Protection des données des véhicules connectés : une approche cryptographique reposant sur le chiffrement basé attributs

Rémi Adelin

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *06/07/2023* **par :**

Rémi ADELIN

Protection des données des véhicules connectés : une approche cryptographique reposant sur le chiffrement basé attributs

---

JURY

Jean-Guillaume DUMAS	Professeur des universités	Président du jury
Yves ROUDIER	Professeur des universités	Rapporteur
Nesrine KAÂNICHE	Maîtresse de conférences	Examinatrice
Vincent NICOMETTE	Professeur des universités	Directeur de thèse
Éric ALATA	Maître de conférences	Co-directeur de thèse
Vincent MIGLIORE	Maître de conférences	Co-directeur de thèse
Alain FILIPOWICZ	Ingénieur	Invité

École doctorale et spécialité :

EDMITT : Informatique et Télécommunications

Unité de Recherche :

Laboratoire d'Analyse et d'Architecture des Systèmes

Directrice et directeur de Thèse :

Vincent NICOMETTE, *Éric ALATA* et *Vincent MIGLIORE*

Rapporteurs :

Jean-Guillaume DUMAS et *Yves ROUDIER*

Remerciements

Durant ces longues années de thèse, j'ai eu le privilège de rencontrer et de côtoyer de nombreuses personnes qui ont toutes contribué, à leur manière, à la réussite de ces travaux. Je souhaite exprimer ma gratitude à chacune d'entre elles dans les remerciements suivants.

En premier lieu, je tiens à remercier les institutions qui ont rendu cette thèse possible. Les travaux présentés dans ce manuscrit ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS). Je suis profondément reconnaissant envers le LAAS-CNRS dans son ensemble. En particulier, je remercie Liviu Nicu et Mohamed Kaâniche, qui ont successivement assuré la direction du LAAS-CNRS depuis mon entrée, pour m'avoir accueilli au sein de ce laboratoire. Je remercie également Mohamed Kaâniche et Hélène Waeselynck, qui ont successivement dirigé l'équipe Tolérance aux fautes et Sûreté de Fonctionnement informatique (TSF), pour m'avoir accueilli et plus particulièrement pour m'avoir soutenu autant administrativement que techniquement dans la réalisation de mes travaux dans cette équipe. Je leur suis reconnaissant pour l'aide inestimable qu'ils m'ont apportée à maintes reprises. Je remercie les services administratifs et techniques du LAAS-CNRS dont le secrétariat de l'équipe TSF pour leur assistance essentielle pendant ma thèse.

Les travaux présentés dans ce manuscrit ont été réalisés en collaboration avec l'entreprise Continental Digital Services France (CDSF), et je tiens à exprimer ma gratitude envers l'ensemble des équipes de CDSF. En particulier, je remercie Alain Filipowicz, responsable partenariats innovation recherche, ainsi que Stève Hugon et Hugues Bonnin, ingénieurs chez CDSF, pour m'avoir également permis de réaliser cette thèse.

Je remercie chaleureusement l'Institut National des Sciences Appliquées (INSA) de Toulouse et l'École Doctorale de Mathématiques, Informatique et Télécommunications de Toulouse (ED-MITT) dont Agnès Requis pour sa gentillesse, son professionnalisme et sa patience.

J'exprime également ma gratitude envers les membres du jury qui ont accepté d'évaluer mon travail. Je leur suis très reconnaissant pour l'intérêt qu'ils ont porté à mes travaux et pour la pertinence de leurs questions :

— Jean-Guillaume Dumas, Professeur des universités à l'Université de Grenoble ;

— Yves Roudier, Professeur des universités à l'Université Côte d'Azur ;

— Nesrine Kaâniche, Maîtresse de conférences à Télécom SudParis ;

— Vincent Nicomette, Professeur des universités à l'Institut National des Sciences Appliquées de Toulouse ;

— Éric Alata, Maître de conférences à l'Institut National des Sciences Appliquées de Toulouse ;

— Vincent Migliore, Maître de conférences à l'Institut National des Sciences Appliquées de Toulouse ;

— Alain Filipowicz, Ingénieur à Continental Digital Services France.

Je tiens à remercier tout particulièrement Jean-Guillaume Dumas pour avoir présidé le jury. Je tiens également à remercier Jean-Guillaume Dumas et Yves Roudier pour avoir accepté de rapporter ma thèse. Les commentaires des rapports ainsi que les échanges lors de la soutenance ont été très enrichissants.

Bien que ces travaux aient été réalisés dans le cadre de ma thèse, ils sont néanmoins le résultat d'un travail collaboratif. Je suis profondément reconnaissant envers mes encadrants pour leur soutien scientifique et moral tout au long de ce parcours. Malgré leurs emplois du temps chargés, ils m'ont guidé et conseillé dans les moments difficiles comme dans les moments plus légers. Je tiens donc à exprimer mon immense gratitude à mon directeur de thèse, Vincent Nicomette, pour sa bonne humeur, son énergie et son humanité. Il a été d'un grand soutien en trouvant les mots justes pour m'accompagner et me guider lorsque j'étais confronté à des doutes. Je suis également très reconnaissant envers mon premier co-directeur de thèse, Éric Alata, il a été d'une aide précieuse en m'apportant des idées constructives et des réponses pragmatiques à mes interrogations. Je souhaite également exprimer ma gratitude envers mon second co-directeur de thèse, Vincent Migliore qui m'a également beaucoup apporté techniquement et scientifiquement. Mes remerciements vont également à mon encadrant industriel à CDSF, Laurent Charnot, pour son enthousiasme, ses conseils et ses retours. Je suis reconnaissant envers Cyrius Nugier, qui a commencé en stage puis continué en doctorat dans l'équipe TSF, pour avoir participé à la réalisation de mes travaux en donnant ses retours et ses réflexions. Il est notamment à l'origine des travaux présentés dans le dernier chapitre de cette thèse. Je remercie Louis Girard qui a effectué son stage de quatrième année d'ingénieur dans le cadre de ma thèse, ainsi que Guillaume Brau, pour ses nombreux retours et conseils sur la réalisation de ma thèse. Enfin, je suis reconnaissant envers Daniela Dragomirescu, qui m'a redirigé vers Vincent et Éric lorsque je cherchais à réaliser une thèse et donc sans qui cette réalisation n'aurait pas été possible.

Je souhaite exprimer ma profonde gratitude envers toute l'équipe TSF pour leur accueil chaleureux. Je n'oublierai jamais les rencontres, les moments partagés et les discussions que nous avons eues. Je tiens à remercier particulièrement tous ceux qui m'ont apporté leur soutien technique et moral.

Au cours de ces nombreuses années de thèse, j'ai eu l'occasion de rencontrer de nombreux doctorants et post-doctorants qui ont fait partie de l'équipe. Je vais me risquer à tous les cités en espérant n'oublier personne : Carla S., Benoît M., Roberto P., Kalou C., Rui W., Ulrich A., Julien D., Thierry S., William E., Lola M., Guillaume A., Christophe B., Matthieu A., Jonathan R., Aliénor D., Bilel C., Jean I., Malcolm B., Romain C., Yuxiao M., Cyrius N., Florent G., Mohamed E. M., Luca S., Raul S. F., Pierre-François G., Yassir I. M., Joris G., Mathieu E., Alexis G., Maria-Laura B. M. et Antony D. Je remercie les stagiaires et les autres personnes qui ont réalisé un séjour plus ou moins court dans l'équipe : Louis, Imanol, Quentin, Faical, Mathieu, Enzo, Thomas, Abbass, etc.

Je tiens à exprimer ma reconnaissance particulière envers mes deux compères,

Clément R. et Daniel L., qui m'ont accompagné pendant la majorité de ma thèse. Leur soutien indéfectible, leur aide précieuse et leur détermination m'ont permis de surmonter les moments les plus difficiles. Merci infiniment à vous deux !

Je suis reconnaissant envers mes amis qui m'ont soutenu tout au long de cette thèse, notamment Ibi A., dont le soutien a été précieux, Raphaël A., qui m'a permis de me changer les idées lorsque cela était nécessaire, ainsi que Estelle C. et Sokhna K. Je n'oublie pas la team Xdi Jeu avec Pierre-François G., Fabien M., Julien V., Estelle R., Céline M., Élise P., Victor D. et Mickaël L. J'ai également une pensée pour des amis plus ou moins éloignés qui auraient été fiers de voir mon parcours dont Kévin B., Charles A., Florent B., Thomas B., Benjamin A., Alexandre A., Alexandre D. et Alexandre G.

Ma gratitude va à ma famille, qui m'a soutenu tout au long de ces années. Je tiens particulièrement à remercier ma mère, qui a relu l'intégralité du manuscrit pour corriger les fautes et me donner des conseils de rédaction. Merci du fond du cœur pour votre soutien !

Enfin, je remercie ma compagne pour son soutien et son encouragement indéfectibles. Sans elle, je n'aurais jamais pu aller jusqu'au bout de cette aventure.

Ainsi se termine cette thèse après toutes ces années. Merci à tous d'avoir été présents et d'avoir partagé une partie de votre vie avec moi. Je présente mes excuses à ceux que j'oublie et je les remercie pour l'aide qu'ils m'ont apportée au cours de ces années. Merci infiniment pour tout !

Table des matières

Liste des figures

Liste des tableaux

Introduction

La connectivité des véhicules d'aujourd'hui entraîne de nouvelles problématiques de protection des données. Ces dernières années, l'architecture des véhicules a beaucoup évolué. De nombreux composants ont été ajoutés aux véhicules, qui sont maintenant équipés de nombreux capteurs et actionneurs, ainsi que d'équipements permettant la connectivité des véhicules. Ils sont maintenant considérés *connectés*. Le véhicule connecté peut être vu comme un capteur mobile, obtenant une importante quantité d'informations sur son environnement ou sur son état. L'utilisation de ces informations dépasse le cadre du bon fonctionnement du véhicule, elles peuvent être utiles à différentes entités : au conducteur du véhicule pour bénéficier de nouveaux services telle qu'une assurance basée sur l'utilisation ; au constructeur automobile pour évaluer la performance et la fiabilité de son parc automobile ; à d'autres parties prenantes telle qu'une entreprise météorologique afin d'améliorer leurs modèles météorologiques via l'exploitation des données de capteurs de température et d'humidité. Ainsi, les véhicules connectés sont conçus pour envoyer régulièrement leurs données à un cloud chargé du traitement et du partage des données.

L'externalisation des données des véhicules entraîne des problématiques de sécurité et de vie privée. Les clouds sont régulièrement victimes d'attaques entraînant une fuite des données. À notre connaissance cela ne s'est pas encore produit pour les données de véhicules connectés mais le risque est présent. Par ailleurs, le conducteur n'a généralement que peu ou pas de marge de manœuvre sur ses données, il ne peut décider de qui a le droit d'y accéder. Or la mise en application du RGPD donne certains droits aux conducteurs sur leurs données personnelles. Enfin, la législation est également à prendre en compte. Certains articles de loi, en cours d'élaboration, visent à autoriser ou interdire l'accès aux données à certaines parties prenantes, ces règles doivent être prises en compte lorsqu'une partie prenante souhaite accéder aux données. Il est ainsi important de mettre en place des mécanismes permettant de faire face à ces problématiques.

Les mécanismes de contrôle d'accès classiques nous semblent limités car ils donnent trop de pouvoir au cloud sur le contrôle de l'accès aux données et sur la possibilité d'accéder aux données en clair. En effet, puisque les données sont centralisées sur un cloud, les différentes parties prenantes vont réaliser des requêtes à ce cloud pour accéder à leurs données. Il est donc fondamental que ce cloud implémente correctement la politique d'autorisation prévue et donne aux différentes entités l'accès uniquement aux données pour lesquelles elles sont autorisées. Si par ailleurs, les données sont remontées en clair sur le cloud, ce dernier peut également consulter ces données, même s'il est simplement honnête mais curieux. Une approche qui nous semble plus adaptée est d'utiliser les mécanismes de chiffrement permettant aux différentes parties prenantes d'accéder à leurs données car elles SEULES sont capables de les déchiffrer. Le cloud stocke les données mais il est incapable de les déchiffrer. Les mécanismes de chiffrement que nous souhaitons implanter sont

exécutés en partie dans le véhicule pour protéger les données à la sortie du véhicule. Mais le véhicule est un système très contraint, il possède des ressources en calcul, en stockage et en bande passante limitées, et il doit réaliser des opérations en temps réel. Les mécanismes à implémenter doivent respecter ces contraintes. Ainsi, les mécanismes de chiffrement classiques ne sont pas adaptés car ils nécessiteraient soit un nombre important de clés à stocker, soit plusieurs chiffrements pour une même donnée, soit ils manquent en flexibilité. En revanche, le *chiffrement basé attributs* (en anglais *ABE : Attribute-Based Encryption*), dans lequel le chiffrement "embarque" le contrôle d'accès, semble être une approche plus adaptée pour répondre à ce problème.

Par ailleurs, il est fondamental que la politique d'autorisation soit en conformité avec la loi. Comme dit plus haut dans cette introduction, diverses articles de lois sont en cours d'élaboration sur ce sujet et imposent certaines règles strictes sur la politique d'autorisation à appliquer pour différentes données produites par les véhicules connectés. Aujourd'hui, il n'existe pas à notre connaissance de méthode permettant de générer les règles de contrôles d'accès (qui seront représentées dans le cas d'ABE par des attributs et des arbres d'accès). Nous souhaitons donc proposer une telle méthode dans ce manuscrit.

Cette thèse est menée dans le cadre d'un partenariat entre Continental Digital Services France (CDSF) et le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS) en relation avec le projet e-Horizon réalisé par CDSF. Ce projet s'inscrit dans le contexte précédemment décrit, afin d'externaliser les données des véhicules et de proposer des services exploitant les données des véhicules dans le cloud. Cette thèse présente nos réflexions autour de la protection des données des véhicules connectés en considérant trois thématiques :
1. la protection des données des véhicules via des mécanismes cryptographiques ;
2. la prise en compte du conducteur et de la législation dans la mise en place du contrôle d'accès ;
3. l'optimisation des performances des mécanismes cryptographiques.

Le premier chapitre présente le contexte général de nos travaux. Nous y présentons 1) le contexte du véhicule connecté et l'envoi des données ; 2) un panorama des attaques informatiques touchant les véhicules ; 3) les menaces à la vie privée dans les véhicules connectés ; et 4) les contraintes de la solution et les approches classiques pour protéger les données.

Le second chapitre présente plus précisément le contexte de nos travaux et les menaces associées. Nous y présentons les parties prenantes considérées, le modèle d'attaquant ainsi que les différentes propriétés et exigences que nous souhaitons satisfaire. Puis nous présentons un aperçu des travaux relatifs aux trois axes explorés dans cette thèse.

Les chapitres suivants présentent les travaux réalisés dans cette thèse autour des trois thèmes précédemment évoqués. Le troisième chapitre présente notre première contribution : un protocole cryptographique de remontée de données des véhicules connectés. Il repose sur l'utilisation du chiffrement basé attributs pour mettre en place le contrôle d'accès, du chiffrement symétrique pour protéger la confidentialité

des messages, et de la signature de groupe pour vérifier l'authenticité et l'intégrité des messages. Nous commençons par présenter les schémas cryptographiques utilisés, puis nous présentons les scénarios du protocole. Ensuite nous présentons ProVerif l'outil utilisé pour vérifier automatiquement les propriétés de ce protocole. Enfin, nous terminons par la vérification des propriétés.

Le quatrième chapitre présente notre deuxième contribution : une méthode permettant de générer le matériel cryptographique utilisé lors du chiffrement basé attributs. Cette méthode prend en compte la législation et les choix du conducteur. Elle repose sur une étape de traduction de la législation dans une politique de contrôle d'accès de type OrBAC avant de générer les clés et attributs nécessaires à la mise en place du contrôle d'accès. Nous évaluons cette méthode vis-à-vis des exigences précédemment définies et nous l'illustrons via un cas d'étude.

Le cinquième chapitre présente une évaluation des performances du protocole. L'évaluation est réalisée en considérant le temps de calcul et la taille en mémoire des étapes du protocole sur une Raspberry Pi, ainsi que la taille des messages échangés. Nous commençons par présenter l'implémentation du protocole cryptographique en langage C, puis le protocole expérimental et enfin les résultats des mesures.

Le sixième chapitre présente notre troisième contribution concernant des optimisations du chiffrement basé attributs. Ces optimisations reposent sur l'utilisation de pré-calculs et sur les propriétés des opérations permettant de ré-écrire les opérations sans en changer le résultat. Nous commençons par présenter des rappels techniques nécessaires à la compréhension des optimisations, puis nous présentons les optimisations individuellement et leurs applications sur les schémas cryptographiques considérés, enfin nous évaluons le temps de calcul des optimisations et le coût de stockage des pré-calculs.

Le contenu des chapitres deux à cinq est une version améliorée de nos travaux, initialement évalué par les pairs et publié en anglais dans le *Journal of Computer Virology and Hacking Techniques* le 5 Mai 2022 [Adelin 2022].

Chapitre 1

Contexte général

Sommaire

Ce premier chapitre est consacré à la présentation du contexte général de notre étude et aux problématiques que nous abordons dans cette thèse. Nous commençons ce chapitre par la présentation de notre contexte à savoir le véhicule connecté, son environnement et l'externalisation de ses données à destination d'un cloud. Notre sujet étant clairement focalisé sur la thématique globale de la sécurité informatique, nous introduisons la terminologie de la sûreté de fonctionnement, qui englobe la sécurité informatique. Nous faisons également un rapide état des lieux des principales attaques qui ont récemment visé les véhicules, montrant ainsi l'importance aujourd'hui de s'intéresser à la sécurité de ces véhicules. Nous montrons en quoi l'ajout de connectivité aux véhicules et l'externalisation de ses données soulèvent des questions de sécurité mais aussi de protection de la vie privée. Nous spécifions les contraintes que la réponse à ces problèmes doit satisfaire pour être

acceptable. Nous dressons ensuite un aperçu des principales approches permettant de traiter les problématiques de vie privée. Enfin, nous terminons en concluant ce chapitre.

