaux sur des architectures de publish/subscribe utilisant un PIR dans le processus de souscription. Nous présentons tout de même le protocole Talek [33], qui présente de bons résultats au niveau de ses performances, même s'il ne se base pas sur une architecture de broker de stockage à grande échelle comme Apache Kafka.

5.3.1 Utilisation de PIR dans une architecture Publish/Subscribe

Cheng et al. [33] propose un schéma de publish/subscribe baptisé Talek, basé sur un schéma de PIR qui permet de rendre indistinguables les opérations d'écriture (publications) ainsi que les opérations de lecture (souscriptions). Talek incorpore également un mécanisme d'alertes aux souscripteurs qui leur permet de savoir qu'un nouveau message vient d'être publié, tout en masquant la liste de souscripteurs au serveur (c'est-à-dire private notifications). Le contenu des publications est chiffré avec un chiffrement CCA, et la confidentialité du topic est assuré avec le PIR.

Par rapport à nos exigences de sécurité, même si la solution de [33] répond à E1, E2 et E3, elle ne répond pas aux exigences E4 et E5. De plus, elle ne permet pas de conserver la propriété de découplage entre publieurs et souscripteurs, puisque le chiffrement des données implique que les publieurs connaissent les clés publiques des souscripteurs.

5.3.2 Solution de sécurité pour Zookeeper

Au niveau de la sécurité en elle même de l'architecture Apache Kafka, il n'y a pas, à notre connaissance, de solutions de sécurité spécifiquement appliquée à cette dernière. Des travaux ont cependant été réalisés pour sécuriser Apache Zookeeper [21] en tant qu'outil de coordination. Leur solution a pour objectif d'assurer la confidentialité et l'intégrité des données en utilisant les schémas AES [44] et Hash-Based Message Authentication Codes (HMAC) [77] dans une enclave Intel Software Guard Extensions (SGX) [84]. L'inconvénient de ce type de solution par rapport à notre cas d'utilisation, est que ces schémas gèrent difficilement plusieurs utilisateurs souhaitant accéder à la même donnée alors qu'ils ne partagent pas les mêmes clés. En effet, avec ce type de solutions les données doivent être chiffrées autant de fois qu'il y a de consommateurs de données différents.

5.4 Solution détaillée

Nous détaillons maintenant chaque étape de notre solution de sécurité qui se compose des phases suivantes : Mise en place du système, Nouvel utilisateur, Création d'un topic, Publication,

Souscription et Révocation d'utilisateur. L'ensemble de ces phases est récapitulé dans la Figure 5.5. Les algorithmes MA-CP-ABE utilisés sont présentés en Section 3.1.3. Les algorithmes du protocole cPIR sont eux présentés en Section 5.3.

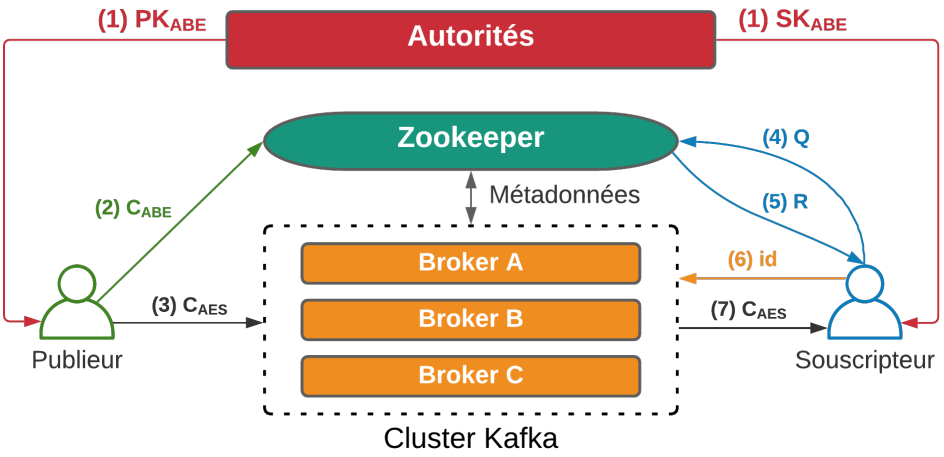


FIGURE 5.5 – Solution de sécurité pour l'architecture Apache Kafka

5.4.1 Mise en place du système

Lors de la mise en place du système, on considère un nombre k d'autorités. Après une génération des paramètres du système avec l'algorithme Global Setup, chaque autorité A_k lance l'algorithme Authority Setup pour générer ses clés maîtresses SK_k et MK_k. L'autorité conserve SK_k et publie PK_k (1).

De plus, une liste de tous les brokers du système est mise à disposition sous la forme de plusieurs Znodes dans les serveurs Zookeeper. Chaque serveur Zookeeper génère des paramètres de cPIR avec l'algorithme Setup. Ces paramètres seront proposés aux souscripteurs lors de leur connexion.

5.4.2 Nouvel utilisateur

Lorsqu'un souscripteur cherche à rejoindre le système, il effectue une requête à chaque autorité, qui après authentification, peut générer sa clé secrète partielle MA-CP-ABE $\{SK_U^k\}$ avec l'algorithme KeyGen. Après cette opération, sa clé secrète est associée à un ensemble d'attributs $\mathcal{A}$ et est notée SK_U (1). Lorsqu'un publieur souhaite rejoindre le système, il récupère les clés

maîtresses publiques PK_k fournies par les autorités de confiance. De plus, s'il souhaite publier dans des topics partagés avec d'autres publieurs, il effectue une requête à chaque autorité afin de générer une clé secrète MA-CP-ABE qu'on notera SK_{Pub}, qui contient l'ensemble des attributs qui le caractérise.

5.4.3 Création d'un topic

Lorsqu'un publieur crée un nouveau topic, il génère tout d'abord une clé AES k qui sera utilisée uniquement pour ce topic. Il effectue ensuite une requête au serveur Zookeeper pour avoir la liste de brokers Kafka qui sont en ligne, afin d'en sélectionner un. Puis, il génère un identifiant aléatoire id pour ce topic et choisit une politique P. Il chiffre ensuite, avec cette politique, la concaténation de la clé AES et de l'identifiant aléatoire avec l'algorithme MA-CP-ABE Encrypt.

Le chiffré généré est ensuite envoyé au serveur Zookeeper (2), afin de créer un Znode dans l'arbre "/topics". Cette intégration est illustrée en Figure 5.7.

Pour répondre au scénario où plusieurs publieurs veulent publier dans le même topic, nous intégrons dans le schéma MA-CP-ABE un certain nombre d'attributs qui ne sont délivrés qu'aux publieurs. Ainsi, si un publieur Pub_A veut partager son topic avec un publieur Pub_B, il doit également chiffrer cette clé AES avec une politique P'. Le publieur Pub_B possédant une clé secrète SK_{PUB_B} (1) correspondant à des attributs validant la politique P'. Cette politique contiendra des attributs qui n'intersectent pas avec les attributs délivrés aux souscripteurs. Ce nouveau chiffré est ensuite envoyé dans l'arbre "/publieurs/topics/" (2), et pourra être récupéré par Pub_B (2b). La suite de la communication reste la même. Ce processus de partage de topic est illustré en Figure 5.6.

Nous distinguons donc dans Zookeeper un arbre de Znodes consacrés aux souscriptions et un autre aux publieurs souhaitant partager leurs topics avec d'autres publieurs.

5.4.4 Publication

Pour publier un message sur le broker, le publieur utilise donc la clé AES précédemment générée pour chiffrer son message. Le topic utilisé est l'identifiant aléatoire correspondant. Ainsi, la publication (topic=random_id,payload=AES(data)) est envoyée au broker (3). À noter que le

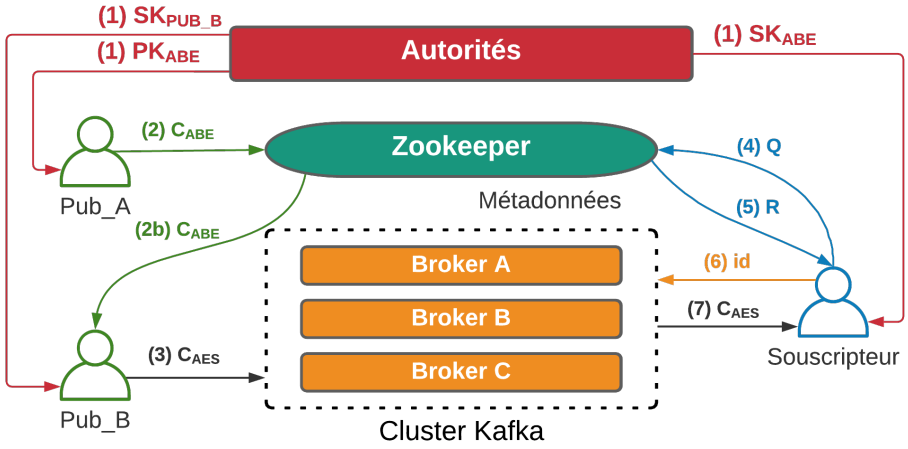


FIGURE 5.6 – Solution de sécurité pour l'architecture Apache Kafka avec plusieurs publieurs pour un même topic

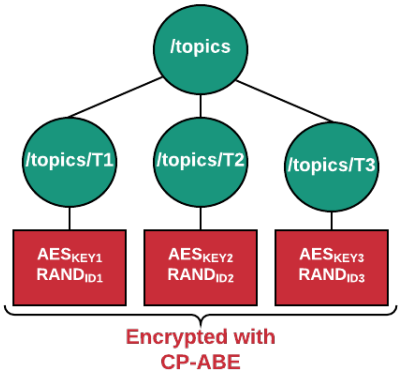


FIGURE 5.7 – Znodes Zookeeper

publieur contacte le broker leader, cette adresse étant contenue en clair dans le Znode du topic.

5.4.5 Souscription

Lorsqu'un souscripteur souscrit à un topic, il utilise le protocole cPIR sur le serveur Zookeeper qu'il requête. Le souscripteur utilise l'algorithme Query pour générer sa requête Q relative au topic auquel il souhaite s'abonner. Il envoit ensuite cette requête au serveur Zookeeper (4). Ce dernier, en recevant cette requête, génère une réponse R avec l'algorithme Reply et la transmet au souscripteur (5). Il récupère le Znode associé avec l'algorithme Extraction.

Pour finir, le souscripteur déchiffre le Znode extrait afin de récupérer l'identifiant aléatoire ainsi que la clé secrète AES du topic. Il peut ainsi contacter un broker Kafka afin de consulter les données de ce topic via l'identifiant aléatoire id (6). Les données récupérées (7) peuvent ensuite

être déchiffrées avec la clé AES k.

5.4.6 Révocation d'utilisateur

Pour révoquer totalement un utilisateur du système, les autorités de confiance ciblent dans un premier temps quels sont les attributs impactés par cette révocation. Puis, chaque autorité gérant ces attributs utilise l'algorithme MA-CP-ABE ReKeyGen afin de re-générer ses clés. Chaque autorité peut ensuite re-générer les clés des utilisateurs légitimes avec l'algorithme KeyUpdate. Ensuite, les clés AES et les identifiants aléatoires sont régénérés. Enfin, ces valeurs sont re-chiffrées avec MA-CP-ABE. L'ensemble de cette procédure a pour but d'empêcher un utilisateur révoqué de déchiffrer de nouvelles publications.

Dans le cas où un publieur souhaite révoquer un attribut de sa publication ou mettre à jour sa politique ABE, il lui suffit de générer une nouvelle clé AES k' et de chiffrer cette clé ainsi que l'identifiant aléatoire avec ABE. Il effectue ensuite une mise à jour du Znode sur Zookeeper en remplaçant son contenu avec ce chiffré. Pour finir, les nouvelles publications seront chiffrées avec la clé AES k'.

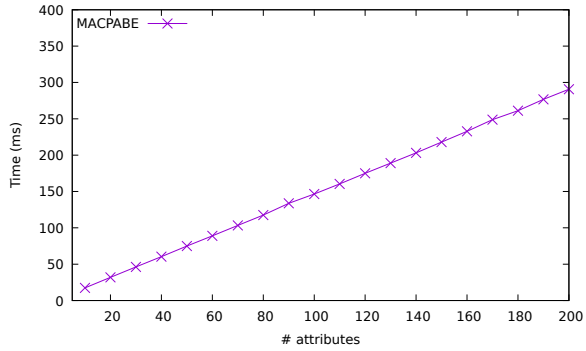
Nous présentons maintenant les résultats expérimentaux de notre solution, dans le but de démontrer la praticabilité de celle-ci dans un scénario réel.

5.5 Implémentation

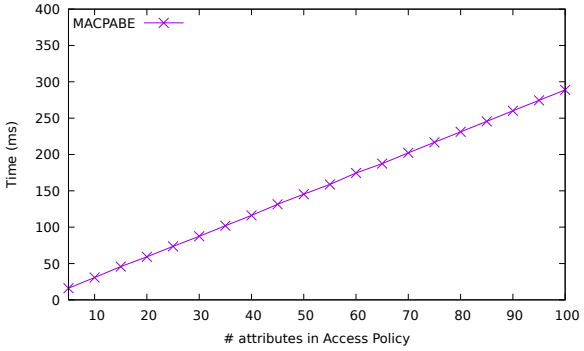
Nous présentons dans cette section un certains nombre d'expérimentations que nous avons effectuées pour évaluer l'efficacité de notre solution.

À notre connaissance, le schéma MA-CP-ABE de Qian et al. [108] n'a pas encore été implémenté. Afin de réaliser ces expérimentations, nous avons utilisé la bibliothèque Charm-Crypto Python [4].

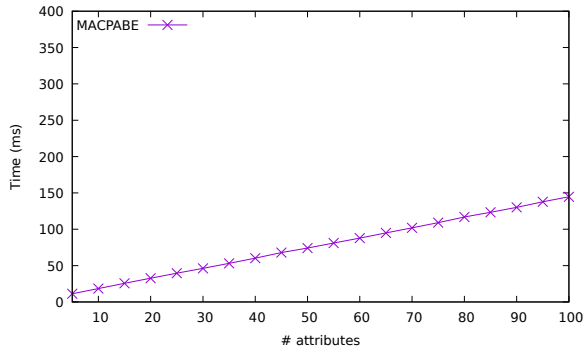
Nos expérimentations ont été réalisées sur une machine virtuelle avec 4 Go de RAM et un processeur Intel Core i5 de 2.3 GHz. Ces mesures ont été faites en chiffrant la concaténation d'une clé AES et d'un identifiant, représentant un message de 512 bits. Les paramètres variables de ces expérimentations sont :



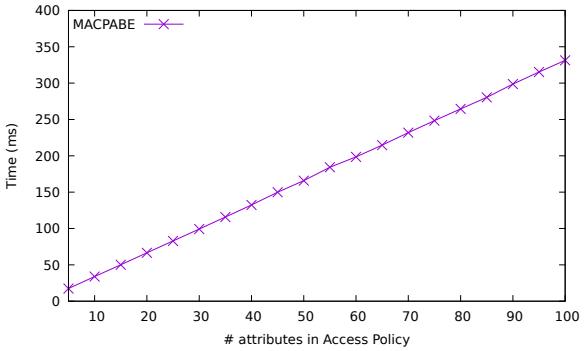
(a) Génération des clés maîtresses



(c) Chiffrement



(b) Génération de clé secrète



(d) Déchiffrement

FIGURE 5.8 – Évaluation des performances du schéma Multi-Authority CP-ABE

— Le nombre d'attributs dans l'univers

— Le nombre d'attributs dans la politique pour chiffrer

— Le nombre d'attributs attachés à la clé secrète

Pour les algorithmes MA-CP-ABE KeyGen, Encrypt et Decrypt, le nombre d'attributs contenus dans la politique ou attachés à la clé secrète est noté n. Dans ces scénarios, le nombre d'attributs dans l'univers est de $2n$. Chaque mesure est effectuée sur une moyenne de 100 exécutions.

Dans la Figure 5.8a, le temps mis par l'algorithme Authority Setup, exécuté par chaque autorité du système, est présenté. Cette exécution prend jusqu'à 290 ms pour 200 attributs. Cette opération n'est effectuée qu'une seule fois par autorité.

Les temps de l'algorithme KeyGen, exécuté par les autorités une seule fois par utilisateur, est illustré dans la Figure 5.8b. Cette exécution prend jusqu'à 144 ms pour 100 attributs.

L'algorithme Encrypt, illustré dans la Figure 5.8c, prend jusqu'à 288 ms pour 100 attributs dans la politique attachée au chiffré.

Enfin, comme illustré dans la Figure 5.8d, l'algorithme Decrypt prend jusqu'à 331 ms pour 100 attributs dans la politique attachée au chiffré. Il peut être noté que la plupart du temps, il n'est pas nécessaire lors du déchiffrement de parcourir chaque feuille de l'arbre de la politique. Dans la majorité des cas, ce temps d'exécution sera donc significativement plus court.

Pour les algorithmes Encrypt et Decrypt, nous prenons l'exemple de scénarios utilisant un grand nombre d'attributs. Cependant, on peut raisonnablement considérer que la plupart des politiques, dans un scénario réel, ne comporteront que quelques dizaines d'attributs, correspondant aux différents corps de métiers présents dans l'établissement de santé. Avec ces paramètres, le chiffrement et le déchiffrement ne prendront respectivement que 30 ms et 33 ms.

Puisque nous utilisons MA-CP-ABE pour déchiffrer une clé AES, le surcoût apporté par la confidentialité des données est négligeable, puisqu'il suffit aux utilisateurs de n'utiliser les algorithmes Encrypt et Decrypt qu'un nombre limité de fois. En effet, le publieur ne chiffre cette clé qu'à la création du topic ou lors de la mise à jour du topic. Il en va de même pour le souscripteur qui déchiffre cette clé que lorsqu'il souscrit pour la première fois ou lors de la mise à jour.

Au niveau des serveurs Zookeeper, le coût du processus de réponse aux souscriptions est nul lorsque l'on utilise un PIR naïf, mais consiste à envoyer l'ensemble des Znodes contenus dans le l'arbre "/topics". Lorsque l'on utilise le protocole XPIR [2], une implémentation du protocole cPIR, nous obtenons un temps de réponse du serveur Zookeeper de moins de 100 ms lorsque la base de données est de 100 Mo. Ce temps de réponse passe à moins de 10 ms pour une base de données de 10 Mo. Lorsque l'on applique notre solution à notre scénario présenté en Section 2.4.2, on considère que la taille de la base de données se situe entre 10 Mo pour les GHT de petite taille et de 100 Mo pour les GHT les plus grands.

L'ensemble de ces résultats démontrent que le surcoût de notre solution de sécurité au niveau du temps de traitement pour l'ensemble des utilisateurs est acceptable et ne compromet pas l'utilisation pratique de l'architecture Apache Kafka dans notre scénario.

5.6 Conclusion du chapitre

Dans ce chapitre, nous avons présenté une solution de sécurité pour l'architecture Apache Kafka, de manière à garantir une protection de la vie privée des utilisateurs ainsi la confidentialité de leurs données.

Pour cela, nous avons pris en compte les différentes spécificités de l'architecture Apache Kafka. Nous avons défini chacune des différentes entités ainsi que leurs rôles. Après cela, nous avons appliqué le scénario décrit en Section 2.4.2 à cette architecture afin d'associer chaque entité avec un acteur du scénario.

Une fois l'architecture et le scénario bien définies, nous avons établi un modèle de sécurité pour cette architecture. Nous avons commencé par nous baser sur la sécurité déjà présente dans Apache Kafka, puis nous avons établis les différentes exigences de sécurité de notre solution ainsi qu'un modèle de menaces.

En se basant sur l'ensemble des prédicats définis précédemment, notre solution de sécurité permet aux utilisateurs d'utiliser Apache Kafka sans que les brokers Kafka ou les serveurs Zookeeper n'aient connaissance des données transmises. Elle permet aux publieurs de chiffrer leurs données avec ABE et AES, et aux souscripteurs de ne pas divulguer leurs souscriptions en utilisant un PIR au niveau du serveur Zookeeper. Les brokers Kafka n'ont également pas connaissance du nom des topics associés aux publications puisque ces noms sont générés aléatoirement. Pour finir, il est possible de révoquer des utilisateurs lorsque ces derniers n'ont plus les droits d'accès aux données.

Enfin, nous présentons un ensemble de résultat expérimentaux de cette solution, qui démontrent la praticabilité de ces mesures de sécurité dans un scénario réaliste.

Chapitre 6

Attribute-Based Designated Verifier Signature

Un schéma de signature à vérifieur désigné permet au vérifieur de contrôler qui peut vérifier sa signature. En effet, à l'inverse d'une signature classique où tous les utilisateurs peuvent vérifier une signature avec une clé de vérification publique, le vérifieur doit utiliser sa clé secrète pour effectuer la vérification.

Cependant, pour certaines applications, il peut être utile de désigner un groupe de vérifieurs plutôt qu'un seul. Pour cela, on peut utiliser des signatures basées sur les attributs. Là où des schémas Attribute-based Signature (ABS) vont permettre à un utilisateur de signer avec ses attributs, un schéma Attribute-Based Designated Verifier Signature (ABDVS) va permettre à un utilisateur de signer de manière classique tout en imposant au vérifieur désigné de s'authentifier selon certains attributs. C'est-à-dire que pour vérifier une signature en ABDVS, le vérifieur désigné doit posséder certains attributs qui valident la politique choisie préalablement par le signataire.

Afin de répondre à cette problématique, Fan et al. [51] ont donc proposé une signature à vérifieur désigné basé sur les attributs. Dans leur schéma, un utilisateur peut ainsi signer un message, et n'autoriser que les utilisateurs possédant certains attributs à procéder à la vérification.

Sommaire

La primitive ABDVS peut être utilisée dans des scénarios où l'on cherche à protéger l'identité du signataire, tout en permettant à plusieurs utilisateurs de vérifier la validité de la signature.

La primitive ABDVS étant très intéressante pour garantir une protection de la vie privée des utilisateurs dans un contexte de santé, nous proposons un schéma ABDVS qui a pour objectif d'obtenir de meilleurs temps de calculs que le schéma de Fan et al. [51] en utilisant des primitives cryptographiques différentes. Cette construction a été travaillée en collaboration avec Olivier Blazy et Laura Brouilhet.

6.1 Définitions

Nous définissons dans cette section les propriétés mathématiques que nous allons utiliser pour la construction de notre ABDVS.

6.1.1 Groupe bilinéaire

Considérons un algorithme probabiliste en temps polynomial ggen qui, prenant en entrée $\mathfrak{K}$ renvoie un groupe bilinéaire $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ avec $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ des groupes cycliques

d'ordre p et g_1 et g_2 les générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$, respectivement, et où $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ est une forme bilinéaire non-dégénérée. On définit également $g_T := e(g_1, g_2)$, qui est un générateur de $\mathbb{G}_T$.

6.1.2 Notation matricielle et hypothèses

Pour une matrice $\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times n}$, $\overline{\mathbf{A}} \in \mathbb{Z}_p^{k\times n}$ correspond à la partie supérieure de la matrice $\mathbf{A}$ et $\underline{\mathbf{A}} \in \mathbb{Z}_p^{1\times n}$ la dernière colonne de la $\mathbf{A}$.

Nous utilisons la représentation implicite des éléments de groupes introduits dans [46]. Pour $s \in \{1, 2, T\}$ et $a \in \mathbb{Z}_p$ on définit $[a]_s = g_s^a \in \mathbb{G}_s$ comme la représentation implicite de a dans le groupe $\mathbb{G}_s$ (on utilise $[a] = g^a \in \mathbb{G}$ si l'on considère un groupe unique).

De manière plus générale, avec $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n\times m}$ on définit $[\mathbf{A}]_s$ comme la représentation implicite de la matrice $\mathbf{A}$ dans $\mathbb{G}_s$:

$$[\mathbf{A}]_s := \begin{pmatrix} g_s^{a_{11}} & \dots & g_s^{a_{1m}} \\ & & \\ g_s^{a_{n1}} & \dots & g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n\times m}$$

On utilise la notation implicite des éléments dans $\mathbb{G}_s$, c'est-à-dire $[a]_s \in \mathbb{G}_s$ est la représentation de a dans le groupe $\mathbb{G}_s$.

À noter qu'avec $[a]_s \in \mathbb{G}_s$ il est la plupart du temps difficile de calculer la valeur de a (problème du logarithme discret dans $\mathbb{G}_s$). De la même manière, il est difficile avec $[b]_T \in \mathbb{G}_T$ de calculer les valeurs $[b]_1 \in \mathbb{G}_1$ et $[b]_2 \in \mathbb{G}_2$ (problème d'inversion de couplage).