1.1 Véhicule connecté et son environnement

1.1.1 Évolution des architectures des véhicules

Au fur et à mesure des avancées de l'automobile et de l'évolution des besoins de la société, les objectifs des constructeurs automobiles et les architectures des véhicules se sont adaptés à ces évolutions. Les premiers véhicules étaient totalement mécaniques, ils ont été conçus afin d'améliorer la rapidité et la facilité de déplacement. Par la suite, les premières années de l'automobile ont été marquées par une recherche de l'augmentation de la vitesse et de l'amélioration de la résistance des véhicules. Avec le développement de véhicules plus rapides, le nombre d'accidents a également augmenté et les véhicules ont dû évoluer afin d'être plus sûrs. Puis les objectifs des constructeurs se sont diversifiés, ils ont cherché à améliorer le rendement du moteur, à diminuer la consommation de carburant, à rendre le véhicule plus maniable, à réduire la pollution, etc. L'électronique a commencé à être intégrée aux véhicules afin de permettre d'avantage de contrôle sur ses composants et d'ajouter de nouveaux composants via par exemple une injection électronique, un système antiblocage des roues électronique ou encore un système de surveillance de pression des pneus. Ce contrôle a été facilité par l'ajout de systèmes embarqués, les *Unités de Commande Électrique (UCE)* permettant de récupérer des informations des capteurs du véhicule et de réaliser des actions via des actionneurs du véhicule. Dès lors, le nombre d'UCE dans les véhicules n'a cessé de croître, les fonctionnalités proposées se sont complexifiées et ont requis une interconnexion des UCEs afin de faciliter leur communication. Ainsi, les véhicules ont été équipés de réseaux permettant cette communication, parmi ces réseaux le Control Area Network (CAN) est le plus utilisé actuellement. L'ajout d'un réseau dans le véhicule a également permis de faciliter le diagnostic du véhicule en cas de panne via l'ajout de systèmes de diagnostic embarqués (OBD : On Board Diagnosis) permettant de se connecter au réseau du véhicule afin de récupérer des informations sur l'état du véhicule et ses composants. Enfin, ces dernières années ont vu le développement de véhicules équipés de dispositifs leur permettant de communiquer avec des équipements extérieurs aux véhicules, et de véhicules équipés de systèmes de conduite semi-autonomes permettant de réaliser certaines portions du trajet de manière autonome. L'avenir sera certainement focalisé sur l'amélioration de l'automatisation de la conduite et de sa sûreté ainsi que sur l'ajout de connectivité dans les véhicules. Dans cette thèse, nous nous intéressons au véhicule équipé de dispositifs de communication, souvent appelé *véhicule connecté*, ainsi qu'à son environnement.

1.1.2 Véhicule connecté

Un véhicule connecté est un véhicule qui peut communiquer avec des équipements qui lui sont extérieurs ou qui ne font pas partie du véhicule directement. Les dispositifs de communication dans un véhicule peuvent être nombreux, ils varient notamment selon leur portée. Un véhicule équipé de dispositifs de communication de très courte portée (e.g. Bluetooth) peut réaliser des communications de type véhicule-à-appareil (v2d : vehicle-to-device) avec des équipements à l'intérieur du véhicule ou dans son environnement très proche tel qu'un téléphone intelligent ou un objet connecté pour, par exemple, que le conducteur puisse réaliser des appels en utilisant son véhicule. Un véhicule équipé de dispositifs de communication de courte portée (e.g. IEEE 802.11p) peut réaliser des communications de type véhicule-à-véhicule (v2v : vehicle-to-vehicle), véhicule-à-infrastructure (v2i : vehicle-to-infrastructure) ou véhicule-à-piéton (v2p : vehicle-to-pedestrian). Les communications véhicule-à-véhicule permettent au véhicule de communiquer avec un autre véhicule afin, par exemple, de le prévenir d'un accident ayant eu lien dans son environnement proche ou d'un freinage d'urgence. Les communications véhicule-à-infrastructure permettent au véhicule de communiquer avec l'infrastructure routière (e.g. feux de signalisation connectés, panneaux de signalisation connectés) afin, par exemple, de récupérer d'un parking intelligent la localisation d'une place de parking disponible. Les communications véhicule-à-piéton permettent au véhicule de communiquer avec un équipement d'un piéton afin, par exemple, de prévenir le véhicule de la présence du piéton lors d'une situation pouvant être dangereuse (e.g. manque de visibilité lorsque le piéton traverse la route). Un véhicule équipé de dispositifs de communication longue portée (e.g. LTE) peut réaliser des communications de type véhicule-à-réseau (v2n : vehicle-to-network) permettant au véhicule de communiquer sur les réseaux mobiles et ainsi d'accéder à Internet, il peut alors accéder à des services tels qu'un service permettant de récupérer une carte mise à jour avec des événements (e.g. l'état du trafic, les accidents sur la route) en direct. L'ensemble des destinataires du véhicule est regroupé sous la mention véhicule-à-tout (v2x : vehicle-to-everything).

Comme illustré en Figure 1.1, le véhicule d'aujourd'hui est un système informatique complexe. Outre les dispositifs de communications, il est équipé de capteurs collectant de nombreuses données sur son état (e.g. capteurs de vitesse, de température du moteur, de pression des pneus, de position angulaire, de géolocalisation) ou des informations sur son environnement (e.g. capteurs de proximité, de température extérieur, caméras). Il est aussi équipé d'actionneurs réalisant des actions sur les composants du véhicule (e.g. moteur, injecteur, frein). Les capteurs et actionneurs sont reliés à des UCEs, elles-mêmes reliées entre elles via un réseau interne au véhicule (e.g. CAN). Le véhicule est aussi équipé de passerelles (gateway) permettant de relier le réseau interne du véhicule aux réseaux externes. Enfin, le véhicule peut être équipé d'un système de positionnement (e.g. GPS) lui permettant de se géolocaliser.

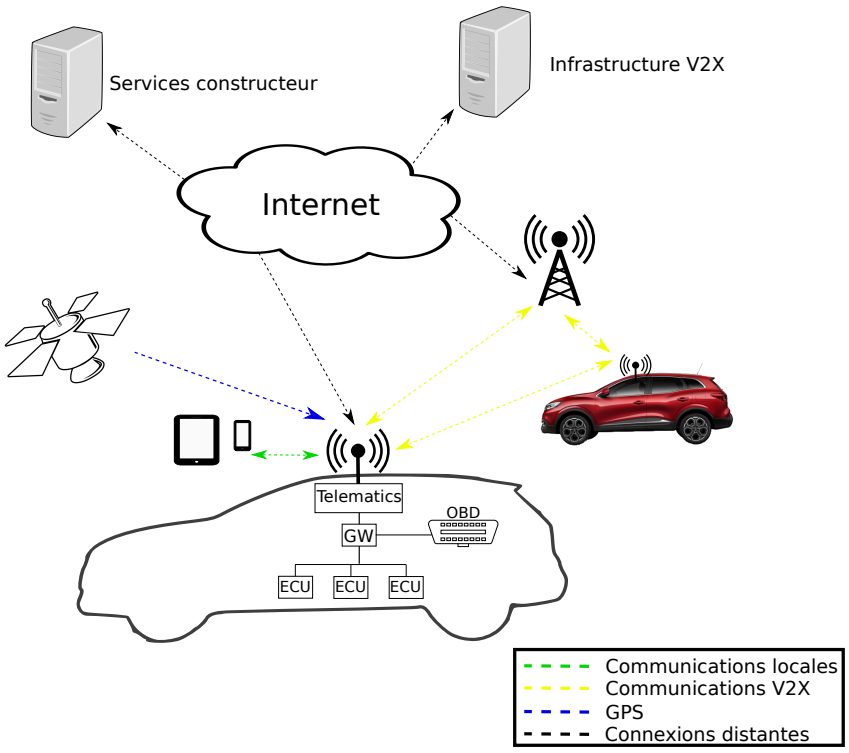

FIGURE 1.1 – Architecture du véhicule (issue de [Studnia 2015])

1.1.3 Données du véhicule connecté

Les données produites par les capteurs du véhicule peuvent permettre au conducteur de bénéficier de nouveaux services. Par exemple, certaines assurances proposent des services de type *pay-as-you-drive* ajustant de manière plus fine la prime d'assurance du conducteur en fonction de son comportement sur la route. Ce comportement peut être évalué via le recoupement de certaines données telles que la vitesse, la position, l'accélération et la force de freinage du véhicule. Cependant, le véhicule connecté étant devenu une véritable source mobile de données, ces données intéressent également d'autres parties prenantes pour leurs bénéfices personnels. Par exemple, les données des capteurs de température et d'humidité du véhicule peuvent intéresser un service météorologique pour améliorer la précision de ses modèles météorologiques et donc de ses prédictions.

Pour faciliter le partage et l'exploitation de ces données, les véhicules connectés sont de plus en plus conçus pour externaliser leurs données vers une plateforme chargée du stockage, du traitement et du partage des données. Cette plateforme est souvent un cloud opéré par le constructeur du véhicule, on parlera alors de communication véhicule-à-cloud (v2c : vehicle-to-cloud). Dans cette thèse, nous nous focalisons sur les communications véhicule-à-cloud dans lesquelles les véhicules connectés stockent leurs données dans un cloud, les conducteurs veulent accéder à leurs données hébergées dans le cloud et différentes parties prenantes veulent accéder

aux données des conducteurs.

Cependant, l'externalisation de ces données entraîne des problèmes de vie privée et une perte de contrôle de l'accès à ces données par les conducteurs. Ainsi, de nombreux travaux de recherche s'intéressent aujourd'hui aux problèmes de sécurité qui peuvent affecter les véhicules connectés, dans leur déploiement actuel mais aussi en tenant compte des futures évolutions qui vont accompagner la production des véhicules. Nos travaux se situent également dans ce contexte et la section suivante propose donc de donner un aperçu de la terminologie de la sûreté de fonctionnement en général, qui englobe la sécurité informatique.

1.2 Terminologie de la sécurité informatique

Le terme de sécurité est ambigu, il peut désigner aussi bien la sécurité-innocuité (*safety*) que la sécurité-confidentialité (*security*), qui sont deux concepts relevant de la sûreté de fonctionnement. Nos travaux s'inscrivent dans le cadre de la sécurité-confidentialité. Dans un premier temps, nous décrivons les concepts généraux de la sûreté de fonctionnement, introduits dans [Laprie 1996] et mis à jour dans [Avizienis 2004], puis nous nous concentrons ensuite sur l'application de ces concepts au domaine de la sécurité-confidentialité.

1.2.1 Sûreté de fonctionnement

La **sûreté de fonctionnement** d'un système informatique est définie comme « la propriété qui permet aux utilisateurs du système de placer une confiance justifiée dans le service qu'il leur délivre ». Le service délivré par un système est son comportement tel que perçu par son (ou ses) utilisateur(s), l'utilisateur étant un autre système (humain ou physique) qui interagit avec le système considéré. La non-sûreté de fonctionnement survient lorsque la confiance ne peut plus, ou ne pourra plus, être placée dans le service délivré. La sûreté de fonctionnement comporte trois axes principaux : les *attributs* qui la décrivent, les *entraves* qui empêchent sa réalisation et les *moyens* d'atteindre celle-ci (voir Figure 1.2).

La sûreté de fonctionnement peut être perçue selon les propriétés suivantes, appelées *attributs* de la sûreté de fonctionnement :

- disponibilité : aptitude du système à être prêt à l'utilisation ;
- fiabilité : continuité du service ;
- sécurité-innocuité : absence de conséquences catastrophiques pour l'environnement ;
- confidentialité : absence de divulgations non autorisées de l'information ;
- intégrité : absence d'altérations inappropriées de l'information ;
- maintenabilité : aptitude aux réparations et aux évolutions.

Ainsi, ces attributs permettent d'une part d'exprimer les propriétés devant être respectées par le système et d'autre part d'évaluer la qualité du service délivré

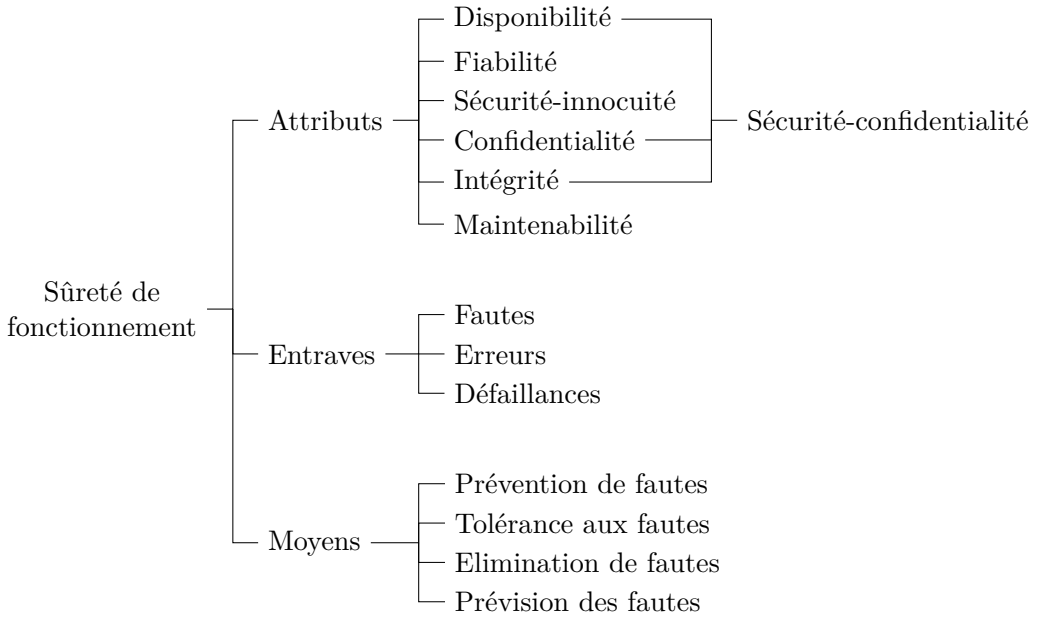

FIGURE 1.2 – Arbre de la sûreté de fonctionnement (issu de [Laprie 1996])

vis-à-vis de ces propriétés. Les attributs à considérer dépendent des applications auxquelles le système est destiné.

Ces attributs peuvent être mis à mal par des *entraves*. Une entrave est une circonstance indésirable mais non-inattendue. Elle est la cause ou le résultat de la non-sûreté de fonctionnement. On distingue trois types d'entraves :

— défaillance : survient lorsque le service délivré dévie de l'accomplissement de la fonction du système ;

— erreur : la partie de l'état du système susceptible d'entraîner une défaillance ;

— faute : la cause adjugée ou supposée d'une erreur.

Une faute est dite active lorsqu'elle produit une erreur. Par propagation, une erreur crée de nouvelles erreurs. Une défaillance survient lorsque, par propagation, elle affecte le service délivré par le système. Cette défaillance peut alors apparaître comme une faute du point de vue d'un autre composant. L'enchaînement de ces entraves crée ainsi la chaîne fondamentale suivante :

$$\ldots \rightarrow \text{défaillance} \rightarrow \text{faute} \rightarrow \text{erreur} \rightarrow \text{défaillance} \rightarrow \ldots$$

Enfin, les *moyens* de la sûreté de fonctionnement permettent de minimiser l'impact des entraves sur les attributs. Ce sont les méthodes et techniques permettant de fournir au système l'aptitude à délivrer un service conforme à l'accomplissement de sa fonction, et de donner confiance dans cette aptitude. On distingue quatre catégories en fonction de l'objectif visé :

— prévention des fautes : empêcher l'occurrence ou l'introduction de fautes ;

— tolérance aux fautes : fournir un service qui remplit la fonction du système en dépit des fautes ;

— élimination des fautes : réduire le nombre et la sévérité des fautes ;

— prévision des fautes : estimer la présence, la création et la conséquence des fautes.

Nos travaux s'inscrivent dans le cadre de la sécurité-confidentialité, qui se définit comme une combinaison de la disponibilité, la confidentialité et l'intégrité. Le reste de cette section est dédié à la définition des concepts relatifs à la sécurité-confidentialité.

1.2.2 Sécurité-confidentialité

La **sécurité-confidentialité** vise à protéger un système contre les fautes malveillantes (ou *malveillances*) pouvant survenir lors de la conception ou de l'exploitation de celui-ci. Les attributs de la sûreté de fonctionnement permettent de différencier les termes de sécurité-innocuité et de sécurité-confidentialité. La sécurité-innocuité définit les propriétés selon lesquelles un système est dit « sûr »(sans défaillance catastrophique pouvant conduire à des pertes de vies humaines ou des conséquences économiques importantes). La sécurité-confidentialité définit les propriétés selon lesquelles un système est dit « sécurisé »(protégé contre les fautes intentionnelles). Les concepts de sûreté de fonctionnement présentés précédemment sont génériques afin de pouvoir couvrir un grand nombre de concepts. Nous allons maintenant nous intéresser à leur application dans le contexte plus spécifique de la sécurité-confidentialité. Dans le reste de ce manuscrit, le terme *sécurité* employé dans l'expression *sécurité informatique* est à considérer au sens de la sécurité-confidentialité, sauf si explicitement précisé. Les définitions qui suivent proviennent de la terminologie des malveillances introduite lors du projet Malicious- and Accidental- Fault Tolerance for Internet Applications [MAFTIA 2003].

Considérés du point de vue de la sécurité-confidentialité, les attributs de la sûreté de fonctionnement que l'on cherche à garantir peuvent être définis ainsi :

— confidentialité : prévention de toute divulgation non autorisée d'information ;

— intégrité : prévention de toute modification non autorisée d'information ;

— disponibilité : prévention de toute rétention non autorisée d'information.

Les fautes malveillantes affectant les logiciels se répartissent en deux catégories : les logiques malignes et les intrusions. Une logique maligne est une partie du système conçue pour provoquer des dégâts (bombe logique) ou pour faciliter des intrusions futures (vulnérabilités créées volontairement). Elle englobe les failles de développement telles que les chevaux de Troie, les bombes logiques ou temporelles et les trappes, ainsi que les failles opérationnelles (par rapport au système donné) telles que les virus ou les vers. Une intrusion est définie grâce aux concepts d'attaque et de vulnérabilité :

— attaque : une faute d'interaction malveillante, par laquelle un attaquant vise à délibérément violer une ou plusieurs propriétés de sécurité ;

— vulnérabilité : une faute créée pendant le développement du système, ou pendant son fonctionnement, qui pourrait être exploitée pour créer une intrusion ;

— intrusion : une faute malveillante, d'origine externe, résultant d'une attaque qui a réussi à exploiter une vulnérabilité.

Les attaques, vulnérabilités et intrusions étant des fautes, les moyens de la sûreté de fonctionnement peuvent être appliqués pour améliorer la sécurité-confidentialité d'un système.

Prévention des fautes La prévention des fautes a lieu au début du cycle de vie du système et vise à empêcher l'occurrence ou l'introduction de fautes (donc dans le cas de la sécurité-confidentialité, d'attaques, de vulnérabilités et d'intrusions). Concernant la sécurité-confidentialité, les techniques relevant de la prévention des fautes incluent l'introduction de mécanismes tels que l'authentification, l'autorisation et les pare-feux, qui empêchent les attaques dans la mesure où ils repoussent les attaques au niveau des barrières supplémentaires que ces mécanismes introduisent, des mesures allant de la spécification semi-formelle et formelle, de la conception rigoureuse et des procédures de gestion du système, jusqu'à l'éducation des utilisateurs (par exemple, le choix des mots de passe).

Tolérance aux fautes La tolérance aux fautes est mise en œuvre en deux étapes : la *détection d'erreur*, qui permet de détecter un état erroné du système et le *rétablissement* dont le but est de ramener le système dans un état de confiance. Concernant la sécurité-confidentialité, un système tolérant aux fautes est défini comme un système capable de s'auto-diagnostiquer, se réparer et se reconfigurer tout en continuant à fournir un service acceptable aux utilisateurs légitimes pendant une attaque [Deswarte 1991]. Parmi ces mécanismes, les méthodes de *détection d'intrusions* concernent l'ensemble des pratiques et des mécanismes utilisés pour détecter les erreurs pouvant conduire à une défaillance de la sécurité, et/ou diagnostiquer les attaques. Parmi les autres mécanismes courants de la tolérance aux fautes nous pouvons également citer le *chiffrement*.

Élimination des fautes Il est difficile de développer correctement un système, ce qui fait qu'en dépit de l'application de techniques de prévention des fautes, certaines fautes peuvent persister, et peuvent alors être exploitées à des fins malveillantes. L'élimination des fautes cherche à en réduire le nombre, en s'appuyant sur des contre-mesures humaines visant directement l'attaquant, d'actions de maintenance visant à supprimer les logiques malveillantes agissant ou pouvant agir comme des agents d'attaque. Pendant le développement du système, il s'agit de procédures de vérification telles que la preuve formelle, la vérification de modèles et les tests, visant spécifiquement à identifier les failles qui pourraient être exploitées par un attaquant. Pendant l'exploitation du système, cela correspond à des actions de maintenance préventive et corrective telles que l'application d'un patch de sécurité, le retrait

d'un service donné, le changement de mot de passe, la suppression d'une logique malveillante mettant en œuvre une trappe, etc.

Prévision des fautes Enfin, la prévision des fautes a pour but d'identifier les vulnérabilités, attaques, intrusions et modes de défaillances potentiels du système et d'analyser leur impact sur son comportement, vis-à-vis des propriétés attendues. Les techniques de prévision des fautes comprennent la collecte de renseignements, l'évaluation des menaces et l'alerte en cas d'attaque, l'évaluation du nombre actuel d'agents d'attaque latents, ainsi que de l'incidence future et des conséquences probables de leur activation, la collecte de statistiques sur l'état actuel des connaissances concernant les failles du système, et les difficultés qu'un attaquant aurait pour en tirer parti.

1.3 Panorama des attaques

L'évolution des architectures des véhicules a entraîné une augmentation des attaques pouvant être réalisées sur celles-ci. Les premières menaces informatiques sont arrivées suite à l'ajout de systèmes embarqués dans les véhicules et se sont développées avec l'augmentation du nombre d'UCE et la mise en place de réseaux dans les véhicules. Certaines attaques réalisées ont eu pour but de prendre contrôle de tout ou partie du véhicule, d'autres attaques ont eu pour but de récupérer des informations concernant le véhicule et son conducteur. Cependant, le véhicule était autrefois un système fermé, les attaques réalisables nécessitaient un accès physique au véhicule par l'attaquant. Le véhicule d'aujourd'hui est équipé de dispositifs de communication, il est devenu un système ouvert. Cette ouverture a pour contrepartie d'augmenter sa surface d'attaque, de nouvelles attaques sont réalisables à distance. Dans cette section, nous présentons une classification des attaques ciblant les calculateurs embarqués d'un véhicule, cette classification est issue des travaux d'Ivan Studnia [Studnia 2015], nous présentons également des exemples d'attaques permettant d'illustrer cette classification.

1.3.1 Classification des attaques

La classification est représentée dans la Figure 1.3 et est détaillée dans ce qui suit. Elle est basée sur quatre caractéristiques : l'interaction avec le système, la portée, le lien et l'impact de l'attaque.

L'interaction avec le système représente le type de composant du système cible utilisé par l'attaquant pour réaliser l'attaque. L'interaction peut être au niveau matériel telle qu'une attaque physique sur le matériel directement ou un signal envoyé aux capteurs du véhicule. L'interaction peut également être au niveau logiciel lorsque l'attaque exploite une vulnérabilité dans la pile logiciel d'une UCE telle qu'une vulnérabilité dans une libraire de l'UCE ou dans l'implémentation de son protocole de communication. Enfin l'interaction peut être au niveau du réseau du véhicule, une telle attaque va chercher à lire des informations sur le réseau, empêcher

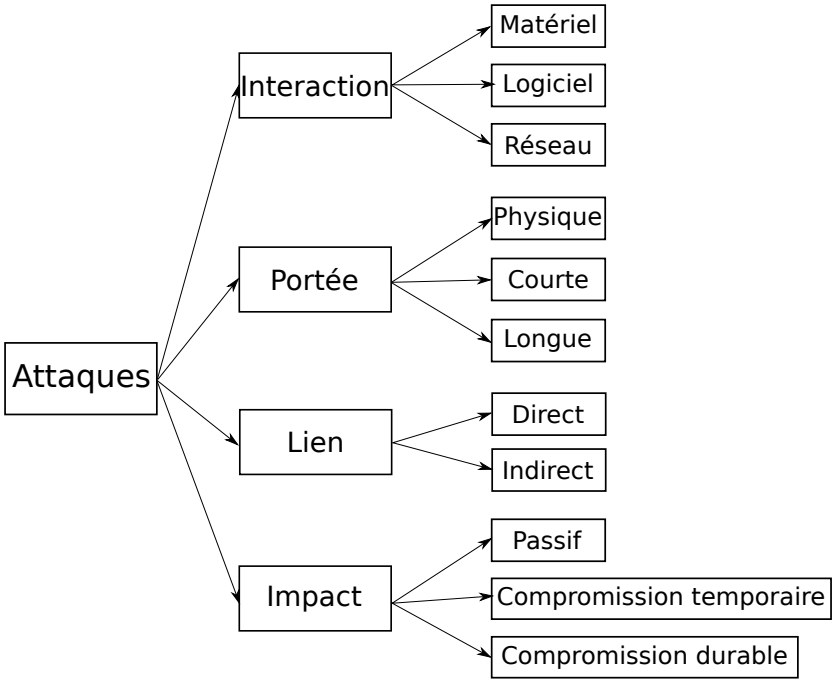

FIGURE 1.3 – Classification des attaques ciblant les calculateurs embarqués dans une automobile (issue de [Studnia 2015])

la réception d'un message, modifier un message émis avant sa réception, rejouer un message émis ou fabriquer un message.

La portée représente le type de distance requis entre l'attaquant et la cible pour réaliser l'attaque. La portée peut être au niveau physique lorsque l'attaquant nécessite une connexion physique directe au véhicule. L'attaque peut également être réalisée à distance, certaines attaques peuvent être réalisées à de faibles portées via des moyens de communication sans fil de l'ordre de quelques mètres tels que le Bluetooth. D'autres attaques peuvent être réalisées à de longues portées via des moyens de communication sans fil de plusieurs mètres voire kilomètres tels que les réseaux de téléphonie mobile.

Le lien désigne la présence ou l'absence d'un intermédiaire entre l'attaquant et la cible pour réaliser l'attaque. Le lien est direct si l'un des composants du véhicule est ciblé directement. Par exemple, une attaque ciblant les interfaces du véhicule ou son réseau. Le lien est indirect si l'attaquant compromet un élément extérieur au véhicule qui interagira avec le véhicule pour réaliser l'attaque. Par exemple, une clé USB compromise qui, lorsqu'elle est branchée sur le véhicule, réalise une attaque.

L'impact représente le type de modifications de comportement entraînées par l'attaquant sur la cible pour réaliser l'attaque. L'impact est considéré passif lorsque l'attaque permet de collecter des informations sur la cible sans entraîner de modification du comportement du véhicule. L'impact est une compromission temporaire lorsque la modification du comportement du véhicule est limitée à la durée de l'attaque, si l'attaquant souhaite réaliser une autre attaque il devra recommen-

FIGURE 1.4 – Tableau de bord affichant une vitesse erronée ainsi qu'un message personnalisé (issu de [Koscher 2010])

cer l'attaque intégralement. L'impact est une compromission durable lorsque la modification du comportement du véhicule persiste dans le temps. Un tel impact peut survenir par exemple si l'attaque reprogramme une UCE en modifiant son comportement durablement.

1.3.2 Exemples d'attaques

Pour illustrer cette classification, quelques exemples d'attaques ayant touché les véhicules sont présentés dans ce qui suit.

Attaque réseau-physique-direct-temporaire : Koscher et al. [Koscher 2010] décrivent une attaque permettant de changer la valeur de la vitesse affichée sur le tableau de bord via l'envoi de trames contenant une vitesse erronée, la Figure 1.4 illustre cette attaque.

Attaque logiciel-longue portée-indirecte-durable : Checkoway et al. [Checkoway 2011] décrivent une attaque qui, à partir d'un appel réalisé avec un véhicule, exploite plusieurs vulnérabilités pour forcer le module télématique à télécharger et exécuter un logiciel via le réseau 3G.

Attaques logiciel-longue portée-indirecte-temporaire, décrites également dans [Checkoway 2011] : la première consiste à utiliser un téléphone intelligent possédant un programme malveillant, téléchargé par le propriétaire du téléphone à son insu, qui lorsqu'il est appairé en Bluetooth avec l'UCE télématique du véhicule, prend le contrôle de cette UCE en exploitant un dépassement de tampon dans l'implémentation du Bluetooth de l'UCE. La deuxième attaque consiste à utiliser un CD contenant un fichier audio malveillant, récupéré par le conducteur à son insu, qui lorsqu'il est joué par le système multimédia du véhicule, entraîne l'émission d'une trame sur le réseau du véhicule.

Toutes ces attaques visent globalement à porter atteinte au fonctionnement nominal du véhicule. D'autres types de menaces nous semblent également fondamentales aujourd'hui, à savoir les menaces à la vie privée des conducteurs et à leur contrôle

sur les données qui sont légitimement émises par le véhicule. Nous présentons plus en détails dans la section suivante en quoi ces menaces consistent et quelles peuvent en être leurs conséquences.

1.4 Menaces à la vie privée dans les véhicules connectés

1.4.1 Définitions

Pour présenter les menaces vis-à-vis de la vie privée, nous commençons par définir les notions de données personnelles et de données sensibles. La définition de données personnelles est une adaptation de la définition donnée dans l'article 4 du *Règlement Général sur la Protection des Données (RGPD)* et la définition de données sensibles est extraite de l'article 9 du RGPD [RGPD 2018]. Une **donnée personnelle** est une information relative à une personne physique (e.g. nom, prénom, date de naissance, numéro de téléphone, numéro de sécurité sociale) identifiée ou identifiable. L'identification d'une personne physique peut se faire soit directement (e.g. via son nom et son prénom), soit indirectement par croisement d'information (e.g. une personne vivant à une adresse donnée, née à un jour donné et travaillant dans une entreprise donnée). Une **donnée sensible** est une donnée personnelle qui révèle l'origine raciale ou ethnique, les opinions politiques, les convictions religieuses ou philosophiques ou l'appartenance syndicale, ainsi que le traitement des données génétiques, des données biométriques aux fins d'identifier une personne physique de manière unique, des données concernant la santé ou des données concernant la vie sexuelle ou l'orientation sexuelle d'une personne physique.

1.4.2 Menaces liées à l'utilisation des données personnelles

Plusieurs exemples de menaces liées à l'utilisation des données personnelles sont présentés dans ce qui suit.

L'atteinte à un individu. Par exemple, en connaissant l'adresse du domicile d'une personne, un individu malintentionné peut se rendre à cette adresse lorsque la personne est absente pour dégrader son logement, voler ses effets personnels, ou lorsque la personne est présente pour porter atteinte physiquement à la personne, exercer des pressions, des menaces, etc.

La réalisation de profils d'utilisateurs et la publicité ciblée. Certains sites web peuvent établir des profils de leurs utilisateurs à partir de leur comportement sur le site, par exemple, à partir des recherches et des liens visités sur un moteur de recherche, à partir des articles consultés et achetés sur un site marchand ou encore à partir des recherches et des vidéos visionnées sur un site d'hébergement de vidéos. Ces profils définissent un ensemble de données personnelles pouvant parfois contenir des données sensibles. Les profils récoltés peuvent être vendus à des agences publicitaires qui vont diffuser de la publicité adaptée à l'utilisateur sur les sites correspondants pour le pousser à l'achat.

L'influence de la population dans leurs prises de décisions. Cette menace fait référence au scandale Facebook-Cambridge Analytica [TheGuardian 2018, LeMonde 2018, TheNewYorkTimes 2018] dans lequel la société anglaise Cambridge Analytica a collecté une quantité importante de données personnelles d'utilisateurs du réseau social Facebook, principalement américain et britannique. Ces données ont été utilisées pour établir des profils d'utilisateurs puis pour envoyer des messages ciblés aux utilisateurs afin d'influencer leurs intentions de vote ou de provoquer l'abstention. Cette influence a eu notamment pour but de favoriser le candidat Donald Trump lors de l'élection présidentielle américaine et de favoriser le retrait du Royaume-Uni de l'Union Européenne lors du référendum sur l'appartenance du Royaume-Uni à l'Union Européenne, tous deux en 2016. Ces affaires ont été rendues publiques en 2018 par un ancien directeur de recherche de Cambridge Analytica. Bien qu'il soit difficile de connaître toute la vérité et d'évaluer le réel impact de cette société sur les votes, il est néanmoins possible qu'une telle influence soit réalisable.

Les menaces aux libertés individuelles. En France, le projet Système Automatisé pour les Fichiers Administratifs et le Répertoire des Individus (SAFARI), démarré sous la présidence de Georges Pompidou, avait pour but de créer un identifiant unique pour chaque Français afin de rassembler tous les fichiers administratifs existants dont ceux du ministère de l'intérieur. Des ingénieurs en informatique du ministère de l'intérieur, inquiets des possibles menaces aux libertés individuelles qu'entraînerait l'exploitation de tels fichiers, ont contacté le journal Le Monde pour les avertir de ce projet. Le journal a par la suite publié un article à ce sujet le 21 mars 1974 [LeMonde 1974], entraînant une forte opposition au projet de la part de la population. Parmi les questions qui se posèrent à l'époque, l'une d'entre elles était la possible utilisation de ce fichier par un gouvernement autoritaire, suite à une crise économique ou politique. Le projet a été par la suite abandonné et cette affaire a grandement influencé la création de la *Commission Nationale de l'Informatique et des Libertés (CNIL)* et de la loi du 6 janvier 1978, loi dite Informatique et Libertés [Commission Informatique et Libertés 1974].

Les menaces à la population. Sous le gouvernement de Vichy, des fichiers ont été élaborés afin de recenser la population de confession juive en France [Poznanski 1997]. Ces fichiers ont notamment été utilisés lors de la majorité des rafles et des arrestation individuelles.

1.4.3 Législation et données personnelles

La récolte de données personnelles est encadrée en France par le Règlement Général sur la Protection des Données qui est un règlement européen. Dans ce qui suit, une définition du terme règlement européen est présentée suivie d'une définition du RGPD. Selon la définition de la Commission Européenne [1], les **règlements européens** sont *des actes législatifs qui s'appliquent, dès leur entrée en vigueur, de manière automatique et uniforme dans tous les pays de l'Union Européenne, sans devoir être transposés dans la législation nationale. Ils sont obligatoires dans tous*

1. `https://ec.europa.eu/info/law/law-making-process/types-eu-law_fr`

leurs éléments dans tous les pays de l'Union Européenne. Ils constituent l'un des types d'actes juridiques de l'Union Européenne (au côté notamment du traité, des directives et des décisions). Lorsqu'un conflit entre un règlement européen et le droit national se présente, le principe de primauté [2] du droit Européen est appliqué, donnant la priorité au droit Européen sur le droit national. Ainsi lorsqu'un règlement européen est voté, en France, il doit être adopté tel quel et doit être respecté même si le droit Français est en contradiction avec le droit Européen. Le **Règlement Général sur la Protection des Données** [RGPD 2018] est le règlement européen de référence traitant de la protection des données personnelles. Le RGPD est un règlement généraliste, il ne spécifie pas de cas d'application concrets mais traite des processus de récolte et de traitement automatique de données personnelles en général. Ce règlement définit notamment la licéité des traitements automatiques de données personnelles, les droits de la personne concernée par les données et les obligations des responsables et sous-traitants du traitement des données.

1.4.4 Inférence

En France, la récupération de données personnelles est donc encadrée par le RGPD. De surcroît, l'article 9 du RGPD précise que le traitement de données sensibles est interdit mais cet article dresse quelques exceptions telles qu'en cas de consentement explicite de la part de l'utilisateur ou lorsque le traitement est nécessaire à la sauvegarde des intérêts vitaux de la personne concernée. Ainsi la récupération de données sensibles est interdite ou très encadrée. Cependant, bien que dans les véhicules connectés d'aujourd'hui les données sensibles ne sont pas explicitement saisies par le conducteur, d'autres données personnelles peuvent, par croisement d'informations, révéler des informations sensibles le concernant. Parmi les données générées par le véhicule connecté, les données de géolocalisation sont réputées permettre de telles inférences. Prenons l'exemple d'un véhicule envoyant régulièrement sa géolocalisation à une partie prenante et d'un conducteur réalisant régulièrement des trajets entre son lieu de travail et son domicile. En utilisant son historique de géolocalisation, certaines informations personnelles peuvent être inférées telles que la position de son lieu de travail et de son domicile [Krumm 2007, Gambs 2010]. Considérons que ce même conducteur utilise son véhicule pour se rendre régulièrement dans un lieu où se rejoignent les membres d'un même partie politique, ou pour se rendre à un hôpital traitant une maladie spécifique, certaines informations sensibles peuvent alors être inférées telles que l'orientation politique ou l'état de santé du conducteur.