Bien sûr, avec $[a]_s \in \mathbb{G}_s$ et un scalaire $x \in \mathbb{Z}_p$, il est possible de calculer efficacement $[ax]_s \in \mathbb{G}_s$. De la même manière, avec $[a]_1, [b]_2$ il est possible de calculer efficacement $[ab]_T$ en utilisant l'opération de couplage e. Pour $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^k$ on définit $e([\mathbf{a}]_1, [\mathbf{b}]_2) := [\mathbf{a}^\top \mathbf{b}]_T \in \mathbb{G}_T$.

Après avoir défini les différentes opérations utilisées dans notre construction, nous rappelons la définition de l'hypothèse de la matrice Diffie-Hellman (MDDH) [46].

Definition 1 (Distribution Matricielle [46])**.** *Soit $k \in \mathbb{N}$. $\mathcal{D}_k$ est une distribution matricielle si elle renvoie une matrice de rang maximal k dans $\mathbb{Z}_p^{(k+1)\times k}$ en temps polynomial.*

On suppose que les k premières lignes de $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ forment une matrice inversible. Le problème

$\mathcal{D}_k$-Matrix Diffie-Hellman consiste à distinguer les distributions $([\mathbf{A}], [\mathbf{Aw}])$ et $([\mathbf{A}], [\mathbf{u}])$ où $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$ et $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Definition 2 (Hypothèse $\mathcal{D}_k$-Matrix Diffie-Hellman $\mathcal{D}_k$-MDDH)**.** *Soit $\mathcal{D}_k$ une distribution matricielle et $s \in \{1, 2, T\}$. On dit que l'hypothèse $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) est dans le groupe $\mathbb{G}_s$ si pour tout adversaire $\mathcal{D}$ on a :*

$$\mathbf{Adv}^{MDDH}_{\mathcal{D}_k,\mathsf{ggen}}(\mathcal{D}) \stackrel{\text{def}}{=} |\Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{Aw}]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{u}]_s) = 1]| = \mathsf{negl}(\lambda)$$

où $\mathcal{G} \xleftarrow{\$} \mathsf{ggen}(\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k, \mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Dans notre construction nous utilisons la définition précédente avec $k = 1$. Dans ce cas spécifique, on revient à l'hypothèse Symmetric External Diffie-Hellman (SXDH).

Definition 3 (Symmetric External Diffie-Hellman (SXDH[7]))**.** *L'hypothèse SXDH tient dans $\mathbb{G}_1$ et $\mathbb{G}_2$ si DDH est difficile à calculer dans $\mathbb{G}_1$ et $\mathbb{G}_2$.*

Nous présentons maintenant un scénario applicatif de notre ABDVS, dans un environnement de santé.

6.2 Application de notre ABDVS

Pour illustrer les avantages de l'utilisation d'une signature ABDVS plutôt qu'une signature à vérifieur désigné classique, ou même qu'une signature électronique, nous nous basons sur le scénario AAL défini dans la section 2.4.1.

Dans ce cas d'utilisation illustré en Figure 6.1, Alice est une patiente qui porte un capteur cardiaque. Les données relevées par ce capteur sont envoyées vers un serveur distant. On peut considérer que ces données sont chiffrées avec un schéma de chiffrement quelconque. Ce serveur est accessible par plusieurs entités, dont différents professionnels de santé.

Dans notre exemple, nous avons un médecin de Limoges, qui prend en charge Alice et qui a pour rôle de récupérer ces données afin de les analyser et pouvoir effectuer des diagnostics.

Lorsque ces données sont envoyées vers le serveur distant, il est nécessaire qu'elles n'aient pas été modifiées ou altérées, puisque ce sont des données de santé sensibles. Il peut également être nécessaire que ses données soient authentifiées, pour être certain que c'est bien Alice qui les a envoyées. Un mauvais traitement des données pourra avoir des conséquences directes sur la santé du patient.

Or, d'un autre côté, Alice ne veut pas que ses données soient attachées à son identité publiquement, c'est-à-dire qu'elle ne veut pas que n'importe quel utilisateur se connectant sur ce serveur puisse s'informer de ses potentiels soucis cardiaques. Il est important de noter que ces données doivent pouvoir être consultées aussi bien par son médecin traitant que par un cardiologue si l'avis d'un spécialiste est nécessaire.

Dans ce cas d'utilisation, un ABDVS permet à la fois d'assurer l'authenticité et l'intégrité du message, tout en protégeant l'identité du signataire. Il permet également de désigner des vérifieurs désignés différents. Dans notre exemple, les données de santé captées par le capteur cardiaque attaché à Alice puis envoyées sur le serveur sont protégées par la politique ABDVS "(MEDECIN ET LIMOGES) OU CARDIOLOGUE". De cette manière, seuls les professionnels validant ses attributs avec leur clé ABDVS peuvent avoir la certitude que ces données proviennent bien de ce patient. Ici, Bob possède une clé avec les attributs "[MEDECIN, LIMOGES]" et peut ainsi vérifier la validité de la signature.

Nous présentons maintenant la construction de notre ABDVS, ainsi que les différents schémas utilisés dans sa construction.

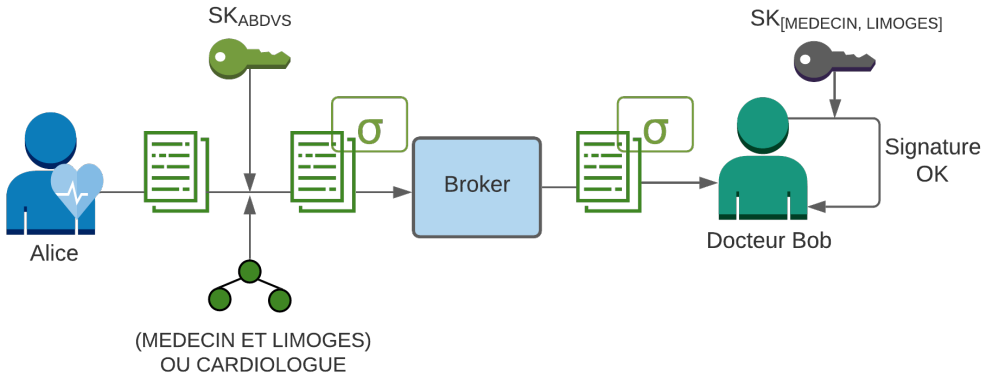


FIGURE 6.1 – Application de la signature ABDVS dans un scénario de santé

6.3 Construction

Notre proposition d'ABDVS se base sur plusieurs schémas cryptographiques que nous avons modifiés. Nous présentons ces schémas dans cette section.

L'objectif de notre ABDVS est de n'autoriser la vérification de la signature que par les vérifieurs désignés possédant un ensemble d'attributs spécifiques. Si les attributs du vérifieur désigné ne vérifient pas la politique de la signature, la vérification échoue.

Dans notre construction, l'ensemble d'attributs qui caractérisent une signature ou un individu sont définis de la manière suivante :

Pour tout sous-ensemble d'attributs $S \subseteq \mathbb{U}$, avec $\mathbb{U}$ l'univers d'attributs, on définit $id_S \in \{0,1\}^l$ où le i-ème bit est défini par :

$$id_s[i] := \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon} \end{cases} \tag{6.1}$$

En d'autres termes, n'importe quel ensemble d'attributs dans l'univers peut être traduit par une chaîne de bits de longueur l, avec l le nombre d'attributs dans l'univers. Si l'utilisateur possède l'attribut i, le i-ème bit sera positionné à 1, sinon il sera positionné à 0.

Prenons l'exemple de l'ensemble d'attributs [MEDECIN,LIMOGES,CHU] d'un utilisateur dans l'univers [MEDECIN,AIDE-SOIGNANT,LIMOGES,PARIS,CLINIQUE,CHU]. Cet ensemble d'attributs pourra donc aussi se noter sous la forme de l'identité [1,0,1,0,0,1].

Le premier bloc de construction de notre schéma est le Downgradable IBE, qui nous permet de masquer la signature.

6.3.1 Downgradable IBE

Le schéma de Downgradable IBE (DIBE) est introduit par [17]. C'est un schéma de chiffrement assez proche d'un IBE, qui permet d'effectuer une transformation d'un IBE vers un ABE. Le principe est de traduire une politique par un ensemble d'identités. En pratique, cette transformation s'opère vers un KP-ABE, de manière à ce que la politique associée à une clé secrète soit divisée en plusieurs sous-clés qui correspondent aux identités qui valident cette politique. Une poli-

tique "MEDECIN ET LIMOGES ET CHU" se traduirait donc, dans l'univers [MEDECIN,AIDE-SOIGNANT,LIMOGES,PARIS,CLINIQUE,CHU], par les identités : [1,0,1,0,0,1], [1,1,1,0,0,1], [1,1,1,1,0,1], [1,1,1,1,1,1], [1,0,1,1,0,1], [1,0,1,1,1,1] et [1,0,1,0,1,1].

Ainsi, lors du déchiffrement, l'utilisateur peut "downgrader" une de ses clés afin de la faire correspondre à la même identité que l'identité qui a permis de chiffrer le message. Si un message est chiffré avec les attributs [MEDECIN,LIMOGES] et donc avec l'identité [1,0,1,0,0,0], l'utilisateur choisira sa clé associée à l'identité la plus proche, soit : [1,0,1,0,0,1]. Il pourra ensuite la "downgrader" en flippant le 6-éme bit de 1 vers 0. Avec cette nouvelle clé, il pourra déchiffrer le message. À noter que s'il est possible de flipper un bit de 1 vers 0, il n'est pas possible de faire l'inverse, puisque dans les faits cette opération augmenterait les privilèges de l'utilisateur.

Un schéma DIBE est défini par les algorithmes $(\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Encap}, \mathsf{Decap}, \mathsf{USKDown})$:

— $\mathsf{Setup}(\mathfrak{K}) \rightarrow \mathsf{mpk}, \mathsf{msk}$: Génère la clé maîtresse publique mpk et la clé maîtresse secrète msk.

— $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id}) \rightarrow \mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}]$: Génère la clé secrète d'un utilisateur associé à son identité id $\mathsf{usk}[\mathsf{id}]$ ainsi qu'une clé de délégation $\mathsf{udk}[\mathsf{id}]$.

— $\mathsf{Encap}(\mathsf{mpk}, \mathsf{id}) \rightarrow c, \mathsf{K}$: Génère un chiffré c ainsi qu'une clé symétrique K.

— $\mathsf{USKDown}(\mathsf{usk}[\mathsf{id}], \tilde{\mathsf{id}}) \rightarrow \mathsf{mpk}, \mathsf{msk}$: Dérive une clé secrète $\mathsf{usk}[\tilde{\mathsf{id}}]$ tant que $\tilde{\mathsf{id}} \preceq \mathsf{id}$.

— $\mathsf{Decap}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c) \rightarrow \mathsf{mpk}, \mathsf{msk}$: Génère la clé de décapsulation K ou $\perp$.

Couplé au schéma DIBE, on utilise la transformation de Naor afin de signer notre message.

6.3.2 Transformation de Naor

La transformation de Naor permet d'obtenir un protocole de signature à partir d'un chiffrement basé sur l'identité. Cette transformation est formalisée par Cui et al. dans [36].

On rappelle qu'un IBE est composé de quatre algorithmes : $\mathsf{Setup}_{\mathsf{IBE}}, \mathsf{KeyGen}, \mathsf{Encrypt}$ et $\mathsf{Decrypt}$.

La sécurité d'une signature obtenue à l'aide de cette transformation c'est-à-dire la résistance aux contrefaçons est assurée par la sécurité sémantique du protocole IBE utilisé.

Nous utilisons le principe de cette transformation sur le schéma DIBE afin d'obtenir notre signature ABDVS. La formalisation de la transformation de Naor est présentée en Figure 6.2.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$	$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{vk}, m) :}$	$\underline{\mathsf{Verify}(\mathsf{vk}, \sigma, m) :}$
$(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathsf{IBE}}$	$\mathsf{id} = m$	$\mathsf{id} = m$
$\mathsf{vk} = \mathsf{param}$	$\mathsf{usk}[\mathsf{id}] \leftarrow \mathsf{KeyGen}(\mathsf{vk}, \mathsf{sk}, \mathsf{id})$	$\mathsf{usk}'[\mathsf{id}] \leftarrow \sigma$
$\mathsf{sk} = \mathsf{msk}$	$\sigma \leftarrow \mathsf{usk}[\mathsf{id}]$	$m_1 \xleftarrow{\$} \mathcal{M}$
Return $(\mathsf{vk}, \mathsf{sk})$	Return (σ, m)	$c \leftarrow \mathsf{Encrypt}(\mathsf{vk}, \mathsf{id}, m_1)$
		$m' \leftarrow \mathsf{Decrypt}(\mathsf{usk}'[\mathsf{id}], c, \mathsf{vk})$
		Return 1 si $m' = m_1$ et 0 sinon

FIGURE 6.2 – Formalisation de la transformation de Naor

6.3.3 Attribute-Based Designated Verifier Signature

Notre schéma Attribute-Based Designated Verifier Signature (ABDVS) utilise les schémas présentés précédemment afin de permettre la vérification d'une signature avec un ensemble d'attributs. La Figure 6.3 présente l'utilisation de ces schémas dans notre construction.

— Avec le schéma de DIBE, nous générons une clé pour $\mathsf{id}_1||m$ en dérivant la clé reçue par l'utilisateur id_1 en utilisant la transformation de Naor. C'est l'équivalent de la signature d'identifiant m.

— Le schéma DIBE permet la génération des clés du vérifieur désigné ainsi que la dérivation de celles ci. Il est également utilisé pour masquer la signature.

— Un algorithme de dérivation de clé permet au vérifieur désigné d'obtenir une clé uusk, à partir de sa clé usk et ses attributs.

La sécurité de notre ABDVS repose sur la sécurité des schémas utilisés. Au niveau des exigences de sécurité, un schéma ABDVS doit être résistant aux contrefaçons, non-transférable et respecter la *perfect privacy*. Ces propriétés de sécurité sont détaillées en Section 6.5.

6.4 Schéma

Pour cette construction ABDVS, de la même manière que dans [18], nous utilisons un univers d'attributs restreint noté $\mathcal{U} = \{1, \ldots, \ell\}$, avec ℓ la longueur des identités utilisées dans notre construction.

Pour l'expression de la politique, on utilise des formules booléennes de forme normale dis-

$\underline{\mathsf{Setup}(\mathfrak{K}) :}$
$\mathsf{mpk}, \mathsf{msk} = \mathsf{DIBE.Setup}(\mathfrak{K})$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}_{\mathsf{Sign}}(\mathsf{msk}, \mathsf{id}_1) :}$
$\mathsf{sk}, \mathsf{ek} \leftarrow \mathsf{DIBE.USKGen}(\mathsf{msk})$
Return sk, ek

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id}_2) :}$
$\mathsf{usk}, \mathsf{dk} \leftarrow \mathsf{DIBE.USKGen}(\mathsf{msk})$
Return $\mathsf{usk}, \mathsf{dk}$

$\underline{\mathsf{Sign}(\mathsf{mpk}, (\mathsf{usk}, \mathsf{ek}, \mathsf{id}_1), m, \mathbb{F}) :}$
$\sigma_1, \sigma_2 \leftarrow \mathsf{DIBE.USKGen}(\mathsf{usk}, \mathsf{id}_1 || m)$
Parse $\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{i, \mathsf{id}_{j,i}=1} 1)$
For all $j \in [1, k]$, compute :
$(c_j, K_j) \leftarrow \mathsf{DIBE.Encap}(\mathsf{mpk}, \mathsf{id}_j, \sigma_2)$
Return $C = (c_1, ..., c_k)$, $\mathsf{K} = (\mathsf{K}_1, ..., \mathsf{K}_j)$ and σ_1

$\underline{\mathsf{Verify}(\mathsf{sk}_2, C) :}$
Parse $\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{a \in I} a)$
Find $j \in [1, k]$ s.t. $S_j \subseteq \mathbb{A}$
$\mathsf{uusk} \leftarrow \mathsf{USKUp}(\mathsf{usk}, \mathsf{id}_{2,S_j})$
$K_j \leftarrow \mathsf{DIBE.Decap}(\mathsf{uusk}, \mathsf{id}_{2,S_j}, C_j)$

FIGURE 6.3 – Vue d'ensemble du schéma ABDVS

jonctive - Disjunctive Normal Form (DNF), c'est-à-dire un ensemble de disjonctions de clauses conjonctives. On note k le nombre de disjonctions dans la formule DNF. Par exemple, la politique "(MEDECIN ET LIMOGES) OU (AIDE-SOIGNANT ET CHU)" est de taille $k = 2$.

La construction de notre ABDVS est présentée en Figure 6.4.

Afin de pouvoir effectuer une vérification, l'utilisateur légitime doit effectuer un certain nombre d'opérations.

Tout d'abord, chaque potentiel vérifieur désigné demande à l'autorité de confiance (TA) de générer un ensemble de k clés secrètes qui correspondent à sa politique où k est le nombre de disjonctions de cette politique.

Lorsqu'un utilisateur cherche à vérifier une signature, il sélectionne une de ces clés qui peut être dérivée de manière à correspondre à la clé de vérification attendue. Cette clé dérivée par l'algorithme USKUp est notée uusk.

Le vérifieur désigné peut donc contrôler la validité de la signature seulement si sa politique correspond à l'ensemble d'attributs attachés à la signature. Si sa politique ne correspond pas, aucune de ses clés dérivées ne pourra effectuer la vérification, et il ne récupérera aucune information sur la validité de cette signature.

On note id_1 l'identité qui contient les attributs nécessaires à la vérification. id_2 est l'identité du vérifieur désigné, c'est-à-dire que id_2 contient tous les attributs du vérifieur désigné.

Dans notre schéma, on utilise deux algorithmes KeyGen différents :

$\underline{\mathsf{Setup}(\mathfrak{K}) :}$
For $i \in \{1, 2\ell\} : z_i \xleftarrow{\$} \mathbb{Z}_p$;
$a, z' \xleftarrow{\$} \mathbb{Z}_p; y' = az'$;
$\mathsf{mpk} = (([z_i]_{1,2})_{1 \le i \le 2\ell}, [a]_2, [z']_{1,2})$
$\mathsf{msk} = ((z_i)_{1 \le i \le 2\ell}), y')$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}_{\mathsf{Sign}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id}_1, \perp) :}$
$s \xleftarrow{\$} \mathbb{Z}_p$
$v = y' + (\sum_1^\ell \mathsf{id}_{1,i} z_i)s$
For $i \in [\ell + 1, 2\ell]$:
$e_i = z_i \cdot s \in \mathbb{Z}_p$;
$\mathsf{sk} = ([s]_1, [s]_2, [v]_1)$
Return $(\mathsf{sk}, \mathsf{ek} = [e_i]_1)$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id}_2, \perp) :}$
$s \xleftarrow{\$} \mathbb{Z}_p$
$v = y' + (\sum_1^\ell \mathsf{id}_{2,i} z_i)s$
For $i, \mathsf{id}[i] = 1$:
$d_i = z_i \cdot s \in \mathbb{Z}_p$;
$\mathsf{usk} = ([s]_1, [s]_2, [v]_1)$
Return $(\mathsf{usk}, \mathsf{dk} = [d_i]_1)$

$\underline{\mathsf{Sign}(\mathsf{mpk}, (\mathsf{usk}, \mathsf{ek}, \mathsf{id}_1), m, \mathbb{F}) :}$
$u \xleftarrow{\$} \mathbb{Z}_p$, $s' = s + u$
$v' = v + (\sum_{i=\ell+1}^{2\ell} m_{i-\ell} z_i)s' + (\sum_{i=0}^{\ell} \mathsf{id}_{1,i} z_i)u$
$\sigma = (\sigma_{1,1}, \sigma_{1,2}, \sigma_2) = ([s']_1, [s']_2, [v']_1)$
Parse $\mathbb{F} = \bigvee_{j=1}^{k} \bigwedge_{\mathsf{id}_{j,i}=1} 1$
For all $j \in [1, k]$, compute :
$r_j \xleftarrow{\$} \mathbb{Z}_p$
$c_{j,0} = ar_j$
$c_{j,1} = (\sum_{i=1}^{\ell} \mathsf{id}_{j,i} z_i) \cdot r_j$
$K_j = az' \cdot r_j + \sigma_2 \in \mathbb{Z}_p$
Return $C = [(c_{j,0}, c_{j,1})_j]_2$, $\mathsf{K} = [K_j]_T$, $\sigma_1 = ([s']_1, [s']_2)$.

$\underline{\mathsf{USKUp}(\mathsf{usk}[\mathsf{id}_2], \tilde{\mathsf{id}}, \mathsf{dk}) :}$
If $\neg(\mathsf{id}_2 \preceq \tilde{\mathsf{id}})$, then return $\perp$.
Set $\mathcal{I} = \{i | \tilde{\mathsf{id}}[i] = 1 \wedge \mathsf{id}_2[i] = 0\}$
Upgrading the key :
$\hat{v} = v + \sum_{i \in I} \mathsf{id}_{2,S_j,i} d_i \in \mathbb{Z}_p$
$\mathsf{uusk}[\tilde{\mathsf{id}}] = ([s]_1, [\hat{v}]_1)$
Return $\mathsf{uusk}[\mathsf{id}]$

$\underline{\mathsf{Verify}(\mathsf{mpk}, \mathsf{uusk}[\tilde{\mathsf{id}}], m, C, \mathsf{K}, \sigma_1) :}$
Parse $\mathsf{uusk}[\tilde{\mathsf{id}}] = ([s]_1, [\hat{v}]_1)$.
Find a j, where $\mathsf{id}_j \in \mathbb{F} \wedge \mathsf{id} \preceq \mathsf{id}_j$
From C, K, extract $([c_{j,0}]_2, [c_{j,1}]_2), [K_j]_T$.
Check whether :
$[c_0 \mathsf{uusk}_2 - c_1 \mathsf{uusk}_1]_T =$
$[\mathsf{K} - (\sigma_{1,2}(\sum_{i=0}^{\ell} \mathsf{id}_i z_i + \sum_{i=\ell+1}^{2\ell} m_{i-\ell} z_i))]_T$
and $[\sigma_{1,1}]_T = [\sigma_{1,2}]_T$

FIGURE 6.4 – Construction ABDVS basée sur SXDH

— $\mathsf{KeyGen}_{\mathsf{Sign}}$: génère les clés de signature sk et ek.

— KeyGen : génère les clés de vérifications usk for id_2, ainsi que dk qui est utilisée pour randomiser usk.

Grâce au mécanisme d'encapsulation du protocole DIBE, on masque la signature avec la clé K. La capsule, notée C, permet la vérification de la signature dans l'algorithme Decap.

Dans l'algorithme USKUp, on modifie la sous clé usk contenue dans la clé secrète uusk pour que usk corresponde à la politique associée à la signature. De cette manière, le vérifieur désigné peut avec uusk vérifier la signature si uusk contient les attributs requis.

Decap vérifie la signature avec la décapsulation de C.

L'ensemble de ces algorithmes permet donc à un utilisateur de signer un message tout en désignant un ensemble d'utilisateurs qui peuvent vérifier la signature. Cette construction doit également satisfaire un certain niveau de sécurité.

6.5 Sécurité

Au niveau de la sécurité du protocole, notre schéma ABDVS doit vérifier les trois propriétés suivantes : résistant aux contrefaçons (Théorème 1), non-transférable (Théorème 2) ainsi que la *perfect privacy* (Théorème 3).

Nous démontrons tout d'abord l'exactitude de notre schéma. Une signature générée de manière honnête doit pouvoir être vérifiée par un vérifieur honnête possédant les bons attributs. Soit la signature générée $C = [(c_{j,0}, c_{j,1})_j]_2$, $\mathsf{K} = [K_j]_T$, $\sigma_1 = ([s']_1, [s']_2)$.

En simplifiant le premier couplage de la vérification, nous obtenons l'équation suivante :

$$[c_0 \mathsf{uusk}_2 - c_1 \mathsf{uusk}_1]_T = [\mathsf{K} - \sigma_1 \sum_{i=0}^{\ell} \tilde{\mathsf{id}}_i z_i \sum_{i=\ell+1}^{2\ell} m_{i-\ell} z_i]_T$$

L'équation précédente est obtenue, de manière honnête, lorsque $\mathsf{uusk}_2 = \sum_{i=1}^{l} \mathsf{id}_i z_i$, qui est dans c_1. Nous pouvons également observer que cette égalité est vérifiée tant que les attributs détenus par le vérifieur sont ceux exigés par l'utilisateur ayant généré la signature. En utilisant la bilinéarité, on obtient $v' = az' + u \sum_{i=0}^{2\ell} \tilde{id}_i z_i = az' + \sigma_1 \sum_{i=0}^{\ell} \tilde{\mathsf{id}}_i z_i + \sum_{i=\ell+1}^{2\ell} m_{i-\ell} z_i$.