1.4.5 Cloud et véhicules connectés

Considérons que les données du véhicule connecté sont externalisées vers un cloud, bien que la récupération de données personnelles soit encadrée, le cloud peut

2. `https://eur-lex.europa.eu/legal-content/FR/TXT/?uri=LEGISSUM%3Aprimacy_of_eu_law&qid=1637156549533`

être malveillant et utiliser ces données pour des usages dépassant le but initial de leur récolte. De tels usages peuvent être l'inférence de données sensibles, le partage des données à des destinataires non prévus ou l'utilisation des données pour le bénéfice personnel du fournisseur ou de l'opérateur du cloud. Considérons que le cloud n'est pas malveillant, il peut néanmoins être vulnérable à des attaques. Il arrive d'entendre régulièrement dans les médias les termes brèches de données (data breach) et fuites de données (data leak) que nous définissons dans le cadre des données personnelles dans ce qui suit. Une **brèche de donnée** est un accès non prévu à tout ou partie d'une base de données privée contenant des données personnelles. Une **fuite de donnée** est une diffusion non prévue de données personnelles en dehors de la base de donnée privée dans laquelle elles étaient stockées, cette diffusion peut être réalisée dans un espace privée ou publique. Certains sites web recensent de telles brèches et fuites de données [3] [4], nous pouvons constater que ces événements arrivent de manière régulière et concernent parfois des quantités de données très conséquentes. Il est fort probable qu'à l'avenir une brèche ou fuite de données puisse toucher un cloud hébergeant les données de véhicules connectés.

En résumé, dans cette thèse nous considérons que la vie privée d'un conducteur est menacée lorsque le cloud ou tout autre destinataire non prévu accèdent aux données en clair de véhicules connectés suite à un accès non prévu aux données par le cloud, à une erreur de la part du cloud ou à la réussite d'une attaque par un attaquant externe au cloud, entraînant alors une brèche ou fuite de données. Notre objectif est donc de protéger les données personnelles des véhicules connectés stockées dans le cloud à l'encontre du cloud et des destinataires non prévus.

1.5 Contraintes de la solution

Le véhicule est un système très contraint, l'ajout dans le véhicule de mécanismes de protection des données qu'il communique est un véritable défi. Les calculateurs du véhicule sont des systèmes embarqués aux ressources restreintes, ils possèdent une faible puissance de calcul, une mémoire limitée et de faible capacités de stockage, ce qui contraint les ressources utilisables par les programmes s'exécutant sur ceux-ci. Les calculateurs du véhicule sont également des systèmes temps réel, ils doivent réaliser des tâches critiques périodiquement, les programmes disposent alors d'une fenêtre de temps pour s'exécuter. Par ailleurs, les véhicules ont une longue durée de vie, environ d'une dizaine d'année, les programmes s'exécutant sur les calculateurs doivent rester valides pendant cette durée. Enfin, les véhicules connectés émettent une grande quantité de données, les données se partagent la bande passante entraînant un temps d'envoi limité pour chaque donnée, les messages à émettre doivent être conçus afin d'avoir une taille minimale.

Nous considérons que la législation doit être prise en compte dans la mise en place de mécanismes de protection des données. Le RGPD étant un règlement généraliste,

3. `https://en.wikipedia.org/wiki/List_of_data_breaches`
4. `https://www.zdnet.com/article/the-biggest-data-breaches-of-2021`

il n'est pas adapté à des situations spécifiques dans lesquelles certaines données doivent être accessibles de manière obligatoire à certaines parties prenantes sous certaines conditions, ces situations font l'objet de lois plus spécifiques. En France, la *Loi d'Orientation des Mobilités (LOM)* [LOM 2019] du 24 décembre 2019 traite des mobilités en général avec notamment pour objectifs de permettre un meilleur investissement dans les transports du quotidien ou encore permettre le développement de transports plus écologiques, par ailleurs elle définit également des situations de partage obligatoire de données des véhicules connectés. L'article 32 de LOM définit de telles situations, en voici un exemple tiré du 2° du paragraphe I : *Rendre accessibles, en cas d'accident de la route, les données des dispositifs d'enregistrement de données d'accident et les données d'état de délégation de conduite enregistrées dans la période qui a précédé l'accident aux officiers et agents de police judiciaire aux fins de détermination des responsabilités ainsi qu'aux organismes chargés de l'enquête technique et de l'enquête de sécurité prévues à l'article L. 1621-2 du code des transports.* Cependant, cet article n'est pas encore abouti, il manque de précisions comme nous pouvons le constater au travers de la phrase suivante du 1° du paragraphe I : *Rendre accessibles les données pertinentes des systèmes intégrés aux véhicules terrestres à moteur, équipés de dispositifs permettant d'échanger des données avec l'extérieur du véhicule ...*, les données pertinentes en question ne sont pas précisées dans la suite de l'article. Ainsi, bien que le cadre législatif est en cours d'élaboration, nous considérons que les solutions techniques doivent donner la possibilité d'intégrer des contraintes de partage de données provenant de la législation présente et future.

Dans les véhicules d'aujourd'hui, aucun moyen n'est proposé au conducteur pour qu'il puisse intervenir dans le contrôle d'accès des données émises par son véhicule, ce contrôle étant à la charge de l'entité stockant les données. Nous considérons qu'il est primordial que le conducteur puisse exprimer son choix de partager ou non certaines de ses données. Toutefois, le conducteur ne peut pas tout se permettre, même s'il souhaite protéger au mieux sa vie privée, il reste soumis à la loi. En particulier, si du point de vue de la loi, une donnée doit être lisible, par exemple, par le constructeur du véhicule, alors le conducteur doit s'y soumettre.

La section suivante donne un rapide aperçu des principales approches qui peuvent être adoptées pour protéger les données issues des véhicules connectés, ainsi que leurs avantages et inconvénients.

1.6 Approches pour la protection des données

Les sections précédentes permettent de réaliser les constats suivants concernant la protection des données des véhicules connectés : 1) le conducteur n'a pas le contrôle sur les données régulièrement émises par son véhicule, il ne peut donc pas empêcher certaines informations qu'il considère privées d'être transmises ; 2) les données sont stockées sur un espace de stockage, en général, un cloud privé, et l'entité qui gère ce cloud peut accéder à toutes ces données à sa discrétion ; et 3) aucun mécanisme

n'est aujourd'hui à notre connaissance mis en place pour que les données puissent être consommées par certaines entités en respect des lois et des choix du conducteur. Face à ces constats, plusieurs mécanismes de protection des données peuvent être mise en place. Nous considérons que les mécanismes de protection des données personnelles doivent notamment être déployés dans le véhicule afin que les données bénéficient de propriétés permettant de protéger la vie privée du conducteur une fois émises. Ces mécanismes de protection peuvent être répartis en deux grandes familles : les mécanismes réduisant le lien entre les données et la personne concernée, et les mécanismes protégeant la confidentialité des données. Dans la suite de cette section, nous exposons ces approches en indiquant leurs avantages et inconvénients et en considérant aussi bien la facilité de déploiement, la surcharge entraînée et la capacité à garantir que la loi et les choix du conducteurs sont respectés.

1.6.1 Réduction du lien entre les données et la personne concernée

Les mécanismes permettant de réduire le lien entre les données et la personne concernée sont constitués essentiellement de mécanismes de pseudonymisation et d'anonymisation. Dans ce qui suit, la pseudonymisation et l'anonymisation sont définis à partir des définitions de la CNIL [5].

La **pseudonymisation** est *un traitement de données personnelles réalisé de manière à ce qu'on ne puisse plus attribuer les données relatives à une personne physique sans information supplémentaire. En pratique, la pseudonymisation consiste à remplacer les données directement identifiantes (nom, prénom, etc.) d'un jeu de données par des données indirectement identifiantes (alias, numéro séquentiel, etc.). La pseudonymisation permet ainsi de traiter les données d'individus sans pouvoir identifier ceux-ci de façon directe.* Plusieurs approches peuvent être imaginées pour pseudonymiser les données des véhicules connectés. Une première approche consisterait à remplacer toutes les informations directement identifiantes dans les données par un pseudonyme, stocker ce pseudonyme dans le véhicule afin de pouvoir lever le pseudonymat par une autorité compétente si nécessaire et d'émettre les données pseudonymiser à direction du cloud. Le problème de cette approche réside dans les capacités d'inférence du cloud, en effet le cloud va accumuler une quantité importante de données rattachées au même pseudonyme, il sera en mesure d'inférer des informations qui pourraient, sur la durée, révéler l'identité de la personne cachée derrière le pseudonyme. Une seconde approche, permettant de limiter l'inférence, consisterait à réaliser le procédé précédent en générant un pseudonyme différent pour chaque donnée. Cependant, le véhicule possède des capacités de stockage limitées, une telle approche n'est pas viable pour une grande quantité de données. Il est alors nécessaire de chercher des compromis entre les capacités d'inférence du cloud, la quantité de pseudonymes à générer et le stockage de ces pseudonymes.

L'**anonymisation** est un *traitement qui consiste à utiliser un ensemble de mécanismes de manière à rendre impossible, en pratique, toute identification de la personne par quelque moyen que ce soit et de manière irréversible.* Ainsi, une donnée

5. `https://www.cnil.fr/fr/lanonymisation-de-donnees-personnelles`

anonymisée ne doit pas permettre de retrouver l'identité de la personne concernée de manière directe ou indirecte. Pour illustrer les inconvénients de l'anonymisation, considérons qu'un processus d'anonymisation est effectué sur les données émises par le véhicule connecté. Tout d'abord, les mécanismes d'anonymisation peuvent nécessiter de réduire la précision des données, ce qui, dans le cadre d'un service fourni via l'exploitation des données, peut avoir un impact sur la qualité du service délivré ou sur la possibilité même de pouvoir délivrer le service. Ensuite, ces mécanismes demandent, dès la conception du système, d'anticiper les possibilités de réidentification et d'inférence pouvant être réalisées. Cependant, ces mécanismes ne sont pas parfaits, ainsi, avec le temps et l'accumulation de données, il est possible de trouver des méthodes non anticipées permettant de réidentifier les données. Enfin, nous considérons que les données, bien qu'anonymisées, appartiennent au conducteur et qu'il doit être en mesure de pouvoir décider du partage de ses données. Il est donc nécessaire également d'améliorer le contrôle d'accès à ces données de façon à disposer d'un contrôle d'accès grains fins sur les données. Cela revient à dire qu'une donnée, en fonction de sa nature, peut être consommée par certains entités qui sont autorisées à y accéder. Donc dans le cadre de nos travaux, nous nous intéressons aux approches permettant de protéger la confidentialité des données et à en contrôler l'accès.

1.6.2 Protection de la confidentialité des données

La protection de la confidentialité des données se fait généralement via l'utilisation de méthode de chiffrement des données.

Le chiffrement asymétrique classique [Diffie 1976, ElGamal 1985, Hoffstein 1998, Paillier 1999, Cramer 2002] permet de garantir la confidentialité des données transmises par le véhicule contre des attaques sur les réseaux de communication, il constitue aussi en lui-même une forme simple de contrôle d'accès puisque seuls les entités possédant la clé de déchiffrement peuvent accéder aux données en clair. Dans le contexte précédemment énoncé, une éventuelle approche consisterait à la génération d'un couple de clés de chiffrement / déchiffrement par chaque entité, les clés de chiffrement seraient déployées dans chaque véhicule, puis chaque donnée serait chiffrée avec les clés de chiffrement des destinataires désignés dans la législation et les choix du conducteur. Enfin, les destinataires correspondant déchiffreraient les données avec leur clé de déchiffrement. Bien que cette méthode permette de réaliser un contrôle d'accès, elle est trop coûteuse dans notre contexte. Tout d'abord le véhicule devra chiffrer une donnée plusieurs fois avec la clé de chiffrement de chaque destinataire, entraînant un temps de calcul trop élevé au vu du temps accordé à chiffrer une seule donnée. De plus, le véhicule devra stocker les clés de chiffrement de nombreux destinataires entraînant un coût de stockage élevé. Par ailleurs, il devra émettre tous les chiffrés sur le réseau entraînant un coût d'envoi trop important. Le cloud devra également stocker tous les chiffrés entraînant un coût de stockage très élevé.

Le chiffrement de groupe [Kiayias 2007, Libert 2014] peut être une alternative

au chiffrement asymétrique classique pour protéger les données du conducteur. En utilisant ce chiffrement, une approche consisterait à créer des groupes regroupant les destinataires recevant les mêmes données, chaque membre d'un groupe possède une clé de déchiffrement et le véhicule possède les clés de chiffrement de tous les groupes existants. Cette approche permet d'avoir des coûts moindres vis-à-vis du chiffrement asymétrique classique puisqu'elle dépend du nombre de groupes total pour le stockage des clés de chiffrement dans le véhicule et du nombre de groupes destinataires pour le nombre de chiffrement réalisé pour chaque donnée. Son principal inconvénient survient lorsque le véhicule doit envoyer des données à certains membres de groupes différents, de telle sorte que seuls ces membres doivent avoir accès à la donnée et non le groupe tout entier. Il faudra alors créer un groupe pendant l'exécution du système ce qui représente un coût élevé, cette solution manque de flexibilité.

D'autres mécanismes de chiffrement plus avancés peuvent être envisagés tel que le chiffrement homomorphe [Gentry 2009, Gentry 2012, Brakerski 2014, Cheon 2017] qui permet de réaliser des opérations sur les données chiffrées sans avoir besoin de les déchiffrer au préalable. Ce mécanisme peut être très utile lorsque la plus-value que certaines entités peuvent faire de la consommation des données émises par les véhicules réside dans les statistiques qu'ils peuvent établir à partir de ces données. Le chiffrement homomorphe peut être un moyen de réaliser ce type de plus-value sur les données sans pour autant en connaître la valeur des données individuellement. Cependant, ce type de chiffrement reste coûteux actuellement, le temps de chiffrement d'une donnée est assez élevé ainsi que la taille du chiffré généré. De manière similaire, les opérations homomorphes réalisables sur ces données sont également coûteuses, entraînant un temps de calcul élevé pour l'entité qui les réalise. Cette solution ne semble pas adaptée aux véhicules connectés dont les contraintes de stockage sont élevées, dont les performances de calcul sont limitées et dont les contraintes temps réel entraînent une durée d'exécution maximale.

Toutefois, d'autres mécanismes de chiffrement sont plus adaptés pour mettre en place un contrôle d'accès à grains fins, le *chiffrement basé attributs* (en anglais *ABE : Attribute-Based Encryption*) est un mécanisme de chiffrement asymétrique proposant un tel contrôle d'accès. ABE repose sur deux concepts pour spécifier le contrôle d'accès : les attributs et les arbres d'accès. Les arbres d'accès sont des arbres dont les nœuds internes sont des connecteurs logiques ET/OU et dont les feuilles sont des attributs. L'accès est accordé si l'arbre d'accès est satisfait par les attributs. Il existe deux constructions principales de schémas ABE : *Key-Policy Attribute-Based Encryption (KP-ABE)* [Sahai 2005, Goyal 2006, Ostrovsky 2007] lorsque les attributs sont définis lors du chiffrement et *Ciphertext-Policy Attribute-Based Encryption (CP-ABE)* [Bethencourt 2007, Waters 2011, Zhang 2012b] lorsque l'arbre d'accès est défini lors du chiffrement. ABE a l'avantage d'apporter la flexibilité manquante au chiffrement de groupe de part sa capacité à spécifier les attributs ou l'arbre d'accès au chiffrement, de nécessiter qu'une seule clé de chiffrement ce qui réduit l'espace de stockage requit dans le véhicule et de nécessiter un seul chiffrement pour chiffrer selon une liste d'attributs (KP-ABE) ou un arbre d'accès (CP-ABE).

Nous avons donc décidé dans ces travaux de thèse d'utiliser le chiffrement basé attributs pour mettre en place un mécanisme de contrôle d'accès, respectueux des choix du conducteur et de la législation. Cependant, les schémas ABE sont généralement considérés coûteux, au déchiffrement notamment, du fait de nombreuses exponentiations et de nombreux calcul d'une fonction de couplage. Il est alors important d'évaluer les performances de notre approche et de considérer l'implémentation d'optimisations pour réduire le coût de cet algorithme.

Nous présentons dans la section suivante, une conclusion à ce chapitre ainsi que le principal objectif de cette thèse.

1.7 Conclusion

En résumé, le contexte de nos travaux est relatif à l'augmentation du nombre de capteurs, d'actionneurs et de dispositifs de communication dans les véhicules. De ce fait, de nombreuses données sont produites par les véhicules, externalisées dans des clouds et consommées par différentes parties prenantes pour de multiples usages. Cependant, les données personnelles et notamment les données sensibles, lorsqu'elles sont utilisées par une entité malveillante, peuvent menacer les personnes concernées par ces données. Ainsi dans notre contexte, un cloud malveillant hébergeant les données des véhicules connectés en clair peut représenter une menace pour les conducteurs de par sa capacité à lire les données, les traiter, inférer des informations ou les partager avec des destinataires de son choix. De plus, dans le cas où le cloud n'est pas malveillant, il reste néanmoins vulnérable à des attaques pouvant entraîner des fuites de données. Il apparaît alors important de protéger la confidentialité des données à l'encontre du cloud et de destinataires non prévus en mettant en place des mécanismes de protection de données dans les véhicules. Les grandes approches pouvant traiter notre problème souffrent de plusieurs limitations, elles sont trop coûteuses au vu des ressources des véhicules et ne permettent pas de prendre en compte simplement la législation ou les choix de partage de données du conducteur. Ainsi, l'objectif de cette thèse est de protéger la confidentialité des données des conducteurs en proposant un contrôle d'accès cryptographique au moyen du chiffrement basé attributs. Ce chiffrement permet un contrôle d'accès des données des véhicules connectés à grains fins et permet de prendre en compte la législation et les choix du conducteur. Le chapitre suivant présente plus précisément le contexte de nos travaux ainsi qu'un état de l'art des travaux connexes.