Theorem 1. *De part la résistance aux contrefaçons du* DIBE *utilisé, notre construction est résistante aux contrefaçons sous l'hypothèse SXDH.*

Démonstration. Comme présenté dans l'article [36] par Cui et al., une signature générée avec la transformation de Naor ne peut pas être forgée si le schéma IBE utilisé est sémantiquement sécurisé.

Le schéma IBE utilisé dans notre construction est IND-ID-CPA sous l'hypothèse BDH. Par conséquent, notre ABDVS est résistant aux contrefaçons sous l'hypothèse SXDH.

□

Theorem 2. *Notre schéma ABDVS est non-transférable sous l'hypothèse de sécurité* IND-ID-CPA *du schéma* IBE.

Démonstration. Soit $\mathcal{A}$ un adversaire contre la non-transférabilité de notre signature. Dans cette preuve, un simulateur $\mathcal{B}$ donne accès à $\mathcal{A}$ aux algorithmes USKGen, USKUp et Encap et se sert de $\mathcal{A}$ pour casser la sécurité IND-ID-CPA du schéma IBE.

$\underline{\mathsf{Sign}(\mathsf{id}^*)}$: G_0, $\boxed{G_1}$

Parse $\mathbb{F} = \bigvee_{j=1}^{k} \mathsf{id}^{(j)}$

For all $j \in [1, k]$, compute :

$r_j \xleftarrow{\$} \mathbb{Z}_p$

$\alpha \xleftarrow{\$} \mathbb{Z}_p$

$c_{j,0} = a \cdot r_j \in \mathbb{Z}_p$

$c_{j,1} = (\sum_{i=1}^{\ell} \mathsf{id}_i z_i) \cdot r_j$

$K_j = z' \cdot r_j + \sigma_2$

$\boxed{K_j = z' \cdot r_j + \alpha}$

Return $C = [(c_{j,0}, c_{j,1})_j]_2$ and $K = [K_j]_T$.

FIGURE 6.5 – Jeux de sécurité $\mathbf{G_0}$-$\mathbf{G_1}$ pour la non-transferabilité

Game 0. C'est le jeu réel, on ne modifie par les algorithmes existants.

Game 1. $\mathcal{A}$ effectue m requêtes USKGen et Encap pour des identités différentes de celle qui est challengée. Le simulateur répond honnêtement à chaque requête.

Pour l'identité challengée, id^*, on remplace la signature σ_2 par un élément de groupe aléatoire α. Cette modification dans l'algorithme Sign est présentée en Figure 6.5.

C'est à dire que l'on randomise une partie de la clé K_j. L'indiscernabilité d'une clé légitime par rapport à une clé aléatoire est assurée par la propriété d'indiscernabilité du schéma DIBE utilisé dans notre construction. Par conséquent, $\mathbf{G_1}$ est indiscernable de $\mathbf{G_0}$.

Donc, on a : $\mathsf{Adv}^{\mathbf{G_0},\mathbf{G_1}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{ANON\text{-}ID\text{-}CPA}}_{\mathsf{IBE}}(\mathcal{A})$. □

Theorem 3. *Notre signature accomplit la perfect privacy sous l'hypothèse de sécurité* ANON-ID-CPA *du schéma* IBE

Démonstration. Soit $\mathcal{A}$ un adversaire contre la *perfect privacy* de notre signature. Dans cette preuve, un simulateur $\mathcal{B}$ donne accès à $\mathcal{A}$ aux algorithmes USKGen, USKUp et Encap et se sert de $\mathcal{A}$ pour casser la sécurité ANON-ID-CPA du schéma IBE.

Game 0. C'est le jeu réel, on ne modifie par les algorithmes existants.

Game 1. $\mathcal{B}$ simule la clé de signature associée à différents ensembles d'attributs. Cette simulation est présenté en Figure 6.6.

$\mathcal{A}$ reçoit donc différentes signatures valides générées par ces clés. Si $\mathcal{A}$ est capable de distinguer la différences entre deux signatures, il gagne le jeu. C'est-à-dire que dans notre construction, $\mathcal{A}$ doit distinguer deux chiffrés différents. S'il y arrive, il casse la propriété d'anonymat du schéma IBE utilisé. Donc, on a : $\mathsf{Adv}^{\mathbf{G_0},\mathbf{G_1}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{ANON\text{-}ID\text{-}CPA}}_{\mathsf{IBE}}(\mathcal{A})$. □

$\underline{\mathsf{KeyGen}_{\mathsf{Sign}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id}, \perp) :}$
$s \xleftarrow{\$} \mathbb{Z}_p$
$v = y' + (\sum_1^\ell \mathsf{id}_{1,i} z_i)s$
For $i \in [\ell + 1, 2\ell]$:
  $e_i = z_i \cdot s \in \mathbb{Z}_p$

$\mathsf{usk} := ([s]_1, [s]_2, [v]_1)$
  Return $(\mathsf{usk}, \mathsf{ek} = [e_i]_1)$

$\underline{\mathsf{Sign}(\mathsf{mpk}, \mathsf{id}, \mathsf{usk}, \mathsf{ek}, m, \mathbb{F}) :}$
Parse $\mathbb{F} = \bigvee_{j=1}^k \mathsf{id}_\alpha^{(j)}$
For all $j \in [1, k]$, compute :
  $r_j \xleftarrow{\$} \mathbb{Z}_p$
  $c_{j,0} = a \cdot r_j \in \mathbb{Z}_p$
  $c_{j,1} = (\sum_{i=1}^\ell \mathsf{id}_{\alpha,i} z_i) \cdot r_j$
  $K_j = z' \cdot r_j + \sigma_2$
  Return $C = [(c_{j,0}, c_{j,1})_j]_1$ and $K = [K_j]_T$.

Parse $\mathbb{F} = \bigvee_{j=1}^k \mathsf{id}_\beta^{(j)}$
For all $j \in [1, k]$, compute :
  $r_j \xleftarrow{\$} \mathbb{Z}_p$
  $c_{j,0} = a \cdot r_j \in \mathbb{Z}_p$
  $c_{j,1} = (\sum_{i=1}^\ell \mathsf{id}_{\beta,i} z_i) \cdot r_j$
  $K_j = z' \cdot r_j + \sigma_2$
  Return $C = [(c_{j,0}, c_{j,1})_j]_1$ and $K = [K_j]_T$.

FIGURE 6.6 – Jeux de sécurité $\mathbf{G}_0$-$\mathbf{G}_1$ pour la *perfect privacy*

Nous présentons dans la section suivante les résultats expérimentaux de notre construction.

6.6 Résultats expérimentaux

Dans cette section, nous présentons dans un premier temps la complexité théorique de notre construction ainsi qu'une comparaison de celle-ci avec d'autres schémas. Nous nous concentrons d'abord sur le coût des communications en se basant sur la taille des clés et des signatures. Nous détaillons dans un second temps nos résultats expérimentaux, en les comparant avec une implémentation du schéma de Fan et al. [51].

6.6.1 Comparaison théorique

	Type	PK Size	Sign. Size	Hypothesis
[74]	ABS	$(4n+1)\cdot\mathbb{G}_1$	$2\cdot\mathbb{G}_2$	SXDH
[78]	DVS	$1\cdot\mathbb{G}_1$	$2\cdot\mathbb{Z}_p$	DBDH
[51]	ABDVS	$2n\cdot\mathbb{Z}_p$	$3k\cdot\mathbb{G}+2u\cdot\mathbb{G}+5\cdot\mathbb{Z}_p$	DBDH+ROM
Our ABDVS	ABDVS	$n\cdot\mathbb{G}_1$	$3k\cdot\mathbb{G}_1+k\cdot\mathbb{Z}_p$	SXDH

TABLE 6.1 – Comparaison avec d'autres schémas

Dans la Table 6.1, nous comparons notre construction avec d'autres schémas existants. Il est important de noter que seul le schéma de Fan et al. offre le même niveau de sécurité que notre construction comme nous le montrons dans la Table 6.2. Les deux autres schémas auxquels nous nous comparons dans la Table 6.1 permettent de mettre en parallèle les tailles des clés et des signatures avec des primitives similaires. Dans chacun de ces schémas, les éléments dans le groupe cible sont évalués en tant que chaîne de bit hachées pour gagner en efficacité.

Nous héritons des constructions attribute-based, la taille de la clé publique est donc linéaire par rapport au nombre n d'attributs. Contrairement aux autres schémas DVS, nous ne sommes pas dépendant du nombre d'utilisateurs u dans le système au niveau de la taille de la signature. Nous restons linéaire par rapport au nombre de disjonctions k dans la politique.

	Fan *et al.*	Our ABDVS
Setup	$3\ell\cdot t_{inv}+9\ell\cdot t_m$	$(2\ell+1)\cdot t_e$
Sign	$5\cdot t_p+1\cdot t_{inv}+(k+7\ell)\cdot t_m$	$3\ell\cdot t_e+k\cdot(\ell+1)\cdot t_m+\ell\cdot t_p$
Verify	$5\cdot t_p+1\cdot t_{inv}+5\cdot t_s+5\ell\cdot t_m$	$4\cdot t_p+\ell\cdot t_m$

TABLE 6.2 – Comparaison théorique avec le schéma de Fan *et al.*

Dans la Table 6.2, les temps d'exécution des algorithmes Setup,Sign et Verify de notre construction sont présentés. t_{inv} représente le temps nécessaire pour effectuer une inversion ; t_m représente le temps nécessaire pour effectuer une multiplication entre deux éléments de groupe ; t_e représente le temps nécessaire pour effectuer une exponentiation modulaire entre un élément de groupe et un scalaire dans $\mathbb{Z}_p$; t_p représente le temps nécessaire pour effectuer une opération de couplage entre deux éléments de groupe et t_s représente le temps nécessaire pour effectuer une multiplication scalaire dans $\mathbb{G}_1$. Pour rappel, ℓ représente le nombre d'attributs et k le nombre de disjonctions qui sont utilisées.

En se basant sur ces opérations, la Table 6.2 illustre que notre construction a une meilleure complexité que la construction de Fan *et al.* pour les algorithmes Sign et Verify. Il est important de noter qu'à la différence de la Table présente dans le papier original, nous considérons que l'algorithme Sign de Fan *et al.* est dépendant de manière linéaire à la taille de la politique utilisée, tandis que leur algorithme Verify effectue une opération pour chaque nœud de l'arbre. La construction de Fan *et al.* possède de meilleurs résultats pour l'algorithme Setup, qui est uniquement composé d'inversions et de multiplications entre des éléments de groupe, tandis que nous effectuons des exponentiations modulaires, des opérations qui sont plus longues à effectuer.

Cette différence n'est pas un problème pour une application réelle puisque cette étape n'est effectuée qu'une seule fois, avant la communication entre les parties (et donc hors ligne).

6.6.2 Résultats expérimentaux

Dans cette section, nous présentons les résultats expérimentaux de notre construction. L'objectif de ce prototype et de ces résultats expérimentaux est d'illustrer l'efficacité et la praticabilité de notre schéma dans des environnements réels. Ces expérimentations ont été mises en places sur des machines virtuelles disposant de 4 Go de RAM et d'un processeur Intel Core i7 de 2.6 GHz.

À notre connaissance, Fan et al. sont les seuls à avoir proposé un schéma Attribute-Based Designated Verifier Signature. C'est pourquoi nos résultats sont comparés à leur solution.

Notre schéma est implémenté à l'aide de la librairie Charm-crypto Python [4] en utilisant la courbe elliptique asymétrique *MNT159* disponible sur Charm, tandis que le schéma de Fan et al. est implémenté avec la courbe elliptique symétrique *SS512*.

À noter que les deux schémas utilisent des courbes différentes puisqu'ils utilisent des opérations de couplage différentes (asymétrique vs symétrique).

Au niveau des performances, l'écart entre nos temps et ceux du schéma de Fan et al. s'explique pour deux raisons. Tout d'abord, le type de courbe utilisé dans notre schéma permettent d'effectuer des opérations bien plus rapidement. Ensuite, la gestion de la politique utilisée permet de réduire au maximum la perte d'efficacité lorsque le nombre de disjonctions augmente.

Pour l'ensemble de ces expérimentations, l'opération de signature est effectuée sur un haché de 128 bits, qui représente le haché du message que l'on souhaite signer. Les temps de calcul sont

TABLE 6.3 – Performances ABDVS pour 10 attributs

Algorithme	[51]	Notre construction
Setup	22 ms	62 ms
KeyGen	71 ms	34 ms
USKUp	$\emptyset$	< 1 ms
Sign	63 ms	22 ms
Verify	207 ms	7 ms

présentés en Table 6.3 et ont été calculé pour 10 attributs.

Comme illustré dans la Table 6.3, notre schéma est moins efficace au niveau de l'algorithme Setup que le schéma de [51]. Il reste cependant plus performant au niveau de l'algorithme KeyGen, puisque même si les deux versions dépendent du nombre d'attributs, nous manipulons des clés de plus petite taille. Il est également important de noter que dans notre construction, l'algorithme KeyGen génère à la fois une clé de signature et de vérification.

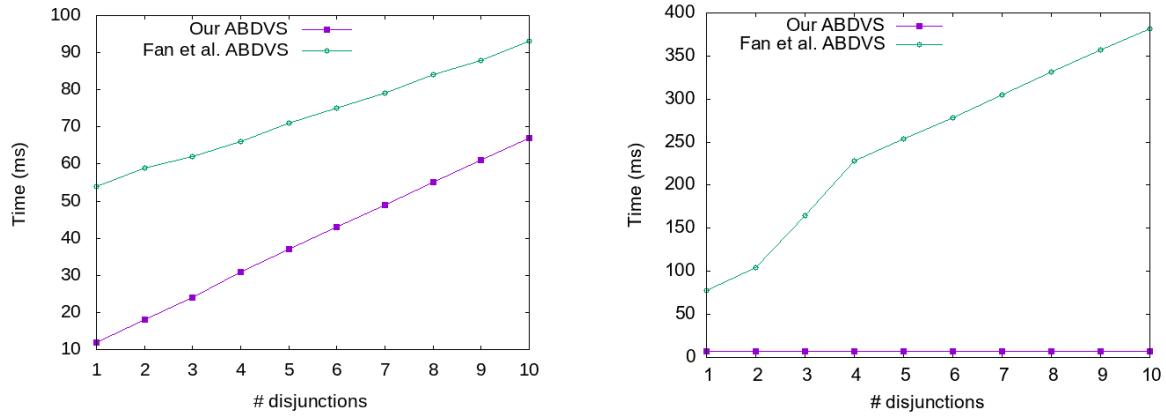
On observe également que l'algorithme Verify de notre construction possède de meilleures performances. En effet, les utilisateurs peuvent repérer au préalable les disjonctions validées par leurs attributs lors de la vérification de la validité d'une signature. Cela permet d'éviter un coût linéaire, et donc d'obtenir un temps de vérification constant.

Comme illustré dans les Figures 6.7a et 6.7b, nos résultats expérimentaux pour les algorithmes Sign et Verify sont plus efficaces que ceux de la construction de Fan *et al.*. Leur algorithme de signature dépend, tout comme le notre, du nombre de disjonctions dans la politique, mais demande plus d'opérations. Le nombre d'opérations de couplage demandées dans l'algorithme Verify reste quant à lui constant.

6.7 Conclusion du chapitre

Nous avons montré dans ce chapitre une nouvelle construction d'ABDVS dans le modèle standard. Cette construction repose sur le schéma IBE proposé par Blazy-Kiltz-Pan, la transformation de Naor ainsi que la transformation pour passer d'un DIBE à un ABE.

Afin d'illustrer l'intérêt d'une telle signature, nous avons également proposé un scénario de santé où l'objectif est de garantir l'authenticité et l'intégrité des données tout en protégeant la vie privée du patient. Dans notre scénario, il est également primordial que la signature soit



(a) Signature (b) Vérification

FIGURE 6.7 – Résultats expérimentaux de [51] et notre ABDVS

vérifiable par des entités différentes.

Notre construction offre de meilleurs résultats expérimentaux que la proposition de Fan et al. pour l'opération de signature et de vérification. Puisque ce sont les opérations répétées le plus souvent dans un scénario pratique, cela rend notre schéma plus utilisable dans une situation réelle.

Pour finir, notre construction peut être généralisée afin de rendre possible au signeur de signer avec des attributs plutôt que son identité. Cette propriété ouvre vers de nouvelles perspectives pour des applications spécifiques au domaine de la santé où il peut être pratique d'utiliser des attributs plutôt que son identité, comme nous avons pu le montrer dans les solutions de sécurité présentées aux Chapitres 4 et 5.

Cette construction offre la possibilité d'améliorer la sécurité des solutions précédentes en étant utilisée pour protéger la vie privée des patients. Elle conclut les contributions présentées dans cette thèse.

Chapitre 7

Conclusion

L'utilisation des objets connectés dans le domaine de la santé a connu une forte croissance ces derniers années. De part la quantité des données qu'ils collectent et leur diversité, ils permettent d'améliorer la prise en charge des patients et d'assurer des soins de meilleures qualités dans les établissements de santé. Ils offrent également la possibilité de mettre en place des environnements d'aide à l'autonomie à domicile, afin de faciliter la vie des personnes âgées en leur permettant de rester à domicile plus longtemps. Cependant, ces nouveaux soins et ces nouveaux environnements s'accompagnent de nombreuses problématiques de sécurité, qui sont associées à la confidentialité des données personnelles des patients ainsi qu'à la protection de leur vie privée. Afin de sécuriser ces données, il est important de considérer les architectures de communications utilisées pour effectuer les échanges de ces données entre les équipements médicaux et les applications qui vont les manipuler/utiliser.

7.1 Récapitulatif des contributions

Dans cette thèse, nous avons travaillé sur l'architecture de communication publish/subscribe, qui est une architecture permettant de découpler les objets connectés produisant des données et les utilisateurs/applications qui les consomment. Cette architecture permet également de gérer un très grand nombre d'utilisateurs, d'être flexible au niveau de leur ajout et de leur suppression via l'utilisation de plusieurs brokers de messages entre les publieurs et les souscripteurs.

Nous avons étudié cette architecture sous ses deux variantes : broker relais et broker de

stockage. Là où le premier transmet les données des publieurs vers les souscripteurs en temps réel, le deuxième permet la gestion d'un très gros volume de données afin d'avoir une communication asynchrone entre les acteurs. Cela nous a conduit à l'élaboration de deux solutions de sécurité.

— Notre première solution de sécurité est appliquée à une architecture topic-based publish/subscribe à broker-relais, en prenant l'exemple du protocole MQTT. Dans cette solution, nous améliorons la confidentialité du contenu des publications avec le chiffrement CP-ABE. De plus, nous utilisons notre modification du schéma ABKS-UR, qui permet au broker d'effectuer les correspondances entre des publications et des souscriptions chiffrées. Enfin, nous utilisons le schéma ABS pour permettre aux publieurs de s'authentifier aux travers de leurs publications. Nous avons également apporté des résultats expérimentaux très satisfaisants sur l'utilisation et la praticabilité de ces schémas dans cette architecture. Cette contribution a fait l'objet d'une publication dans le journal« IEEE Access » en 2021 [16].

— Ensuite, nous avons présenté l'itération de ce scénario de santé, à plus grande échelle : la gestion des données partagées dans un GHT. L'objectif de cette solution est de sécuriser l'architecture Apache Kafka, tout en protégeant les intérêts des souscripteurs vis-à-vis des serveurs Zookeeper. Notre solution utilise un schéma CP-ABE avec plusieurs autorités, de manière à correspondre à ce scénario à plus grande échelle. Combiné à ce schéma, nous utilisons le schéma de chiffrement AES afin de garantir la confidentialité des données stockées sur les brokers Kafka. Enfin, nous utilisons le protocole XPIR pour restreindre les informations obtenues par les serveurs Zookeeper lors de la souscription d'un nouveau consommateur de données. Cette contribution fait l'objet d'un article qui est actuellement en cours de finalisation et fera l'objet d'une soumission future.

— Enfin, nous avons présenté la construction d'une signature à vérifieur désigné basé sur des attributs, notée ABDVS. Ce schéma permet, dans une architecture publish/subscribe, aux publieurs de désigner des vérifieurs avec des attributs pour que ceux-ci aient une preuve d'authentification anonyme non opposable. Notre construction repose sur le schéma de chiffrement DIBE, sur lequel nous appliquons la transformation de Naor pour en faire une signature. Cette construction nous permet d'assurer les propriétés de sécurité suivantes à notre signature : résistance aux contrefaçons, non-transférabilité et respect de la vie privée

de l'utilisateur. Cette dernière contribution a passé la première phase de revues et est actuellement en cours de révision dans le journal "Journal of Ambient Intelligence and Humanized Computing".

7.2 Travaux futurs

Les travaux effectués dans cette thèse proposent des solutions pour sécuriser des architectures publish/subscribe, appliquées au domaine de la santé et plus spécifiquement aux scénarios AAL et à la gestion de données partagées entre plusieurs établissements de santé.

Ces solutions sont construites avec plusieurs schémas cryptographiques, tel que ABE, ABKS-UR, ABS ou encore notre construction ABDVS. L'harmonisation de ces schémas appliqués à une architecture décentralisée comme le publish/subscribe représente la contribution majeure de cette thèse. Dans la continuité de ces travaux, nous avons identifié plusieurs pistes de recherche, dont certaines que nous avons commencé à explorer.

Tout d'abord, nous proposons une solution de sécurité pour une architecture publish/subscribe avec un broker relais type MQTT, où le processus de correspondance s'effectue en utilisant un schéma ABKS-UR. Cependant, même si ce schéma est efficace, il peut être intéressant de limiter au maximum le nombre de fois où ce processus de correspondance est appelé. Pour cela, nous souhaitons intégrer un schéma de filtrage de souscriptions, tel qu'un Bloom Filter qui est très utilisé en pratique [22] ou un Cuckoo Filter qui offre de meilleurs résultats expérimentaux [50]. Cette intégration qui a été réalisée sur un premier prototype montre des résultats intéressant en permettant de réduire le temps de réponse du broker lors d'une nouvelle souscription d'un souscripteur. Une évaluation à plus grande échelle reste encore à mener.

Une autre amélioration de nos travaux consiste à utiliser des cryptosystèmes basés sur des problèmes difficiles résistants à l'ordinateur quantique. En effet, les schémas de chiffrement, de searchable encryption et de signature que nous utilisons dans nos solutions reposent sur des problèmes difficiles dans les courbes elliptiques et sont donc vulnérables à l'arrivée de l'ordinateur quantique [30]. Il existe par ailleurs dans la littérature des schémas ABE résistants à l'ordinateur quantique [20] puisqu'ils sont basés sur des problèmes difficiles reposant sur les réseaux euclidiens. Un nouveau protocole de searchable encryption basé sur les réseaux euclidiens est actuellement

en cours de définition pour permettre de répondre à ce défi.

Publications et interventions

Interventions

— 09/2019 : Présentation de l'article *An IoT Attribute-Based Security Framework for Topic-Based Publish/Subscribe Systems* dans le cadre du projet MIRES SPOK2 (conjoint avec le projet région SVP-IoT) à l'université de La Rochelle, France.

— 03/2019 : Présentation d'un poster portant sur la sécurité des architecture publish/subscribe lors du forum du SILPC à Beaublanc, Limoges, France.

— 05/2019 : Stage doctoral d'un mois portant sur la sécurité des objets connectés dans un environnement d'aide à l'autonomie à domicile à l'Université de Sherbrooke, Quebec.

Publications

[BCMS21] An IoT Attribute-Based Security Framework for Topic-Based Publish/Subscribe Systems (Olivier Blazy, Emmanuel Conchon, Mathieu Klingler, Damien Sauveron), In *IEEE Access, 2021, vol. 9, p. 19066-19077.*

En cours de soumission

1. *Efficient and Round-Optimal Secret Handshake* avec Olivier Blazy, Céline Chevalier, Emmanuel Conchon, Neals Fournaise et Olivier Levillain.
2. *Attribute-based Designated Verifier Signature* avec Olivier Blazy, Emmanuel Conchon et Laura Brouilhet.

3. *A Secure Apache Kafka Architecture* avec Olivier Blazy, Emmanuel Conchon, Neals Fournaise et Damien Sauveron.

Bibliographie

[1] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes : Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4) :1–35, 2018.

[2] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. Xpir : Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, 2016(2) :155 – 174, 01 Apr. 2016.

[3] Carlos Aguilar-Melchor and Philippe Gaborit. A lattice-based computationally-efficient private information retrieval protocol. *Cryptol. ePrint Arch., Report*, 446, 2007.

[4] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. Charm : a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2) :111–128, 2013.

[5] Ebrahim Al Alkeem, Chan Yeob Yeun, and M Jamal Zemerly. Security and privacy framework for ubiquitous healthcare iot devices. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 70–75. IEEE, 2015.

[6] Hamidreza Arasteh, Vahid Hosseinnezhad, Vincenzo Loia, Aurelio Tommasetti, Orlando Troisi, Miadreza Shafie-khah, and Pierluigi Siano. Iot-based smart cities : A survey. In *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1–6. IEEE, 2016.