Chapitre 2

Contexte technique et État de l'art

Sommaire

Ce second chapitre est consacré à la présentation plus précise du contexte et des hypothèses de notre étude, et à notre positionnement vis-à-vis de l'état de l'art. Nous commençons ce chapitre par la présentation de notre contexte concret, à savoir des différentes entités de l'architecture et des flux de données entre ces entités. Nous continuons ce chapitre par la présentation du modèle d'attaquants et des propriétés que le protocole doit satisfaire pour être considéré sécurisé au regard du modèle d'attaquants. Puis nous présentons des exigences que la méthode de génération du contrôle d'accès doit satisfaire. Nous poursuivons par une présentation d'un état de l'art des travaux connexes à notre approche. Une conclusion est proposée en dernière section.

2.1 Architecture et scénarios d'échanges de données

L'architecture considérée est celle définie dans le chapitre précédent, représentée en Figure 2.1, dans laquelle quatre entités interviennent : le constructeur, les véhicules, le centre de stockage et les parties prenantes. Le constructeur des véhicules (e.g. bus, voitures, camions) est le plus souvent considéré comme responsable du déploiement du centre de stockage et de la politique de contrôle d'accès. Les véhicules sont conduits par des conducteurs et envoient régulièrement des données, sur leur état ou leur environnement, au centre de stockage, par le biais de messages. Les véhicules sont

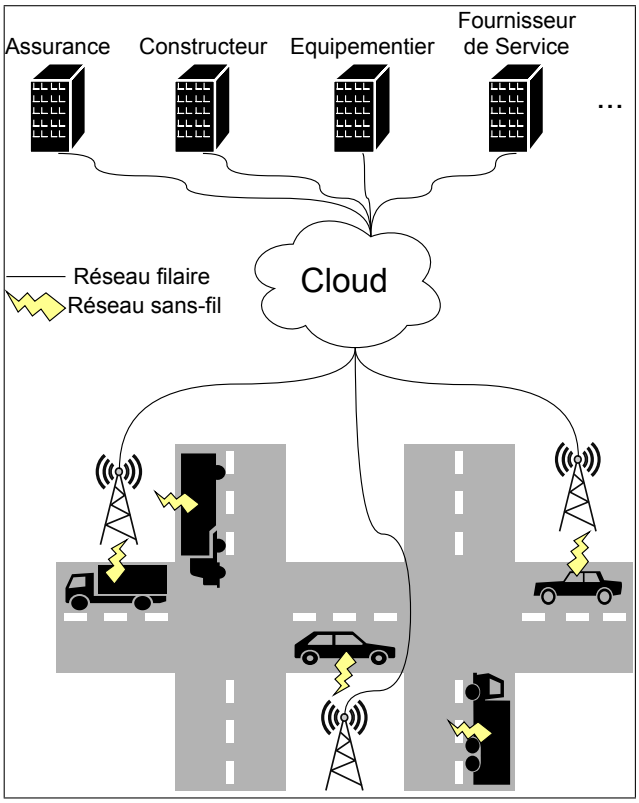

FIGURE 2.1 – Architecture conventionnelle

considérés avoir des ressources limitées. Le centre de stockage stocke les messages et les délivre par le biais de requêtes dédiées, il est considéré posséder des ressources quasiment illimitées et il est généralement déployé dans un cloud. Les parties prenantes peuvent être des services publics (e.g. police, palais de justice) ou des entreprises privées (e.g. sociétés informatiques, compagnies d'assurance, entreprises météorologiques). Elles sont connectées au centre de stockage pour récupérer les messages pertinents. Le constructeur du véhicule est également considéré comme une partie prenante qui cherche à accéder aux données auxquelles il est autorisé à accéder. Les véhicules communiquent avec le cloud via des moyens de communications sans-fil, ce lien est considéré avoir un débit limité. Les parties prenantes communiquent avec le cloud via des moyens de communications filaires, ce lien est considéré avoir un débit élevé.

L'architecture doit gérer trois scénarios d'échange de données. Dans les scénarios de récupération de données, les langages de requêtes ne sont pas représentés, nous considérons que les entités récupèrent toutes les données auxquelles elles sont autorisées à accéder. (**S1**) Un véhicule doit pouvoir envoyer des messages au centre de stockage et le centre de stockage doit stocker ces messages. (**S2**) Une partie prenante doit pouvoir interroger le centre de stockage pour récupérer les messages qu'elle est autorisée à lire et le centre de stockage doit délivrer les messages correspondants. (**S3**) Un véhicule doit pouvoir interroger le centre de stockage pour récupérer ses

propres messages et le centre de stockage doit renvoyer les messages correspondants au véhicule.

2.2 Modèle d'attaquant

Le modèle d'attaquant considéré est composé d'attaquants externes et d'entités légitimes honnêtes mais curieuses.

Un attaquant externe est un participant malveillant et illégitime aux communications. Un tel attaquant est supposé avoir accès à la totalité des messages échangés sur le réseau. Ce scénario pessimiste est volontairement envisagé, bien que dans des situations réelles, il est peu probable qu'un attaquant externe puisse réussir à intercepter tous les messages. L'attaquant est également supposé connaître tous les schémas cryptographiques utilisés pendant les communications et toutes les informations publiques associées (i.e. les clés publiques), cependant il n'a pas accès aux informations secrètes (i.e. les clés maîtresses et secrètes). Les opérations qu'un attaquant peut effectuer sont spécifiées dans le modèle de Dolev-Yao [Dolev 1983]. Par exemple, il peut récupérer un message, modifier un message, chiffrer un message, injecter un message ou effectuer une attaque de type *homme du milieu*.

Un participant légitime honnête mais curieux est censé effectuer ses opérations correctement, mais peut essayer d'obtenir plus d'informations sur les messages reçus légitimement sans coopérer avec d'autres entités. Ces participants ne sont pas considérés malveillants, ils ne cherchent pas délibérément à nuire au système mais ils cherchent à lire les données qu'ils ont obtenues de manière légitime ou non. Toutes les entités interagissant dans les scénarios de flux de données sont considérées honnêtes mais curieuses à savoir les véhicules, le centre de stockage et les parties prenantes. Par exemple, une partie prenante peut essayer de lire un message reçu alors qu'elle n'est pas le destinataire légitime du message. Une telle situation est considérée comme similaire à celle d'un attaquant externe possédant les informations secrètes d'un participant honnête mais curieux.

Le modèle de Dolev-Yao est limité car il ne prend pas en compte la fuite d'informations secrètes. Comme une fuite d'informations secrètes ne peut être exclue, même si elle n'est pas intentionnelle, l'impact d'une telle fuite doit être dûment analysé. Le modèle de Canetti-Krawczyk [Canetti 2001] étend le modèle de Dolev-Yao en prenant en compte de telles fuites. Dans le modèle d'attaquant considéré dans nos travaux, le modèle de Canetti-Krawczyk est partiellement pris en compte. En effet, la fuite des clés secrètes permettant de prouver l'identité de l'émetteur n'est pas considérée, mais la fuite des clés secrètes du véhicule, du centre de stockage ou d'une partie prenante, permettant à l'attaquant de déchiffrer les messages chiffrés correspondants, est considérée.

2.3 Propriétés de sécurité

Les propriétés de sécurité identifiées dans ce qui suit prennent en considération le modèle d'attaquant décrit précédemment, ce sont des propriétés de haut niveau ne tenant pas compte des détails d'implémentation de l'architecture.

Les premières propriétés concernent l'intégrité et l'authenticité. (**P1**) Le destinataire légitime (véhicule, centre de stockage, partie prenante) d'un message envoyé par un véhicule doit être en mesure de vérifier son intégrité et donc de détecter sa corruption potentielle par un attaquant. (**P2**) Le centre de stockage doit pouvoir identifier si les messages reçus proviennent d'un véhicule légitime enregistré auprès du constructeur ou non. Cette propriété permet de faire face à un attaquant externe injectant ou rejouant un message d'un véhicule.

Les propriétés suivantes concernent la disponibilité. (**P3**) Un véhicule doit pouvoir accéder aux données d'un message qu'il a précédemment envoyé. Cette propriété est d'autant plus importante qu'elle concerne la vie privée des conducteurs, telle que stipulée dans l'article 15 du RGPD : *La personne concernée a le droit d'obtenir du responsable du traitement la confirmation que des données à caractère personnel la concernant sont ou ne sont pas traitées et, lorsqu'elles le sont, l'accès aux-dites données à caractère personnel ....* (**P4**) Une partie prenante doit pouvoir accéder aux données d'un message pour lequel elle a été autorisée.

Les propriétés suivantes concernent la confidentialité. (**P5**) Le centre de stockage ne doit pas être en mesure de récupérer les données contenues dans les messages. (**P6**) Un destinataire qui récupère un message ne doit pas pouvoir accéder aux données incluses dans le message s'il n'a pas été autorisé à y accéder. Cette propriété est essentielle au vu des nombreuses fuites d'informations privées provenant de centres de stockage, elle permet de faire face aux participants honnêtes mais curieux ainsi qu'aux attaquants externes.

2.4 Exigences de contrôle d'accès

Dans cette section, nous définissons des exigences que le contrôle d'accès doit satisfaire pour répondre à la problématique.

La législation spécifie des situations dans lesquelles certaines données sont autorisées ou interdites d'accès par certaines parties prenantes. Ces lois sont en élaboration (telles que l'article 32 de la Loi d'Orientation des Mobilités [LOM 2019]), cependant nous considérons qu'il faut anticiper leur évolution. (**E1**) Le contrôle d'accès doit prendre en compte la législation. La législation ne spécifie pas une partie prenante précise mais spécifie des règles via des notions de rôles (e.g. forces de l'ordre, assurances, gestionnaires d'infrastructure routière). Par conséquent, de notre point de vue, la conformité à la législation semble être naturellement transposée à une approche basée sur des rôles.

Un conducteur peut souhaiter souscrire à un service proposé par une partie prenante, nécessitant certaines de ses données pour délivrer le service. Par exemple, un conducteur souscrit à un service de type *pay-as-you-drive* proposé par une

assurance en échange des données de vitesses, positions, d'accélérations et de forces de freinage. Nous considérons qu'un contrat est établi entre le conducteur et la partie prenante lorsque le conducteur souscrit au service. (**E2**) Le contrôle d'accès doit prendre en compte les contrats des conducteurs, pour autant qu'ils soient conformes à la loi. Nous considérons qu'un contrat spécifie les données requises par le service ainsi que la partie prenante traitant les données.

Un conducteur peut également consentir à partager ses données avec certaines parties prenantes pour leurs bénéfices personnels. Par exemple, un conducteur possédant un véhicule équipé de capteurs de pollution, consent à partager ses données de pollution avec des laboratoires d'analyse de pollution dans le but de contribuer à leurs recherches. (**E3**) Le contrôle d'accès doit prendre en compte le consentement des conducteurs. Le consentement doit permettre aux conducteurs de partager leurs données avec des parties prenantes.

Pour s'organiser, une partie prenante peut souhaiter déléguer ses droits d'accès aux données à une autre partie prenante. Par exemple, un gestionnaire d'infrastructure routière ayant des agences dans plusieurs régions délègue ses droits d'accès aux données à ses agences régionales. Nous considérons que la délégation est possible uniquement si la partie prenante a une relation légale (e.g. sous-traitant, filiale, département d'une entreprise) avec la partie prenante obtenant les droits délégués. (**E4**) Un mécanisme de délégation doit être inclus dans l'application du contrôle d'accès. La délégation ne doit pouvoir donner que des droits plus restreints à la partie prenante recevant les droits.

Enfin, certaines situations peuvent nécessiter qu'une partie prenante n'ayant initialement pas le droit d'accéder à une donnée puisse outrepasser ses droits pour accéder à certaines données précises. Nous considérons que cette situation a lieu lorsqu'une partie prenante devient assermentée auprès d'une autorité. Par exemple, considérons que les forces de l'ordre ne sont pas autorisées à accéder aux données de vitesse. À la suite d'un accident, un commissariat de police peut être assermenté afin d'accéder aux données de vitesse, précédent l'accident, des véhicules impliqués dans le but de réaliser une enquête. (**E5**) Le contrôle d'accès doit permettre un accès à certaines données aux parties prenantes assermentées. Cet accès doit se faire en spécifiant précisément l'identité du véhicule, le type de la donnée, le lieu et la date d'émission de la donnée.

2.5 Analyse des propositions industrielles existantes

Dans les architectures traditionnelles (telles que [1]), les mécanismes de sécurité sont généralement mis en œuvre 1) pour garantir la confidentialité des données pendant les communications contre un attaquant passif écoutant le réseau ; 2) pour authentifier un expéditeur (un véhicule) afin de s'assurer qu'un message a été envoyé par un expéditeur légitime ; et 3) pour authentifier un récepteur (une partie prenante)

1. `https://www.bmwgroup.com/en/innovation/innovationen-und-mobilitaet/cardata.html`

afin de s'assurer qu'il a les droits d'accéder aux données. Ainsi, cette architecture couvre les propriétés **P1**, **P2** et **P4**.

Cependant, puisque la confidentialité des données est assurée par des communications chiffrées alors que les données sont stockées en clair dans le centre de stockage, la propriété **P5** n'est pas satisfaite. Le centre de stockage est une entité clé responsable de l'application du contrôle d'accès, elle a un contrôle total sur les données en clair. Une fuite ou un mauvais comportement de sa part peut exposer les données à un destinataire non autorisé : la propriété **P6** est également insatisfaite. Enfin, les architectures classiques ne font pas de la possibilité pour les conducteurs d'accéder à leurs propres données un objectif prioritaire. Ainsi, il est nécessaire de donner cette possibilité aux conducteurs pour assurer la propriété **P3**.

Pour faire face à ces problèmes, nous proposons un protocole de sécurité visant à améliorer les architectures conventionnelles et ainsi assurer les propriétés manquantes : **P3**, **P5** et **P6**. Ce protocole s'appuie sur le chiffrement basé attributs et le chiffrement symétrique pour garantir la confidentialité des données contre les destinataires non autorisés (y compris le centre de stockage honnête mais curieux) et sur la signature de groupe pour authentifier les véhicules. Les schémas cryptographiques sous-jacents sont suffisamment génériques pour s'adapter aux algorithmes post-quantiques. Nous proposons également une méthode de génération de la politique de contrôle d'accès satisfaisant les exigences de **E1** à **E5**. Pour cela, une autorité de confiance est incluse dans l'architecture, elle est en charge de la gestion des permissions pour les différentes parties prenantes, en respect de la loi.

2.6 Analyse des propositions académiques existantes

Un état de l'art des travaux de la littérature connexes aux nôtres est proposé dans cette section. Nos travaux étant composés de trois principales contributions, l'état de l'art dans ce qui suit est réalisé pour chacune d'entre elles : le protocole cryptographique, la génération du contrôle d'accès en conformité avec la loi et l'optimisation des algorithmes cryptographique ABE.

2.6.1 Protocole cryptographique

Pour adresser les problématiques de sécurité et de confidentialité dues à l'externalisation des données dans les infrastructures cloud, des schémas et protocoles cryptographiques adaptés sont proposés dans la littérature. L'étude récente de Domingo-Ferrer et al. [Domingo-Ferrer 2019] résume ces différentes propositions. Dans cette étude, le *chiffrement basé identités* (en anglais *IBE : Identity-Based Encryption*) et l'Attribute-Based Encryption sont considérés comme pertinents dans ce contexte. Alors que les schémas IBE représentent les destinataires autorisés en utilisant une identité, les schémas ABE spécifient un ensemble de destinataires autorisés en utilisant des attributs. Les schémas IBE ne nous semblent pas adaptés à la mise en place d'un contrôle d'accès dérivé de la législation car cela entraînerait

une gestion complexe des identités pour traduire la législation et nécessiterait certainement plusieurs chiffrements pour une même donnée selon les différentes identités réceptrices. Les schémas ABE sont préférés aux schémas IBE dans nos travaux, car ils sont plus flexibles pour appliquer une politique de contrôle d'accès à grain fin.

De nombreux travaux utilisent ABE pour sécuriser les données des utilisateurs dans le Cloud. Nous donnons deux exemples de tels travaux dans ce qui suit à titre illustratif.

Zhang et al. [Zhang 2022] proposent un mélange entre un schéma CP-ABE et un protocole permettant de cacher partiellement l'arbre d'accès du chiffré, et de révoquer des clés et des attributs d'utilisateurs jugés illégaux. Leur proposition permet au propriétaire des données de protéger leur confidentialité et à en contrôler l'accès lorsqu'il souhaite les partager avec des utilisateurs via un Cloud. Leur proposition permet de protéger la confidentialité des données contre un Cloud curieux et contre n'importe quels autres utilisateurs non-autorisés à accéder à la donnée contenue dans le chiffré. Leur proposition est particulièrement pertinente dans des situations où, par exemple, un patient souhaite partager certaines données médicales avec un médecin, en chiffrant les données avec CP-ABE, en précisant le département médical et l'identité du médecin. Dans une telle situation, si l'arbre d'accès est accessible, le Cloud est donc en mesure d'inférer des informations sur la maladie du patient. Ainsi, les auteurs proposent une solution dans laquelle cet arbre est partiellement caché et l'inférence n'est plus possible. Leur solution permet également, lorsque le médecin quitte l'hôpital par exemple, de révoquer ses clés, de telle sorte que les nouveaux chiffrés émis avec les attributs du médecin ne puissent plus être déchiffrés par celui-ci.

Yang et al. [Yang 2022] proposent un mélange entre un schéma CP-ABE et un protocole post-quantique avec révocation d'utilisateurs et d'attributs. Leur schéma est également multi-autorités, c'est-à-dire qu'il existe une autorité centrale et plusieurs sous autorités qui gèrent un sous-domaine de l'espace d'attributs, ces sous-autorités pouvant émettre des clés de déchiffrement ainsi que révoquer ou mettre à jour les clés des utilisateurs. Leur schéma a pour but de protéger la confidentialité des données de propriétaires de données lorsqu'ils les stockent dans un Cloud, tout en permettant à des utilisateurs autorisés de récupérer ces données. La confidentialité des données est protégée contre n'importe quel utilisateur non autorisé suite à une fuite de donnée, ou contre un cloud curieux tout en permettant un contrôle d'accès sur ces données.

Le reste de cette section se concentre sur les articles qui traitent spécifiquement de la sécurité des données dans les scénarios de communication des véhicules connectés. Une classification de ces articles est présentée en Table 2.1. Notons que la colonne **Comm** fait référence au destinataire du message, en particulier la direction de la communication est considérée comme v2v (vehicle-to-vehicle) si le destinataire est un véhicule, même si pour certains articles les données sont temporairement stockées dans un cloud avant d'atteindre leur destinataire final. De même, certains articles traitent des communications sécurisées entre un véhicule et un cloud, et ne mentionnent que l'existence d'une partie prenante qui traitera les données pour fournir un service. Ces articles sont classés dans la catégorie v2s (vehicle-to-stakeholder).

TABLE 2.1 – Comparaison des caractéristiques de la proposition avec celles des travaux connexes de la littérature

	Comm	Schéma	Généricité	Proto	Post-Q	Loi	Vérif
[Xiong 2020]	v2v	IBE	Non	○	○	○	○
[Ruj 2011]	v2v	ABE	Non	○	○	○	○
[Huang 2009]	v2v	ABE	Non	●	○	○	○
[Liu 2016]	v2v	ABE	Non	●	○	○	○
[Huang 2018]	v2v	ABE	Non	●	○	○	○
[Zhao 2019b]	v2v	ABE	Non	●	○	○	○
[Pan 2019]	v2v	ABE	Non	●	○	○	○
[Huang 2019]	v2v	ABE	Non	●	○	○	○
[Feng 2020]	v2v	ABE	Limité	○	○	○	○
[Zhao 2019a]	v2s	IBE	Non	○	○	○	○
[Vaanchig 2020]	v2s	IBE	Limité	○	○	○	○
[Luo 2018]	v2s	ABE	Non	●	○	○	○
[Horng 2020]	v2s	ABE	Non	●	○	○	○
Notre proposition	v2s	ABE	Générique	●	●	●	●

● représente un critère satisfait et ○ représente un critère insatisfait.
(**Comm**) Les messages sont-ils émis d'un véhicule vers d'autres véhicules (vehicle-to-vehicle : v2v) ou d'un véhicule vers des parties prenantes (vehicle-to-stakeholder : v2s) ? (**Schéma**) Le schéma s'appuie-t-il sur ABE ou IBE ? (**Généricité**) La solution proposée peut-elle être adaptée à d'autres schémas cryptographiques de la littérature ? (**Proto**) L'article propose-t-il un nouveau schéma cryptographique ou un protocole qui utilise des schémas existants / un mélange entre un schéma et un protocole ? (**Post-Q**) La contribution est-elle valable face à un adversaire quantique ? (**Loi**) L'article prend-il en compte les contraintes juridiques de la loi ? (**Vérif**) L'article fournit-il des garanties de sécurité formellement vérifiées avec des outils de vérification automatique ?

La plupart des articles de la Table 2.1 se concentrent sur la sécurité des communications v2v [Feng 2020, Zhao 2019b, Pan 2019, Huang 2019, Huang 2018, Liu 2016, Ruj 2011, Huang 2009, Xiong 2020] et traitent principalement de la sécurité des messages envoyés entre véhicules pour améliorer la conduite ou prévenir un accident, par exemple en anticipant les collisions ou en avertissant le conducteur des embouteillages. Comme la législation établit des règles spécifiant les accès autorisés ou interdits à des données spécifiques pour des parties prenantes spécifiques, les approches axées sur la sécurité des communications v2v ne sont pas appropriées car elles ne prennent pas en compte le contrôle d'accès aux données envoyées par les

véhicules et consommées par les différentes parties prenantes.

Seuls quelques articles se concentrent sur les communications v2s [Zhao 2019a, Vaanchig 2020, Horng 2020, Luo 2018] et ont quelques liens avec notre proposition. Les propositions de Zhao et al. [Zhao 2019a] et Vaanchig et al. [Vaanchig 2020] sont basées sur des schémas IBE et les protocoles proposés par Horng et al. [Horng 2020] et Luo et Ma [Luo 2018] reposent sur ABE.

Zhao et al. [Zhao 2019a] considèrent un scénario dans lequel des véhicules envoient leurs données à un Cloud pour réaliser une sauvegarde de celles-ci, afin par exemple de fournir des preuves lors d'un accident de la route. Ces données devront être accessibles à des utilisateurs autorisés. Les auteurs souhaitent protéger la confidentialité des données des véhicules stockées dans le Cloud. Pour ce faire, ils proposent un schéma de chiffrement basé identité et une architecture composée : 1) d'une autorité qui génère les paramètres du système et les clés de déchiffrement ; 2) de véhicules qui chiffrent leurs données avant de les envoyer au Cloud ; 3) d'un Cloud qui stocke les chiffrés et répond aux requêtes ; et 4) d'utilisateurs qui envoient des requêtes, récupèrent et déchiffrent des chiffrés. Étant donné qu'une base de données contenant uniquement des informations chiffrées est difficile à traiter, les auteurs souhaitent permettre une comparaison des chiffrés afin que le Cloud puisse regrouper des chiffrés contenant le même clair sans les déchiffrer. Cela permettrait alors aux utilisateurs de chiffrer des mots-clés, puis de rechercher des chiffrés sur la base de ces mots-clés. Pour ce faire, leur schéma inclut également deux algorithmes : 1) un algorithme de génération d'une trapdoor exécuté par un utilisateur ; et 2) un algorithme de comparaison des chiffrés exécuté par le Cloud qui nécessite deux chiffrés et deux trapdoors pour vérifier si leur contenu est le même. Par ailleurs, les auteurs souhaitent également empêcher le déchiffrement par l'autorité de confiance. Pour ce faire la génération de la clé est décomposée en deux parties : une première partie réalisée par l'autorité et une seconde partie réalisée par le réceptionnaire de la clé de déchiffrement. Cette proposition souffre de plusieurs limitations. Tout d'abord leur schéma ne gère pas l'authentification, n'importe quel émetteur possédant les paramètres publics peut chiffrer des données, ainsi un attaquant pourrait perturber le Cloud en le surchargeant de messages. Les auteurs ne proposent pas de protocole, ainsi certaines problématiques se posent : une attaque par rejeu est-elle possible ? Comment se déroule concrètement l'utilisation des mots-clés, faut-il réaliser une comparaison avec tous les mots-clés de la base de données ce qui peut être coûteux ? Comment ces données peuvent être utilisées lors d'un accident ? Par ailleurs, leur proposition ne repose pas sur un schéma ABE. Elle n'est pas modulaire et s'appuie sur le problème Bilinear Diffie-Hemman qui est un problème vulnérable à l'ordinateur quantique. Enfin, l'évaluation des performances de leur schéma ayant été réalisée sur un ordinateur dont la puissance dépasse certainement celle des calculateurs d'un véhicule, nous questionnons le coût de leur schéma sur un ordinateur ayant une puissance similaire à celle d'un véhicule.

Vaanchig et al. [Vaanchig 2020] considèrent un scénario dans lequel un véhicule connecté envoie régulièrement une grande quantité de données à un Cloud afin que ces données puissent être utilisées par de nombreux services tel qu'un service

d'assurance automobile basé sur l'utilisation. Les auteurs souhaitent protéger la confidentialité des données externalisées vers le Cloud tout en permettant de réaliser une comparaison des chiffrés. Pour ce faire, ils proposent un schéma basé identité incluant un mécanisme de comparaison des chiffrés et une architecture identique à celle de l'article ci-dessus, composée d'une autorité, de véhicules, d'un Cloud et d'utilisateurs. En revanche, leur construction cryptographique est différente puisque basée sur deux schémas génériques : un schéma de chiffrement asymétrique reposant sur celui d'ElGamal [Guo 2018] et un schéma IBE reposant sur celui de Sakai-Kasashara [Sakai 2003, Chen 2005]. Le mécanisme de comparaison des chiffrés repose, comme pour l'article ci-dessus, sur l'utilisation de trapdoors. Ce mécanisme peut représenter une vulnérabilité lors d'une fuite d'une trapdoor, car n'importe qui serait en mesure de réaliser des comparaisons de chiffrés ce qui peut permettre de retrouver le message si l'espace de message n'est pas d'une grande taille. Pour faire face à ce problème, le mécanisme proposé dans cet article nécessite un secret possédé par le Cloud pour réaliser les comparaisons. Cependant, les auteurs restreignent la comparaison aux chiffrés générés par la même identité. Leur proposition souffre des mêmes limitations que la proposition du papier précédent : pas de gestion de l'authentification, pas de protocole concret, un schéma basé identité pas adapté à la mise en place d'un contrôle d'accès issu de la législation, des algorithmes cryptographiques vulnérables à l'ordinateur quantique et des évaluations de performances qui n'ont pas été réalisées sur des calculateurs d'un véhicule.

Horng et al. [Horng 2020] considèrent un scénario dans lequel des véhicules connectés peuvent communiquer entre eux et envoient régulièrement des données à un Cloud. Les données stockées dans le Cloud peuvent être consommées par des fournisseurs de service afin par exemple de permettre à un service de gestion du trafic de gérer les signaux de circulation pour éviter la congestion du trafic. Les auteurs souhaitent protéger la confidentialité des données en utilisant un schéma CP-ABE. Cependant, les schémas CP-ABE de base reposent sur une seule autorité ce qui peut être un goulot d'étranglement lorsque de nombreux véhicules doivent s'authentifier et obtenir des attributs auprès de l'autorité. Par ailleurs, dans ces schémas la révocation des attributs n'est généralement pas prise en compte, et les algorithmes peuvent être coûteux. Pour faire face à ces problèmes, ils proposent un schéma et un protocole multi-autorités, incluant un mécanisme de révocation des attributs, et permettant d'externaliser une partie du chiffrement ou du déchiffrement. Le schéma proposé dans cette contribution est plus complexe que celui des deux articles précédents et repose sur : 1) une autorité en charge de générer les paramètres publics ; 2) des fournisseurs de services qui fournissent des services aux véhicules et qui jouent le rôle de sous autorité afin de gérer les attributs en lien avec le service qu'ils fournissent ; 3) un Cloud qui stocke les messages chiffrés des véhicules et des fournisseurs de services, répond aux requêtes, et réalise la révocation lorsque cela est nécessaire ; 4) des unités en bord de route qui sont des intermédiaires entre les véhicules et le Cloud ; 5) des véhicules qui peuvent souscrire à un service et peuvent donc émettre et recevoir des chiffrés ; et 6) des nœuds déployés dans le Cloud qui peuvent réaliser une partie des opérations CP-ABE à la demande des véhicules. Leur

proposition souffre cependant des mêmes limitations que les articles précédents : pas de gestion de l'authentification, du rejeu, pas de preuve formelle, et des algorithmes vulnérables à l'ordinateur quantique.

Luo et Ma [Luo 2018] considèrent un scénario dans lequel des véhicules partagent des données entre eux et avec un Cloud afin d'améliorer les services fournis par, par exemple, des constructeurs automobiles, des agents de santé, et des centres de gestion du trafic. Les véhicules peuvent également s'organiser pour pouvoir louer leurs ressources (capacité de stockage, puissance de calcul, connectivité réseau) à d'autres véhicules. Les auteurs souhaitent protéger la confidentialité des données. Pour ce faire, ils proposent, comme la proposition précédente, un schéma CP-ABE auquel ils ajoutent également la possibilité d'avoir plusieurs autorités, la possibilité de révoquer des attributs, ainsi que d'externaliser une partie du déchiffrement pour que le véhicule réalise le moins d'opérations. Leur architecture est similaire à celle de la contribution précédente et souffre des mêmes limitations.

Dans l'ensemble, même si ces quatre articles évaluent la sécurité de leur schéma ou protocole à l'aide de preuves manuelles, les propriétés de sécurité sont discutées mais non prouvées formellement à l'aide d'un outil de vérification formelle. En outre, ces propositions reposent sur des schémas qui peuvent être réduits au problème du logarithme discret et sont donc vulnérables à un attaquant quantique. De plus, ils ne vérifient pas si l'émetteur d'une donnée chiffrée est un véhicule légitime ou non et les auteurs ne permettent pas à un véhicule légitime d'accéder ultérieurement à certaines données qu'il a préalablement envoyées au centre de stockage. Enfin, les auteurs n'expliquent pas comment la gestion du contrôle d'accès peut être appliquée, notamment pour être conforme à la loi, ce qui est, de notre point de vue, essentiel puisque la communication prend en compte les parties prenantes censées consommer les données.

2.6.2 Génération du contrôle d'accès

Une des contributions de ce travail de thèse est de générer automatiquement les règles de contrôles d'accès à partir de la loi sous forme de textes de loi informelles. Cette contribution fait l'objet du Chapitre 4. Nous présentons dans cette section les travaux connexes, qui se sont intéressés conjointement à la législation et à la mise en œuvre technique dans les systèmes informatiques des lois qui y sont décrites. Ce sujet est déjà abordé dans la littérature, même s'il reste un sujet relativement peu exploré aujourd'hui. En particulier, depuis que le RGPD doit être appliqué, certains travaux s'intéressent à la prise en compte de la législation pour vérifier la conformité des systèmes informatiques d'une organisation.

Agarwal et al. [Agarwal 2018] cherchent à proposer un questionnaire à un membre d'une organisation pour déterminer si l'organisation est conforme au RGPD. Pour ce faire, les auteurs décomposent leur approche en deux étapes : une première étape de génération d'un modèle du RGPD et une deuxième étape de vérification de la conformité avec le RGPD. La première étape de génération du modèle consiste à filtrer le contenu du RGPD pour en extraire toutes les obligations, ce que l'organisation

doit respecter. Ces obligations sont ensuite transformées et enrichies pour faciliter leurs écriture dans un format utilisable par la machine. Puis, elles sont traduites dans le formalisme Open Digital Rights Language (ODRL) permettant de représenter une politique sous forme de règles spécifiant un Asset, une Action, un Party et une autorisation de type Permission, Prohibition, ou Duty. Les auteurs étendent ce modèle pour rajouter des concepts nécessaires pour modéliser le RGPD. Lorsque ce modèle est établi, il est enrichi avec des questions permettant d'attester la conformité. La seconde étape de vérification de la conformité consiste tout d'abord à filtrer les règles qui ne sont pas applicables à l'organisation via des questions préliminaires portant sur les actions réalisées par l'organisation. Puis, les questions en lien avec les actions sélectionnées sont générées et l'utilisateur doit répondre à ces questions. Lorsque le questionnaire est rempli, un rapport évaluant la conformité au RGPD est renvoyé.

Torre et al. [Torre 2019, Torre 2021] cherchent à vérifier la conformité au RGPD d'une organisation en confrontant un modèle de l'organisation basé sur le RGPD avec un ensemble de règles que l'organisation doit respecter. Pour ce faire, leur approche est composée de quatre étapes :

— La première étape de génération d'un modèle conceptuel du RGPD sous la forme d'un diagramme de classe UML et d'un ensemble de contraintes au format Object Constraint Language (OCL) ;

— La deuxième étape de personnalisation du modèle en fonction de la loi du pays et d'informations contextuelles en rapport notamment au domaine d'application, cette étape renvoie un modèle et un ensemble de contraintes personnalisées pour le domaine d'application considéré ;

— La troisième étape d'instanciation du modèle pour l'organisation à évaluer en se basant sur des documents légaux et techniques fournis par l'organisation qui sont en lien avec le RGPD ;

— La quatrième étape consiste à vérifier la conformité du modèle instancié en troisième étape par rapport aux contraintes personnalisées renvoyées en deuxième étape.

Cependant, en Sous-Section 2.4, nous avons défini plusieurs exigences que doit satisfaire le contrôle d'accès, la première étant de prendre en compte la législation via des lois qui définissent des règles de contrôle d'accès aux données. Les travaux précédents n'ont pas pour objectif de générer de règles de contrôle d'accès à partir du RGPD. Cependant, dans la littérature un travail s'y intéresse pour le RGPD.

Bartolini et al. [Bartolini 2019] cherchent à générer des règles de contrôle d'accès à partir du RGPD. Pour ce faire leur approche est composée de trois étapes :

— Une première étape de sélection des articles du RGPD en lien avec des obligations ;

— Une deuxième étape d'extraction d'exigences techniques du RGPD et de formulations de ces exigences sous la forme de User Stories empruntées aux méthodes Agiles ;

TABLE 2.2 – User Stories axées sur le RGPD : perspectives du contrôleur et de la personne concernée (issus de la Table 1 de [Bartolini 2019])

Article	User story	AC rule
Art. 6.1(a)	As a [Controller], I want [to process Personal Data only if Data Subject has given consent for one or more specific purpose], so that [the processing shall be lawful].	[Controller] can [Process] [Personal Data] If [PersonalData.purpose = Processing.purpose AND PersonalData.purpose.consent = TRUE]
Art. 7.3	As a [Data Subject], I want [to withdraw my consent], so that [I can exercise my right as stated in Art. 7.3]	[Data Subject] can [Withdraw] [PersonalData.purpose.consent] If PersonalData.owner = DataSubject AND PersonalData.purpose.consent = TRUE]
Art. 15.1	As a [Data Subject], I want [to access my Personal Data and all the information], so that [I can be aware about my privacy]	[Data Subject] can [Action = access] [PersonalData] AND [Resource = PersonalData.purposes] AND [Resource = PersonalData.categories] if [PersonalData.owner = Data Subject]
Art. 15.3	As a [Data Subject], I want [to download a copy of my Personal Data], so that [I can check their correctness]	[Data Subject] can [download] [Personal Data] If [PersonalData.owner = Data Subject]

— Une troisième étape de traduction des User Stories sous la forme de règles de contrôle d'accès dans un formalisme adapté du formalisme eXtensible Access Control Markup Language (XACML).

La Table 2.2 est issue de cet article, elle représente un extrait des règles générées via leur méthode. Nous pouvons observer que les règles générées sont très génériques et sont applicables à n'importe quelle partie prenante traitant les données sans prendre en compte le type de la donnée, la partie prenante précise ou le contexte. Ces vérifications sont bien évidemment nécessaires et doivent être effectuées dans un système de contrôle d'accès. Cependant, notre objectif est plutôt d'établir, à partir de textes de loi de la législation en vigueur, des règles de contrôles d'accès explicites, définies et directement applicables pour des parties prenantes précises et des types de données identifiés. Seul un travail se rapproche de cet objectif.