[7] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno De Medeiros. Practical group signatures without random oracles. *IACR Cryptol. ePrint Arch.*, 2005 :385, 2005.

[8] A Badii, J Khan, M Crouch, and S Zickau. Hydra : Networked embedded system middleware for heterogeneous physical devices in a distributed architecture. In *Final External Developers Workshops Teaching Materials*, page 4, 2010.

[9] Guruduth Banavar, Tushar Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E Strom, and Daniel C Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*, pages 262–272. IEEE, 1999.

[10] Guruduth Banavar, Tushar Chandra, Robert Strom, and Daniel Sturman. A case for message oriented middleware. In *International Symposium on Distributed Computing*, pages 1–17. Springer, 1999.

[11] Andrew Banks and Rahul Gupta. Mqtt version 3.1. 1. *OASIS standard*, 29, 2014.

[12] Raphaël Barazzutti, Thomas Heinze, André Martin, Emanuel Onica, Pascal Felber, Christof Fetzer, Zbigniew Jerzak, Marcelo Pasin, and Etienne Riviere. Elastic scaling of a high-throughput content-based publish/subscribe engine. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 567–576. IEEE, 2014.

[13] Umesh Bellur, Nanjangud C Narendra, and Swarup Kumar Mohalik. Ausom : autonomic service-oriented middleware for iot-based systems. In *2017 IEEE world congress on services (SERVICES)*, pages 102–105. IEEE, 2017.

[14] András Belokosztolszki, David M Eyers, Peter R Pietzuch, Jean Bacon, and Ken Moody. Role-based access control for publish/subscribe middleware architectures. In *Proceedings of the 2nd international workshop on Distributed event-based systems*, pages 1–8. ACM, 2003.

[15] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE, 2007.

[16] Olivier Blazy, Emmanuel Conchon, Mathieu Klingler, and Damien Sauveron. An iot attribute-based security framework for topic-based publish/subscribe systems. *IEEE Access*, 9 :19066–19077, 2021.

[17] Olivier Blazy, Paul Germouty, and Duong Hieu Phan. Downgradable identity-based encryption and applications. In *Cryptographers' Track at the RSA Conference*, pages 44–61. Springer, 2019.

[18] Olivier Blazy, Paul Germouty, and Duong Hieu Phan. Downgradable identity-based encryption and applications. In *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, pages 44–61, 2019.

[19] Hendrik Bohn, Andreas Bobek, and Frank Golatowski. Sirena-service infrastructure for real-time embedded networked devices : A service oriented framework for different domains. In *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, pages 43–43. IEEE, 2006.

[20] Xavier Boyen. Attribute-based functional encryption on lattices. In *Theory of Cryptography Conference*, pages 122–142. Springer, 2013.

[21] Stefan Brenner, Colin Wulf, David Goltzsche, Nico Weichbrodt, Matthias Lorenz, Christof Fetzer, Peter Pietzuch, and Rüdiger Kapitza. Securekeeper : confidential zookeeper using intel sgx. In *Proceedings of the 17th International Middleware Conference*, pages 1–13, 2016.

[22] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters : A survey. *Internet mathematics*, 1(4) :485–509, 2004.

[23] Niels Brouwers and Koen Langendoen. Pogo, a middleware for mobile phone sensing. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 21–40. Springer, 2012.

[24] Fengyun Cao and Jaswinder Pal Singh. Efficient event routing in content-based publish-subscribe service networks. In *IEEE INFOCOM 2004*, volume 2, pages 929–940. IEEE, 2004.

[25] Mauro Caporuscio, Pierre-Guillaume Raverdy, and Valerie Issarny. ubisoap : A service-oriented middleware for ubiquitous networking. *IEEE Transactions on Services Computing*, 5(1) :86–98, 2010.

[26] Antonio Carzaniga, David S Rosenblum, and Alexander L Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, 19(3) :332–383, 2001.

[27] Anil Chacko and Thaier Hayajneh. Security and privacy issues with iot in healthcare. *EAI Endorsed Transactions on Pervasive Health and Technology*, 4(14), 2018.

[28] Melissa Chase. Multi-authority attribute based encryption. In *Theory of cryptography conference*, pages 515–534. Springer, 2007.

[29] David Chaum and Hans Van Antwerpen. Undeniable signatures. In *Conference on the Theory and Application of Cryptology*, pages 212–216. Springer, 1989.

[30] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[31] Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A vision of iot : Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things journal*, 1(4) :349–359, 2014.

[32] Weifeng Chen, Jianchun Jiang, and Nancy Skocik. On the privacy protection in publish/subscribe systems. In *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, pages 597–601. IEEE, 2010.

[33] Raymond Cheng, William Scott, Bryan Parno, Arvind Krishnamurthy, and Thomas Anderson. Talek : a private publish-subscribe protocol. Technical report, Technical Report. University of Washington, 2016.

[34] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50, 1995.

[35] Iván Corredor, José F Martínez, Miguel S Familiar, and Lourdes Lopez. Knowledge-aware and service-oriented middleware for deploying pervasive services. *Journal of Network and Computer Applications*, 35(2) :562–576, 2012.

[36] Yang Cui, Eiichiro Fujisaki, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Formal security treatments for signatures from identity-based encryption. In *International Conference on Provable Security*, pages 218–227. Springer, 2007.

[37] Edward Curry. Message-oriented middleware. *Middleware for communications*, pages 1–28, 2004.

[38] Daniel Demmler, Amir Herzberg, and Thomas Schneider. Raid-pir : Practical multi-server pir. In *Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security*, CCSW '14, page 45–56, New York, NY, USA, 2014. Association for Computing Machinery.

[39] Y Justin Dhas and P Jeyanthi. A review on internet of things protocol and service oriented middleware. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0104–0108. IEEE, 2019.

[40] Johannes Diechmann, Kersten Heineke, Thomas Reinbacher, and Dominik Wee. The internet of things : How to capture the value of iot. Technical report, Technical Report, Technical Report, 2018.

[41] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976.

[42] Angelika Dohr, Robert Modre-Opsrian, Mario Drobics, Dieter Hayn, and Günter Schreier. The internet of things for ambient assisted living. In *2010 seventh international conference on information technology : new generations*, pages 804–809. Ieee, 2010.

[43] Changyu Dong, Giovanni Russello, and Naranker Dulay. Shared and searchable encrypted data for untrusted servers. *Journal of Computer Security*, 19(3) :367–397, 2011.

[44] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. Advanced encryption standard (aes), 2001-11-26 2001.

[45] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985.

[46] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Rafols, and Jorge Villar. An algebraic framework for diffie–hellman assumptions. *Journal of cryptology*, 30(1) :242–288, 2017.

[47] Christian Esposito and Mario Ciampi. On security in publish/subscribe services : A survey. *IEEE Communications Surveys & Tutorials*, 17(2) :966–997, 2015.

[48] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2) :114–131, 2003.

[49] Françoise Fabret, François Llirbat, Joao Pereira, and Dennis Shasha. Efficient matching for content-based publish/subscribe systems. In *Proc. CoopIS*. Citeseer, 2000.

[50] Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. Cuckoo filter : Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 75–88, 2014.

[51] Chun-I Fan, Chien-Nan Wu, Wei-Kuei Chen, and Wei-Zhe Sun. Attribute-based strong designated-verifier signature scheme. *Journal of Systems and Software*, 85(4) :944–959, 2012.

[52] Nicolaie L Fantana, Till Riedel, Jochen Schlick, Stefan Ferber, Jürgen Hupp, Stephen Miles, Florian Michahelles, and Stefan Svensson. Iot applications—value creation for industry. *Internet of Things : Converging technologies for smart environments and integrated ecosystems*, page 153, 2013.

[53] Bahar Farahani, Farshad Firouzi, and Krishnendu Chakrabarty. Healthcare iot. In *Intelligent Internet of Things*, pages 515–545. Springer, 2020.

[54] Ludger Fiege, Andreas Zeidler, Alejandro Buchmann, Roger Kilian-Kehr, Gero Mühl, and Tu Darmstadt. Security aspects in publish/subscribe systems. In *Third Intl. Workshop on Distributed Event-based Systems (DEBS'04), Edinburgh, Scotland, UK*, page 44–49. IET, 07 2004.

[55] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Servilla : a flexible service provisioning middleware for heterogeneous sensor networks. *Science of Computer Programming*, 77(6) :663–684, 2012.

[56] Anders Fongen and Federico Mancini. Identity management and integrity protection in publish-subscribe systems. In *IFIP Working Conference on Policies and Research in Identity Management*, pages 68–82. Springer, 2013.

[57] Paul Fremantle and Philip Scott. A survey of secure middleware for the internet of things. *PeerJ Computer Science*, 3 :e114, 2017.

[58] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[59] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *International Colloquium on Automata, Languages, and Programming*, pages 803–815. Springer, 2005.

[60] Hemant Ghayvat, S Mukhopadhyay, B Shenjie, Arpita Chouhan, and W Chen. Smart home based ambient assisted living : Recognition of anomaly in the activity of daily living for an elderly living alone. In *2018 IEEE international instrumentation and measurement technology conference (I2MTC)*, pages 1–5. IEEE, 2018.

[61] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, 2003.

[62] Nikesh Gondchawar, RS Kawitkar, et al. Iot based smart agriculture. *International Journal of advanced research in Computer and Communication Engineering*, 5(6) :838–842, 2016.

[63] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

[64] Bhole Rahul Hiraman et al. A study of apache kafka in big data stream processing. In *2018 International Conference on Information, Communication, Engineering and Technology (ICICET)*, pages 1–3. IEEE, 2018.

[65] Danny Hughes, Klaas Thoelen, Wouter Horré, Nelson Matthys, Javier Del Cid, Sam Michiels, Christophe Huygens, and Wouter Joosen. Looci : a loosely-coupled component infrastructure for networked embedded systems. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, pages 195–203, 2009.

[66] Mihaela Ion, Giovanni Russello, and Bruno Crispo. Providing confidentiality in content-based publish/subscribe systems. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–6. IEEE, 2010.

[67] Mihaela Ion, Giovanni Russello, and Bruno Crispo. Design and implementation of a confidentiality and access control solution for publish/subscribe systems. *Computer networks*, 56(7) :2014–2037, 2012.

[68] ITU-T. ITU-T Y.4000/Y.2060 (06/2012). handle.itu.int/11.1002/1000/11559. Accessed on 20.09.2021.

[69] Hans-Arno Jacobsen. *Content-Based Publish/Subscribe*, pages 464–466. Springer US, Boston, MA, 2009.

[70] Hans-Arno Jacobsen. *Topic-based Publish/Subscribe*, pages 3127–3129. Springer US, Boston, MA, 2009.

[71] Hans-Arno Jacobsen, Alex Cheung, Guoli Li, Balasubramaneyam Maniymaran, Vinod Muthusamy, and Reza Sherafat Kazemzadeh. The padres publish/subscribe system. In *Principles and Applications of Distributed Event-Based Systems*, pages 164–205. IGI Global, 2010.

[72] Hans-Arno Jacobsen, Alex Cheung, Guoli Li, Balasubramaneyam Maniymaran, Vinod Muthusamy, and Reza Sherafat Kazemzadeh. The padres publish/subscribe system. In *Principles and Applications of Distributed Event-Based Systems*, pages 164–205. IGI Global, 2010.

[73] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 143–154. Springer, 1996.

[74] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Compact nizks from standard assumptions on bilinear maps. *IACR Cryptol. ePrint Arch.*, 2020 :223, 2020.

[75] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31, 1988.

[76] Marie Kim, Jun Wook Lee, Yong Joon Lee, and Jae-Cheol Ryou. Cosmos : A middleware for integrated data processing over heterogeneous sensor networks. *ETRI journal*, 30(5) :696–706, 2008.

[77] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac : Keyed-hashing for message authentication, 1997.

[78] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures : Anonymity and efficient construction from any bilinear map. In *International Conference on Security in Communication Networks*, pages 105–119. Springer, 2004.

[79] In Lee and Kyoochun Lee. The internet of things (iot) : Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4) :431–440, 2015.

[80] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, and Kui Ren. Attribute-based signature and its applications. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 60–69. ACM, 2010.

[81] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE transactions on parallel and distributed systems*, 24(1) :131–143, 2012.

[82] Huang Lin, Zhenfu Cao, Xiaohui Liang, and Jun Shao. Secure threshold multi authority attribute based encryption without a central authority. In *International Conference on Cryptology in India*, pages 426–436. Springer, 2008.