Stieghahn et al. [Stieghahn 2010] considèrent un scénario bancaire dans lequel un membre d'une institution financière doit se rendre dans un pays autre que celui qui héberge les données en considérant que la loi du pays hébergeur interdit le transfert de certaines données vers certains pays. Ainsi, les auteurs cherchent à

mettre en place un contrôle d'accès issu de la législation lorsque l'employé accède aux données. Pour ce faire, les auteurs se basent sur la législation du Luxembourg et opèrent une traduction des articles de la loi vers le formalisme XACML. Ces règles sont contenues dans un serveur qui va vérifier si l'utilisateur a le droit d'accéder aux ressources qu'il demande. Cet article repose sur un système de contrôle d'accès classique et ne réalise donc pas un contrôle d'accès cryptographique tel qu'ABE le permet. En outre, bien que plusieurs articles proposent une méthode de traduction de la législation vers un formalisme de contrôle d'accès, aucun ne propose la mise en place d'un contrôle d'accès cryptographique issu de la législation.

La contribution du chapitre 4 a donc pour but de définir une méthode de génération de règles du contrôle d'accès instanciées sous la forme d'arbres d'accès et d'attributs qui seront utilisés pour réaliser un contrôle d'accès cryptographique via l'utilisation d'ABE. Cette méthode doit satisfaire les exigences de contrôle d'accès précédemment définis en prenant en compte la législation, les contrats du conducteurs, le consentement, la délégation de droits et les situations exceptionnelles.

2.6.3 Optimisation du déchiffrement ABE

Dans la littérature, de nombreux travaux optimisent l'algorithme de chiffrement ou déchiffrement ABE. En effet, ces algorithmes sont coûteux et peuvent induire des problèmes de performance de certains schémas ou protocoles, en particulier dans des contextes avec des ressources limitées ou des contextes avec une surcharge nécessitant d'exécuter l'un de ces algorithmes un très grand nombre de fois par une ou plusieurs entités. Dans cette thèse, ces deux problèmes existent : les ressources limitées au niveau du véhicule (pour le chiffrement), et la surcharge au niveau des parties prenantes (pour le déchiffrement). Nos travaux portent principalement sur l'optimisation du déchiffrement car nous avons tout d'abord pensé nos optimisations sur celui-ci. Nous n'apportons pas de contribution sur l'optimisation du chiffrement par manque de temps, mais il existe néanmoins des travaux et des approches pour l'optimiser, nous aborderons l'une des approches en section Discussion du Chapitre 6.

Certains travaux proposent d'externaliser le déchiffrement [Green 2011, Li 2013, Belguith 2018] de telle sorte que l'équipement de l'utilisateur ne réalise plus la totalité des opérations mais qu'un composant externe plus puissant, tel qu'un cloud, réalise la majorité des opérations et renvoie un résultat intermédiaire à l'équipement de l'utilisateur. Celui-ci réalise ensuite les dernières opérations moins coûteuses permettant de terminer le déchiffrement. Plusieurs travaux proposent également de nouvelles constructions conçues notamment pour diminuer le temps de calcul du chiffrement ou du déchiffrement [Malluhi 2019, Chandrasekaran 2020, Li 2019, Kaâniche 2017, Hohenberger 2013, Malluhi 2017, Ke 2021, Tan 2019]. À noter que les opérations de déchiffrements réalisées par le cloud sont conçues pour ne pas mettre en danger la confidentialité de la donnée car il n'est pas en mesure de réaliser le déchiffrement complet. Un attaquant qui pourrait capturer ces échanges n'aurait pas d'avantage la possibilité de déchiffrer les messages.

Notre objectif est de réduire le temps de calcul du déchiffrement de schémas

existants, sans proposer de nouvelles constructions. Nous souhaitons réduire le temps de calcul en modifiant le moins possible les algorithmes afin de définir des optimisations qui pourraient être applicables à d'autres schémas. Pour ce faire, nous pouvons :

— optimiser les opérations cryptographiques de bas niveaux. Ces opérations sont généralement implémentées par des algorithmes de l'état de l'art dans des bibliothèques cryptographiques de bas niveau telles que la bibliothèque RELIC [Aranha 2022] ;

— réaliser des pré-calculs à un moment opportun ou lors de la première exécution d'un algorithme avec des paramètres spécifiques. Ces pré-calculs sont ensuite utilisés lors de la prochaine exécution de l'algorithme ;

— modifier les opérations d'un schéma sans en modifier le résultat en exploitant les propriétés des opérations.

Nos travaux portent sur ces deux dernières approches.

L'algorithme de déchiffrement des schémas ABE prend généralement en entrée une clé de déchiffrement et un chiffré. La clé de déchiffrement est connue à l'avance et ne varie pas dans le temps, certaines opérations dépendant uniquement de la clé peuvent donc être pré-calculées. Puis, lorsque le chiffré est connu, les dernières opérations permettant de finaliser le déchiffrement peuvent être réalisées. En particulier, des fonctions nommées couplages sont utilisées au déchiffrement. Ces fonctions prennent deux paramètres, généralement un élément de la clé de déchiffrement et un élément du chiffré, puis réalisent le couplage. Plusieurs travaux proposent de pré-calculer l'un des deux paramètres du couplage. Selon le couplage utilisé le paramètre pré-calculable peut être celui de droite ou celui de gauche [Scott 2005, Scott 2006, Scott 2007, Costello 2010, Scott 2011]. Cependant, selon les schémas, tous les éléments de la clé de déchiffrement ne peuvent pas être mis du coté pré-calculable. Dès lors, l'ordre des opérations peut être changé pour pouvoir bénéficier le plus possible du pré-calcul. Certains travaux ont étudié l'utilisation du pré-calcul du couplage dans le cas de schémas basés identité [Scott 2005, Scott 2011] et dans le cas du schéma CP-ABE de base [Scott 2011]. Ces travaux ne considèrent uniquement que le pré-calcul d'éléments de la clé de déchiffrement car le chiffré est généralement considéré totalement aléatoire. Cependant, dans le cas d'ABE, certains éléments du chiffré ne sont pas aléatoires et peuvent être pré-calculés, cela nécessite plus de stockage mais permet de réduire encore plus le temps de calcul. Cette approche n'est pas exploitée dans ces travaux.

Un seul travail se rapproche le plus de notre approche. De la Piedra et al. [de la Piedra 2022] souhaite rendre la réalisation de l'évaluation et de la comparaison des schémas ABE plus précise. Les auteurs considèrent que dans de nombreux travaux de la littérature, les schémas ABE ne sont pas optimisés entraînant une évaluation des schémas pas assez précise. Par ailleurs, dans certains travaux, la comparaison des performances de schémas ABE est biaisée car une comparaison est réalisée d'un schéma ABE optimisé selon un aspect (e.g. le déchiffrement) avec des schémas ABE optimisés selon d'autres aspects (e.g. le chiffrement). Ainsi, les auteurs proposent un

framework pour pouvoir évaluer plus précisément les performances des schémas ABE. Pour ce faire, il proposent des optimisations à plusieurs niveaux, en considérant les opérations arithmétiques, le choix de la courbe elliptique, l'ordre des opérations, et l'instanciation du schéma sur la courbe choisie. Par ailleurs, ils réalisent ces optimisations en considérant l'aspect du schéma à optimiser. En revanche, les auteurs n'étudient pas l'impact du pré-calcul des couplages sur l'optimisation du déchiffrement. En effet, les auteurs n'arrivent pas à déterminer s'il est possible de réaliser leurs optimisations du déchiffrement tout en réalisant les pré-calculs des couplages car, selon l'algorithme de couplage, les modifications à réaliser sur le schéma pour bénéficier du pré-calcul ont un impact fort sur l'ordre des opérations et peuvent entraîner un temps de calcul plus long. Cependant, dans nos travaux, nous utilisons l'algorithme de couplage *ate* qui ne souffre pas de ce problème et permet de bénéficier à la fois du pré-calcul du couplage et des optimisations du déchiffrement.

Ainsi, la contribution du Chapitre 6 a pour but d'étudier l'utilisation du pré-calcul des couplages et l'ordre des opérations au déchiffrement de certains schémas KP-ABE. Nous introduisons une optimisation, le pré-calcul Stream consistant à pré-calculer un élément de la clé puissance des coefficients obtenus via la combinaison de la clé et du chiffré, ce pré-calcul n'a, à notre connaissance, jamais été proposé dans la littérature.

2.7 Conclusion

Ce chapitre présente plus précisément les contraintes que notre proposition doit satisfaire ainsi qu'une étude de l'état de l'art. Notre première contribution, consistant en un protocole cryptographique, est évalué en considérant un attaquant externe supposé malveillant et des participants légitimes considérés honnêtes mais curieux, les actions que peuvent réaliser ces entités correspondent au modèle de Dolev-Yao. Le protocole doit satisfaire des propriétés de sécurité en relation avec l'intégrité, l'authenticité, la disponibilité et la confidentialité des messages. Notre seconde contribution, consistant en une méthode de génération des attributs et arbres d'accès utilisés lors du contrôle d'accès cryptographique, doit satisfaire plusieurs exigences permettant d'assurer que le contrôle d'accès prend en compte la législation, les contrats du conducteur, le consentement du conducteur, permet la délégation de droit, et permet d'outrepasser les droits dans des conditions spécifiques. L'étude de l'état de l'art nous a montré que : 1) l'essentiel de la littérature propose des schémas cryptographiques mais pas de protocole explicite formellement prouvé ; 2) la dérivation des règles de contrôle d'accès depuis la législation n'est pas réalisée ou elle n'est pas utilisée pour mettre en place un contrôle d'accès cryptographique ; et 3) les travaux sur les optimisations d'ABE peuvent être étendus en considérant des éléments déterministes dépendant du chiffré. Le prochain chapitre présente donc notre première contribution, un protocole cryptographique reposant sur ABE dont les propriétés sont formellement prouvées à l'aide de l'outil ProVerif.

Chapitre 3

Protocole cryptographique

Sommaire

Ce troisième chapitre est consacré à la présentation de la première contribution à savoir un protocole sécurisé d'échanges de données, pour des véhicules connectés, reposant sur le chiffrement basé attributs. Ce chapitre débute par la présentation des schémas cryptographiques génériques utilisés dans le protocole. La section suivante est consacrée au protocole et aux scénarios qui le composent. La troisième section est dédiée à la modélisation formelle du protocole et à la vérification formelle de ses propriétés via ProVerif, un outil de vérification automatique de propriétés de protocole. De potentielles améliorations du protocole sont discutées en quatrième section. Ce chapitre se termine par une conclusion sur les travaux réalisés.

3.1 Schémas cryptographiques génériques

Le protocole s'appuie sur des schémas cryptographiques pour satisfaire les propriétés énoncées dans le chapitre précédent. Ces schémas cryptographiques sont génériques afin que le protocole puisse supporter autant que possible les schémas cryptographiques actuels et futurs (y compris les schémas post-quantiques). Tant que les schémas utilisés supportent de manière sécurisées tous les algorithmes définis, ils peuvent être utilisés dans le protocole. Le protocole s'appuie sur trois schémas

cryptographiques génériques : un schéma de chiffrement basé attributs, un schéma de chiffrement symétrique et un schéma de signature de groupe.

3.1.1 Chiffrement basé attributs

3.1.1.1 Définition

Un schéma de **chiffrement basé attributs** est composé généralement de quatre algorithmes, cependant une légère modification de la définition habituelle d'ABE est réalisée dans ce qui suit pour faciliter la compréhension du protocole. En particulier, un schéma KP-ABE est présenté, cette construction est celle utilisée dans le protocole (identifiée par le préfixe **abe**) :

- **abe_setup(1^λ)** : Étant donné un paramètre de sécurité λ, génère une clé maîtresse **MK**. Cette clé est utilisée pour générer à la fois la clé de chiffrement et les clés de déchiffrement ;
- **abe_pkgen(MK)** : En utilisant **MK**, génère la clé de chiffrement **PK** ;
- **abe_skgen(T, MK)** : Étant donné un arbre d'accès **T**, génère une clé de déchiffrement **SK** associée à **T** en utilisant **MK** ;
- **abe_enc(μ, X, PK)** : Étant donné un ensemble d'attributs **X**, chiffre le message μ en utilisant **PK** et génère le chiffré **C** ;
- **abe_dec(C, SK)** : Déchiffre **C** en utilisant **SK**, génère μ si et seulement si les attributs choisis durant la génération de **C** permettent de satisfaire l'arbre d'accès choisi durant la génération de **SK**.

3.1.1.2 Chiffrement basé attributs post-quantique et implications

ABE est généralement construit à partir de couplages bilinéaires [Boneh 2001] ou de réseaux [Zhang 2012b]. Les schémas ABE basés sur des couplages bilinéaires sont sujets à de fortes vulnérabilités contre les ordinateurs quantiques, ceux basés sur des réseaux sont considérés résistants aux ordinateurs quantiques [Gentry 2008, Micciancio 2013, Micciancio 2007, Peikert 2009, Regev 2009]. Historiquement, ABE est divisé en deux notions de sécurité : la sécurité sélective et la sécurité adaptative (également appelée sécurité totale). Pour un jeu de sécurité typique adversaire/challenger, la sécurité sélective exige que les arbres d'accès (ou attributs) soient connus avant que les paramètres publics soient générés. Cette limitation restreint généralement le type d'arbres d'accès réalisable par le schéma (ou le nombre d'attributs), ce qui limite l'expressivité de la politique de contrôle d'accès.

En raison d'une littérature assez mature sur les schémas ABE reposant sur les couplages bilinéaires, la sécurité adaptative est supposée pour tous les schémas récents avec support pour la classe de problèmes $\mathbf{NC^1}$. Pour les réseaux, le paysage est plus contrasté [Boneh 2014, Zhang 2012a], atteindre la sécurité adaptative est un problème ouvert depuis de nombreuses années, et bien que certaines constructions récentes aient atteint la sécurité adaptative, elles ont encore quelques limitations.

Tsabary et al. [Tsabary 2019] ont proposé le premier schéma CP-ABE post-quantique adaptatif en restreignant la structure des arbres d'accès de telle sorte qu'ils doivent suivre une t-Forme Normale Conjonctive (t-FNC), où t est le nombre exact de littéraux dans les clauses. Wang et al. [Wang 2020] ont proposé un schéma CP-ABE post-quantique adaptatif pour tout circuit de taille polynomiale, réduisant le coût du déchiffrement de $\mathcal{O}(n \log n)$ à $\mathcal{O}(n)$ avec des paramètres publics linéaires avec le nombre d'attributs.

3.1.2 Chiffrement symétrique

Un schéma de **chiffrement symétrique** (en anglais *SE : Symmetric Encryption*) est un schéma de chiffrement reposant sur la même clé pour le chiffrement et le déchiffrement. Un schéma SE est composé de trois algorithmes (identifiés par le préfixe **se**), l'implémentation standard la plus courante d'un schéma SE est l'Advanced Encryption Standard (AES) [Daemen 1999]. Une abstraction de ce schéma est présentée dans ce qui suit, cette abstraction est suffisamment générique pour inclure la plupart des schémas SE :

— **se_setup(1^λ)** : Étant donné un paramètre de sécurité λ, génère une clé **K** ;

— **se_enc(μ, K)** : En utilisant **K**, chiffre le message μ, renvoie le chiffré **C** ;

— **se_dec(C, K)** : En utilisant **K**, déchiffre le chiffré **C**, renvoie le message μ si et seulement si la même clé **K** a été utilisée pendant le chiffrement.

3.1.3 Signature de groupe

Un schéma de **signature de groupe** (en anglais *GS : Group Signature*) [Chaum 1991] est un schéma de signature permettant à tous les membres d'un groupe de générer anonymement des signatures au nom du groupe. Un émetteur de clés est généralement défini, il est responsable de la génération et de la distribution des clés de signature aux membres légitimes du groupe. Le premier schéma de signature de groupe post-quantique a été proposé en 2010 par Gordon et al. [Gordon 2010], la plupart des propositions existantes sont basées sur les réseaux [Laguillaumie 2013, Langlois 2014, Ling 2015], mais quelques propositions existantes sont basées sur les codes correcteurs d'erreurs [Ezerman 2015] et sur les fonctions de hachage [Shafieinejad 2021]. Un schéma GS est composé de cinq algorithmes (identifiés par le préfixe **gs**), une abstraction de ce schéma est présentée dans ce qui suit et est suffisamment générique pour inclure la plupart des schémas GS :

— **gs_setup(1^λ)** : Étant donné un paramètre de sécurité λ, génère une clé maîtresse **SIG_MK** ;

— **gs_pkgen(SIG_MK)** : En utilisant **SIG_MK**, génère la clé de vérification **SIG_PK** ;

— **gs_skgen(ID, SIG_MK)** : En utilisant **SIG_MK**, génère la clé de signature **SIG_SK** associée à l'identité **ID** ;

— **gs_sign(μ, SIG_SK)** : En utilisant **SIG_SK**, génère la signature **s** pour le message μ ;

— **gs_verif((μ,s), SIG_PK)** : En utilisant **SIG_PK**, vérifie si la signature **s** est une signature valide du message μ. Renvoie $\top$ si la signature est valide, $\bot$ sinon.

3.2 Description du protocole

L'écosystème des véhicules connectés est aujourd'hui très riche et les données émises par ces véhicules sont potentiellement très nombreuses. Elles peuvent être consommées par différentes parties prenantes, mais il est nécessaire que chaque partie prenante ne puisse accéder qu'aux données pour lesquelles elle est autorisée 1) soit par la loi, 2) soit par un contrat signé avec le propriétaire du véhicule, ou 3) soit par le consentement explicite de ce propriétaire. Ainsi, chaque attribut accompagnant le chiffrement de toutes données émises par le véhicule doit être positionné uniquement s'il entre dans le cadre d'une des trois situations précédentes. La mise en œuvre d'un contrôle d'accès, sécurisé et de confiance des données, provenant de ces différentes origines nécessite une modification de l'architecture classique présentée dans le chapitre précédent. En particulier, une autorité de confiance indépendante est nécessaire pour définir les attributs qui doivent être utilisés lors du chiffrement des données à l'aide d'ABE, dans le respect de la loi, et pour gérer les différents arbres d'accès des parties prenantes, également dans le respect de la loi.