[83] Luca Mainetti, Luigi Patrono, Andrea Secco, and Ilaria Sergi. An iot-aware aal system for elderly people. In *2016 International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, pages 1–6. IEEE, 2016.

[84] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. *Hasp@ isca*, 10(1), 2013.

[85] Yasser Mesmoudi, Mohammed Lamnaour, Yasser El Khamlichi, Abderrahim Tahiri, Abdellah Touhafi, and An Braeken. A middleware based on service oriented architecture for heterogeneity issues within the internet of things (msoah-iot). *Journal of King Saud University-Computer and Information Sciences*, 32(10) :1108–1116, 2020.

[86] Zoltán Miklós. Towards an access control mechanism for wide-area publish/subscribe systems. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 516–521. IEEE, 2002.

[87] Subhas Chandra Mukhopadhyay. Wearable sensors for human activity monitoring : A review. *IEEE sensors journal*, 15(3) :1321–1330, 2014.

[88] S Muthuramalingam, A Bharathi, N Gayathri, R Sathiyaraj, B Balamurugan, et al. Iot based intelligent transportation system (iot-its) for global perspective : A case study. In *Internet of Things and Big Data Analytics for Smart Generation*, pages 279–300. Springer, 2019.

[89] Online. Cnil. `www.cnil.fr`. Accessed : 2021-05-26.

[90] Online. CNIL. `www.cnil.fr/fr/reglement-europeen-protection-donnees`. Accessed on 04.08.2021.

[91] Online. Confluent.io. `www.confluent.io/blog/removing-zookeeper-dependency-in-kafka/`. Accessed on 25.08.2021.

[92] Online. esante.gouv.fr. `esante.gouv.fr`. Accessed on 04.08.2021.

[93] Online. Hadoop. `hadoop.apache.org`. Accessed on 09.12.2020.

[94] Online. Hive. `hive.apache.org`. Accessed on 09.12.2020.

[95] Online. Kafka. `kafka.apache.org`. Accessed on 09.12.2020.

[96] Online. Ministère de la santé. `solidarites-sante.gouv.fr`. Accessed : 2021-05-26.

[97] Online. RabbitMQ. `www.rabbitmq.com`. Accessed on 04.08.2021.

[98] Online. Zookeeper. `zookeeper.apache.org`. Accessed on 09.12.2020.

[99] Online. HBMQTT. `github.com/beerfactory/hbmqtt`, 2015. Accessed on 09.12.2020.

[100] Online. Paho-mqtt, mqtt-sn software. `https://www.eclipse.org/paho/`, 2018. Accessed on 09.12.2020.

[101] Lukasz Opyrchal and Atul Prakash. Secure distribution of events in content-based publish subscribe systems. *Ann Arbor*, 1001 :48109–2122, 2001.

[102] Fulya Ozturk and Ayse Meliha Ozdemir. Content-based publish/subscribe communication model between iot devices in smart city environment. In *2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG)*, pages 189–193. IEEE, 2019.

[103] Ganapathi Padmavathi and Samukutty Annadurai. A security framework for content-based publish–subscribe system. *Electronic Commerce Research and Applications*, 5(1) :78–90, 2006.

[104] Mike P Papazoglou. Service-oriented computing : Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.*, pages 3–12. IEEE, 2003.

[105] Zhen Peng, Zhao Jingling, and Liao Qing. Message oriented middleware data processing model in internet of things. In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, pages 94–97. IEEE, 2012.

[106] BT Prasanna and CB Akki. A comparative study of homomorphic and searchable encryption schemes for cloud computing. *arXiv preprint arXiv :1505.03263*, 2015.

[107] NIST FIPS PUB. 46-3. data encryption standard. *Federal Information Processing Standards, National Bureau of Standards, US Department of Commerce*, 1977.

[108] Huiling Qian, Jiguo Li, Yichen Zhang, and Jinguang Han. Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation. *International Journal of Information Security*, 14(6) :487–497, 2015.

[109] Shiyou Qian, Jian Cao, Yanmin Zhu, and Minglu Li. Rein : A fast event matching approach for content-based publish/subscribe systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2058–2066. IEEE, 2014.

[110] Costin Raiciu and David S Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. In *Securecomm and Workshops, 2006*, pages 1–11. IEEE, 2006.

[111] Gowri Sankar Ramachandran, Kwame-Lante Wright, Licheng Zheng, Pavas Navaney, Muhammad Naveed, Bhaskar Krishnamachari, and Jagjit Dhaliwal. Trinity : A byzantine fault-tolerant distributed publish-subscribe system with immutable blockchain-based persistence. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 227–235. IEEE, 2019.

[112] Marilyn J Rantz, Marjorie Skubic, Richelle J Koopman, Lorraine Phillips, Gregory L Alexander, Steven J Miller, and Rainer Dane Guevara. Using sensor networks to detect urinary tract infections in older adults. In *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*, pages 142–149. IEEE, 2011.

[113] Parisa Rashidi and Diane J Cook. Keeping the resident in the loop : Adapting the smart home to the user. *IEEE Transactions on systems, man, and cybernetics-part A : systems and humans*, 39(5) :949–959, 2009.

[114] Parisa Rashidi and Alex Mihailidis. A survey on ambient-assisted living tools for older adults. *IEEE journal of biomedical and health informatics*, 17(3) :579–590, 2012.

[115] Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and Siobhán Clarke. Middleware for internet of things : a survey. *IEEE Internet of things journal*, 3(1) :70–95, 2015.

[116] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[117] Yannis Rouselakis and Brent Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security*, pages 315–332, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[118] Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz. Music : Middleware support for self-adaptation in ubiquitous and service-oriented environments. In *Software engineering for self-adaptive systems*, pages 164–182. Springer, 2009.

[119] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 457–473. Springer, 2005.

[120] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[121] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[122] Brian Shand, Peter Pietzuch, Ioannis Papagiannis, Ken Moody, Matteo Migliavacca, David M Eyers, and Jean Bacon. Security policy and information sharing in distributed

event-based systems. In *Reasoning in Event-Based Distributed Systems*, pages 151–172. Springer, 2011.

[123] Saeed Shokrollahi and Fereidoon Shams. Rich device-services (rds) : a service-oriented approach to the internet of things (iot). *Wireless Personal Communications*, 97(2) :3183–3201, 2017.

[124] Patrik Spiess, Stamatis Karnouskos, Dominique Guinard, Domnic Savio, Oliver Baecker, Luciana Moreira Sá De Souza, and Vlad Trifa. Soa-based integration of the internet of things in enterprise services. In *2009 IEEE international conference on web services*, pages 968–975. IEEE, 2009.

[125] OASIS Standard. Mqtt version 3.1. 1. *URL http ://docs. oasis-open. org/mqtt/mqtt/v3*, 1, 2014.

[126] Hong Sun, Vincenzo De Florio, Ning Gui, and Chris Blondia. Promises and challenges of ambient assisted living systems. In *2009 Sixth International Conference on Information Technology : New Generations*, pages 1201–1207. Ieee, 2009.

[127] Wenhai Sun, Shucheng Yu, Wenjing Lou, Y Thomas Hou, and Hui Li. Protecting your right : Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In *INFOCOM, 2014 Proceedings IEEE*, pages 226–234. IEEE, 2014.

[128] Willy Susilo, Fangguo Zhang, and Yi Mu. Identity-based strong designated verifier signature schemes. In *Australasian Conference on Information Security and Privacy*, pages 313–324. Springer, 2004.

[129] Manabu Suzuki. Sanitizable signature with secret information. In *2006 Symposium on Cryptography and Information Security (SCIS 2006)*, pages 4A1–4A2, January 2006.

[130] Liane Margarida Rockenbach Tarouco, Leandro Márcio Bertholdo, Lisandro Zambenedetti Granville, Lucas Mendes Ribeiro Arbiza, Felipe Carbone, Marcelo Marotta, and Jose Jair Cardoso De Santanna. Internet of things in healthcare : Interoperatibility and security issues. In *2012 IEEE international conference on communications (ICC)*, pages 6121–6125. IEEE, 2012.

[131] Dirk Thatmann, Sebastian Zickau, Alexander Förster, and Axel Küpper. Applying attribute-based encryption on publish subscribe messaging patterns for the internet of

things. In *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, pages 556–563. IEEE, 2015.

[132] Vlasios Tsiatsis, Alexander Gluhak, Tim Bauge, Frederic Montagut, Jesus Bernat, Martin Bauer, Claudia Villalonga, Payam Barnaghi, and Srdjan Krco. The sensei real world internet architecture. In *Towards the Future Internet*, pages 247–256. IoS Press, 2010.

[133] Md Uddin, Weria Khaksar, Jim Torresen, et al. Ambient sensors for elderly care and independent living : a survey. *Sensors*, 18(7) :2027, 2018.

[134] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 24–43. Springer, 2010.

[135] Brent Waters. Ciphertext-policy attribute-based encryption : An expressive, efficient, and provably secure realization. In *International Workshop on Public Key Cryptography*, pages 53–70. Springer, 2011.

[136] Yan Yan, Yi Huang, Geoffrey C Fox, Shrideep Pallickara, Marlon E Pierce, Ali Kaplan, and Ahmet E Topcu. Implementing a prototype of the security framework for distributed brokering systems. In *Security and Management*, pages 212–218, 2003.

[137] Tsz Hon Yuen, Willy Susilo, and Yi Mu. Towards a cryptographic treatment of publish/subscribe systems. In *International Conference on Cryptology and Network Security*, pages 201–220. Springer, 2010.

[138] Ye Zhao, Kyungbaek Kim, and Nalini Venkatasubramanian. Dynatops : A dynamic topic-based publish/subscribe architecture. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 75–86, 2013.

Résumé

Dans cette thèse, nous élaborons des solutions de sécurité pour des architectures publish/subscribe regroupant un grand nombre d'utilisateurs. Pour illustrer ces solutions de sécurité, nous nous basons sur des scénarios autour du domaine de la santé. Notre objectif principal est d'améliorer le contrôle qu'ont les patients sur leurs données, ainsi qu'améliorer la confidentialité des données personnelles au niveau du broker.

Nous proposons tout d'abord une solution de sécurité pour un environnement d'aide à l'autonomie à domicile, avec plusieurs capteurs collectant des données sur les habitants de ces environnements. Dans cette solution, nous utilisons le schéma de chiffrement CP-ABE pour garantir la confidentialité du contenu des publications. Nous modifions et utilisons également le schéma ABKS-UR qui permet de masquer les topics associés aux publications et aux souscriptions au broker. Enfin, nous utilisons le schéma ABS pour permettre aux publieurs de s'identifier au travers de leurs publications.

Ensuite, nous itérons ce scénario dans un contexte à plus grande échelle, avec un ensemble d'établissements de santé qui partagent les données qu'ils collectent. Notre solution de sécurité appliquée à ce scénario chiffre le contenu des publications avec le schéma CP-ABE combiné au schéma AES. Pour masquer les intérêts des utilisateurs aux serveurs Zookeeper, nous utilisons également un protocole PIR associé aux schémas précédents.

Enfin, nous construisons un schéma de signature à vérifieur désigné basé sur des attributs à partir d'un chiffrement basé sur l'identité. Nous prouvons également la sécurité de cette construction sous une hypothèse classique. Ce schéma permet d'améliorer les solutions précédentes en protégeant les publieurs.

Mots-clés : publish/subscribe, internet des objets, cryptographie, sécurité informatique, confidentialité, respect de la vie privée.

Abstract

In this thesis, we design security frameworks for publish/subscribe architecture gathering a large number of users. To illustrate the practicity of these frameworks, we are basing ourselves on healthcare scenarios. Our main concern in those scenarios is to improve the security and confidentiality of patient data at the broker level.

First, we present a security framework for a AAL environment, with multiple sensors that collect data from patients habits. In this framework we use CP-ABE to improve the payload confidentiality of publications. We use and modify an ABKS-UR scheme to protect the topic attached to publications and subscriptions, at the broker level. Finally, we use ABS scheme to allow publishers to authenticate themselves with their publications.

Second, we develop this scenario for a larger number of users, with a group of healthcare institutions that share the data they collect. Our framework use CP-ABE combined to AES scheme to encrypt the payload of publications. To hide user interests from Zookeeper servers, we also use a PIR protocol.

Finally, we construct an Attribute-Based Designated Verifier Signature scheme from identity-based encryption. We also prove the safety of this construction under a classical hypothesis. This scheme improves the previous solutions by protecting publishers.

Keywords : publish/subscribe, internet of things, cryptography, network security, confidentiality, privacy.