Dans ce chapitre, nous considérons l'existence de deux fonctions fournies par l'autorité de confiance : la fonction **get_attrs** et la fonction **get_access_tree**. La fonction **get_attrs** prend en entrée l'identité du centre de stockage (ID_{cs}) ou l'identité d'un véhicule (ID_v) et un contexte (**C**), elle renvoie les attributs utilisés lors du chiffrement ABE. Cette fonction sélectionne les attributs en utilisant le contexte donné en entrée et associe l'identité donnée en entrée aux attributs renvoyés. Lorsqu'un contexte vide est donné, cette fonction renvoie uniquement l'identité donnée en entrée en tant qu'attribut. La fonction **get_access_tree** prend en entrée une identité d'entité (soit un véhicule, le centre de stockage ou une partie prenante), elle renvoie l'arbre d'accès utilisé lors de la génération de la clé de déchiffrement ABE. L'arbre d'accès d'un véhicule et du centre de stockage est un arbre contenant uniquement une feuille correspondant à l'identité du véhicule ou du centre de stockage. Dans le cas d'une partie prenante, l'arbre contient l'identité de la partie prenante et des attributs supplémentaires liés essentiellement à son rôle et au contexte. Ces fonctions sont établies en respect des exigences **E1** à **E5**, leurs définitions sont détaillées dans le chapitre suivant.

3.2.1 Respect des propriétés de sécurité

Les schémas cryptographiques précédemment décrits sont nécessaires pour satisfaire les propriétés de sécurité **P1** à **P6**.

Un schéma de signature est utilisé pour permettre à chaque entité légitime de vérifier l'intégrité des messages (**P1**) et pour permettre au centre de stockage de vérifier qu'un message a été émis par un véhicule légitime (**P2**). Cependant, un schéma de signature de groupe est utilisé pour éviter de surcharger le centre de stockage avec autant de clés de vérification que de véhicules légitimes. Pour gérer cette authentification, l'autorité de confiance exécute les algorithmes **gs_setup** et **gs_pkgen**. Ensuite, elle génère une clé de signature pour chaque entité légitime à l'aide de l'algorithme **gs_skgen**. Une entité légitime doit signer ses messages en utilisant sa clé de signature. De cette façon, chaque récepteur légitime peut vérifier la signature en utilisant la clé de vérification générée par l'autorité de confiance. La distribution des clés de signature est supposée être réalisée de manière sécurisée. La clé de signature d'un véhicule est intégrée dans le véhicule, avec les fonctions et algorithmes associés, au cours du processus de fabrication du véhicule.

Un schéma de chiffrement basé attributs est utilisé pour mettre en place un contrôle d'accès à grains fins. De plus, dans ce contexte, les schémas KP-ABE sont préférables aux schémas CP-ABE car ils ne nécessitent pas la génération d'arbres d'accès dans le véhicule, ce qui serait trop coûteux dans un tel système embarqué aux ressources limitées. Le chiffrement basé attributs permet à chaque véhicule d'accéder aux messages qu'il a émis au moyen d'une clé de déchiffrement contenant son identité, et de messages chiffrés contenant au moins un attribut correspondant à son identité (**P3**). Le chiffrement basé attributs permet également à chaque partie prenante d'accéder aux messages auxquels elle est autorisée au moyen d'une clé de déchiffrement contenant un arbre d'accès conforme à la loi, et de messages chiffrés avec des attributs conformes également à la loi (**P4**). Pour générer le matériel cryptographique correspondant, l'autorité de confiance exécute les algorithmes **abe_setup** et **abe_pkgen** pour générer la clé maîtresse et la clé de chiffrement, et la fonction **get_access_tree** pour générer l'arbre d'accès d'un lecteur de message autorisé, conformément à la loi. Lors de l'envoi d'un message, un véhicule doit utiliser la fonction **get_attrs** pour obtenir la liste des attributs à utiliser au chiffrement. Cette fonction est fournie au véhicule par l'autorité de confiance lors du processus de fabrication. La dérivation des arbres d'accès conformément à la loi empêche à la fois le centre de stockage (**P5**) et les destinataires non autorisés (**P6**) d'avoir accès aux données d'un message.

3.2.2 Scénarios du protocole

Le protocole est composé d'une étape d'initialisation : génération et distribution des clés ; et de trois scénarios : scénario d'envoi sécurisé des données du véhicule (**S1**), scénario de lecture sécurisée des données par une partie prenante (**S2**) et scénario de lecture sécurisée des données par un véhicule (**S3**). Les symboles utilisés dans le protocole sont représentés en Table 3.1.

TABLE 3.1 – Symboles du protocole

Description	Symbole
Identité d'un Véhicule	ID_v
Identité du Centre de Stockage	ID_{cs}
Identité d'une Partie Prenante	ID_{pp}
Nonce d'un Véhicule	N_v, X_v
Nonce du Centre de Stockage	N_{cs}, X_{cs}
Nonce d'une Partie Prenante	N_{pp}, X_{pp}
Donnée	**D**
Contexte	**C**
Message	M1, M2, M3, M4
Liste d'attributs	L, L1, L2
Requête	R
Clé de Chiffrement et Déchiffrement SE	K
Chiffré SE	C2, C3
Clé Maîtresse ABE	MK
Clé de Chiffrement ABE	PK
Clé de Déchiffrement ABE d'un Véhicule	SK_v
Clé de Déchiffrement ABE du Centre de Stockage	SK_{cs}
Clé de Déchiffrement ABE d'une Partie Prenante	SK_{pp}
Chiffré ABE	C1, CD
Clé Maîtresse GS	SIG_MK
Clé de Vérification GS	SIG_PK
Clé de Signature GS d'un Véhicule	SIG_SK_v
Clé de Signature GS du Centre de Stockage	SIG_SK_{cs}
Clé de Signature GS d'une Partie Prenante	SIG_SK_{pp}
Signature GS	S1, S2, S3, SD

3.2.2.1 Génération et distribution des clés

Cette première étape a pour objectif de générer et de déployer de manière sécurisée les clés des entités légitimes (véhicules, parties prenantes et centre de stockage). Tout d'abord, l'autorité de confiance crée la clé maîtresse et la clé de chiffrement ABE puis la clé maîtresse et la clé de vérification GS. La clé de chiffrement ABE et la clé de vérification GS sont rendues publiques tandis que les clés maîtresses sont gardées secrètes. L'autorité réalise ainsi les opérations suivantes :

$$\mathrm{MK} = \textbf{abe\_setup}(1^{\lambda}) \tag{3.1}$$

$$\mathrm{PK} = \textbf{abe\_pkgen}(\mathrm{MK}) \tag{3.2}$$

$$\mathrm{SIG\_MK} = \textbf{gs\_setup}(1^{\lambda}) \tag{3.3}$$

$$\mathrm{SIG\_PK} = \textbf{gs\_pkgen}(\mathrm{SIG\_MK}) \tag{3.4}$$

Puis l'autorité de confiance génère une clé de déchiffrement ABE et une clé de signature GS pour chaque entité. Pour ce faire, l'autorité de confiance procède de la manière suivante pour toutes les entités légitimes. Pour une entité e, elle définit son identité ID_e. Ensuite, elle génère l'arbre d'accès A_e en utilisant **get_access_tree**, puis elle génère la clé de déchiffrement SK_e et la clé de signature $\mathrm{SIG\_SK}_e$ pour cette entité. Ces clés doivent être gardées secrètes. Pour cette génération, l'autorité réalise les opérations suivantes :

$$\mathrm{A}_e = \textbf{get\_access\_tree}(\mathrm{ID}_e) \tag{3.5}$$

$$\mathrm{SK}_e = \textbf{abe\_skgen}(\mathrm{A}_e, \mathrm{MK}) \tag{3.6}$$

$$\mathrm{SIG\_SK}_e = \textbf{gs\_skgen}(\mathrm{ID}_e, \mathrm{SIG\_MK}) \tag{3.7}$$

Ces clés sont supposées être envoyées à l'entité correspondante par un canal sécurisé. En particulier, pour un véhicule, les clés peuvent être déployées dans un module de sécurité matériel pendant le processus de fabrication. Une partie prenante peut s'enregistrer auprès de l'autorité de confiance à tout moment, en fournissant son identité ID_{pp}. Évidemment, il est supposé que l'autorité de confiance vérifie que la partie prenante a le droit de s'enregistrer avec cette identité. Notons qu'une partie prenante dispose du même matériel cryptographique qu'un véhicule car elle doit également vérifier si l'entité avec laquelle elle communique est enregistrée dans le système.

3.2.2.2 Scénario d'envoi sécurisé des données du véhicule (S1)

Ce scénario illustre l'envoi de données par un véhicule au centre de stockage. Les échanges sont signés afin que le véhicule et le centre de stockage puissent s'authentifier mutuellement. Une clé de chiffrement symétrique est choisie par le véhicule pour éviter l'utilisation systématique d'ABE ce qui permet de réduire le coût du protocole. Les données à stocker sont chiffrées en utilisant l'ensemble d'attributs L2 permettant au véhicule émetteur et aux parties prenantes autorisées de déchiffrer

le message. De plus, des nonces sont également utilisés pour empêcher les attaques par rejeu. Les détails de ce scénario sont présentés dans ce qui suit ainsi qu'en Figure 3.1.

① Un véhicule initie une demande de connexion. Cette demande contient une clé symétrique K et un nonce N_v (généré à l'aide de la fonction **random**). La demande est chiffrée en tant que C1 à l'aide de l'ensemble d'attributs L1 permettant uniquement au centre de stockage de déchiffrer C1 (c'est-à-dire la génération de L1 à l'aide de la fonction **get_attrs** avec un contexte vide et l'identité du centre de stockage). Cette demande chiffrée est signée en tant que S1 en utilisant la clé de signature du véhicule SIG_SK_v. Ensuite, C1 et S1 sont envoyés au centre de stockage en tant que message M1.

② Le centre de stockage reçoit le message M1. Tout d'abord, il vérifie la signature S1 en utilisant la clé publique de vérification SIG_PK. Si la signature est invalide, le centre de stockage réinitialise la communication. Sinon, après avoir extrait la requête chiffrée C1 du message M1 (en utilisant la fonction **get_msg**), il déchiffre C1 à l'aide de sa clé de déchiffrement SK_{cs}. Il obtient la clé symétrique K et le nonce N_v envoyé par le véhicule. Le centre de stockage génère une réponse de connexion contenant le nonce N_v et un nouveau nonce N_{cs}. Cette réponse est chiffrée en tant que C2 à l'aide de K et signée en tant que S2 à l'aide de la clé de signature SIG_SK_{cs}. Ensuite, C2 et S2 sont envoyés au véhicule en tant que message M2.

③ Le véhicule reçoit le message M2. Tout d'abord, il vérifie la signature à l'aide de la clé publique de vérification SIG_PK. Si la signature est invalide, le véhicule réinitialise la communication. Sinon, il extrait la réponse chiffrée C2 et la déchiffre en utilisant la clé symétrique K. Il obtient les deux nonces et vérifie que le nonce N_v a été retourné correctement. Ensuite, il récupère les données **D** et le contexte **C** (en utilisant la fonction **retreive_data**). Le contexte et l'identifiant du véhicule sont utilisés pour dériver l'ensemble d'attributs L2 à l'aide de la fonction **get_attrs**. Les données sont chiffrées en tant que C3 en utilisant cet ensemble, puis chiffrées à nouveau, avec N_{cs}, en utilisant K. De cette façon, le centre de stockage sera en mesure de déchiffrer le message afin de récupérer les données chiffrées. Le véhicule signe le chiffré en tant que S3 en utilisant sa clé de signature. À ce stade, le véhicule a envoyé les données avec succès. Enfin, C3 et S3 sont envoyés au centre de stockage en tant que message M3.

④ Le centre de stockage reçoit le message M3. De la même manière, il vérifie la signature. Il extrait le message et utilise K pour le déchiffrement. Il obtient les données chiffrées et un nonce. Si le nonce a la même valeur que celui qu'il a généré lors de l'étape ② alors il enregistre les données chiffrées dans sa base de données (en utilisant la fonction **db_store**).

3.2.2.3 Scénario de lecture sécurisée des données (S2 et S3)

Les scénarios **S2** et **S3** ne diffèrent qu'en ce qui concerne l'entité qui veut lire les données : une partie prenante (**S2**) ou un véhicule (**S3**). Par conséquent, dans ce qui suit, seul le scénario **S2** est considéré. Ce scénario décrit la récupération

d'une donnée par une partie prenante dans le centre de stockage. Notons que le langage de requête utilisé pour récupérer les données n'entre pas dans le cadre de ces travaux. Les détails de ce scénario sont présentés dans ce qui suit et illustrés dans la Figure 3.2.

Ce scénario utilise les mêmes techniques que le scénario **S1** pour empêcher les attaques par rejeu et assurer l'authentification, les étapes ①, ② et le début de l'étape ③ sont similaires au scénario **S1**. À la fin de l'étape ③, la partie prenante génère sa requête R (en utilisant la fonction **gen_query**), chiffre cette requête avec la clé K et envoie la requête chiffrée au centre de stockage avec sa signature.

④ Le centre de stockage reçoit le message M3. Il vérifie la signature, extrait le message et utilise K pour le déchiffrement. Il obtient la requête et un nonce. Il vérifie la valeur du nonce, qui doit être identique à celui qu'il a généré lors de l'étape ②. Puis il interroge sa base de données (en utilisant la fonction **db_read**). Le résultat est signé et envoyé à la partie prenante en tant que message M4.

⑤ La partie prenante reçoit le message M4. Elle vérifie la signature, extrait le message et utilise SK_{pp} pour le déchiffrement. Elle obtient le résultat de sa requête. À ce moment, la partie prenante peut lire les données avec succès.

Les schémas cryptographiques ayant été définis ainsi que les scénarios d'utilisation du protocole, la section suivante apporte la vérification formelle des propriétés de sécurité du protocole.

3.3 Modélisation et vérification formelle

3.3.1 Outil ProVerif

Plusieurs études ont montré l'existence de faiblesses dans des protocoles définis formellement (e.g. [Kremer 2005] et [Chen 2010]), ainsi une description formelle d'un protocole est une première bonne étape mais pas suffisante en soi. Les véhicules connectés ayant une longue durée de vie, il est nécessaire de vérifier formellement les propriétés de sécurité pour assurer la viabilité du protocole proposé sur le long terme.

Plusieurs outils de vérification automatique peuvent être utilisés pour vérifier formellement les propriétés de sécurité d'un protocole (AVISPA [Armando 2005], ProVerif [Blanchet 2021], YAPA [Baudet 2013], TAMARIN [Meier 2013]). ProVerif a été choisi car il est stable, mature, toujours maintenu, adopté avec réussite [Blanchet 2017, Blanchet 2008] et il supporte un nombre quelconque de sessions (ce qui est utile pour éviter d'imposer un nombre de véhicules et de parties prenantes). De plus, les primitives cryptographiques peuvent être représentées par une théorie équationnelle ou des règles de réécriture, cela permet d'établir une preuve pour des familles de primitives (c'est-à-dire tant que la primitive choisie respecte les équations).

En considérant le modèle d'attaquant discuté dans la Section 2.2, ProVerif est adapté car il permet de représenter un canal public et il utilise le modèle de Dolev-Yao [Dolev 1983]. Certains éléments de syntaxes de ProVerif sont représentés dans

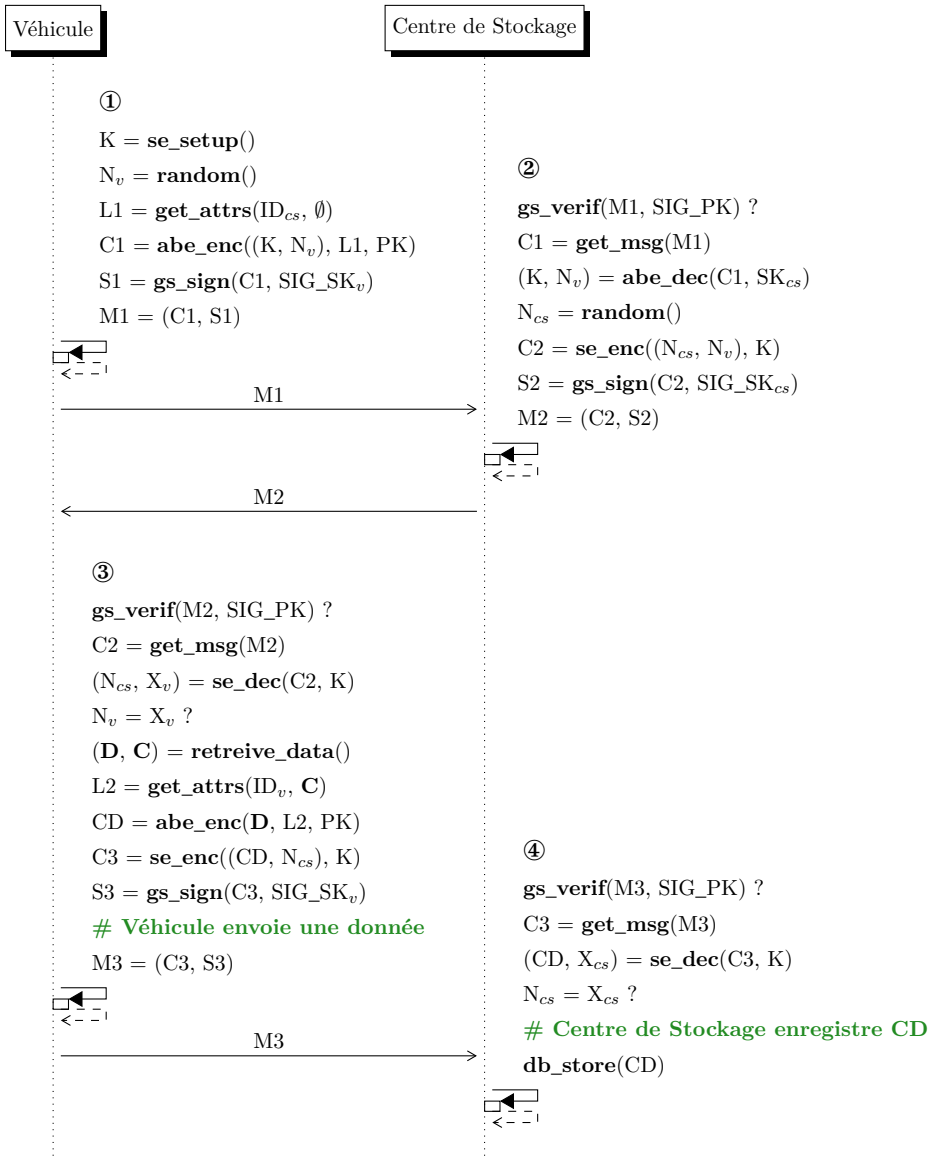

FIGURE 3.1 – Diagramme de séquence d'envoi sécurisé des données d'un véhicule

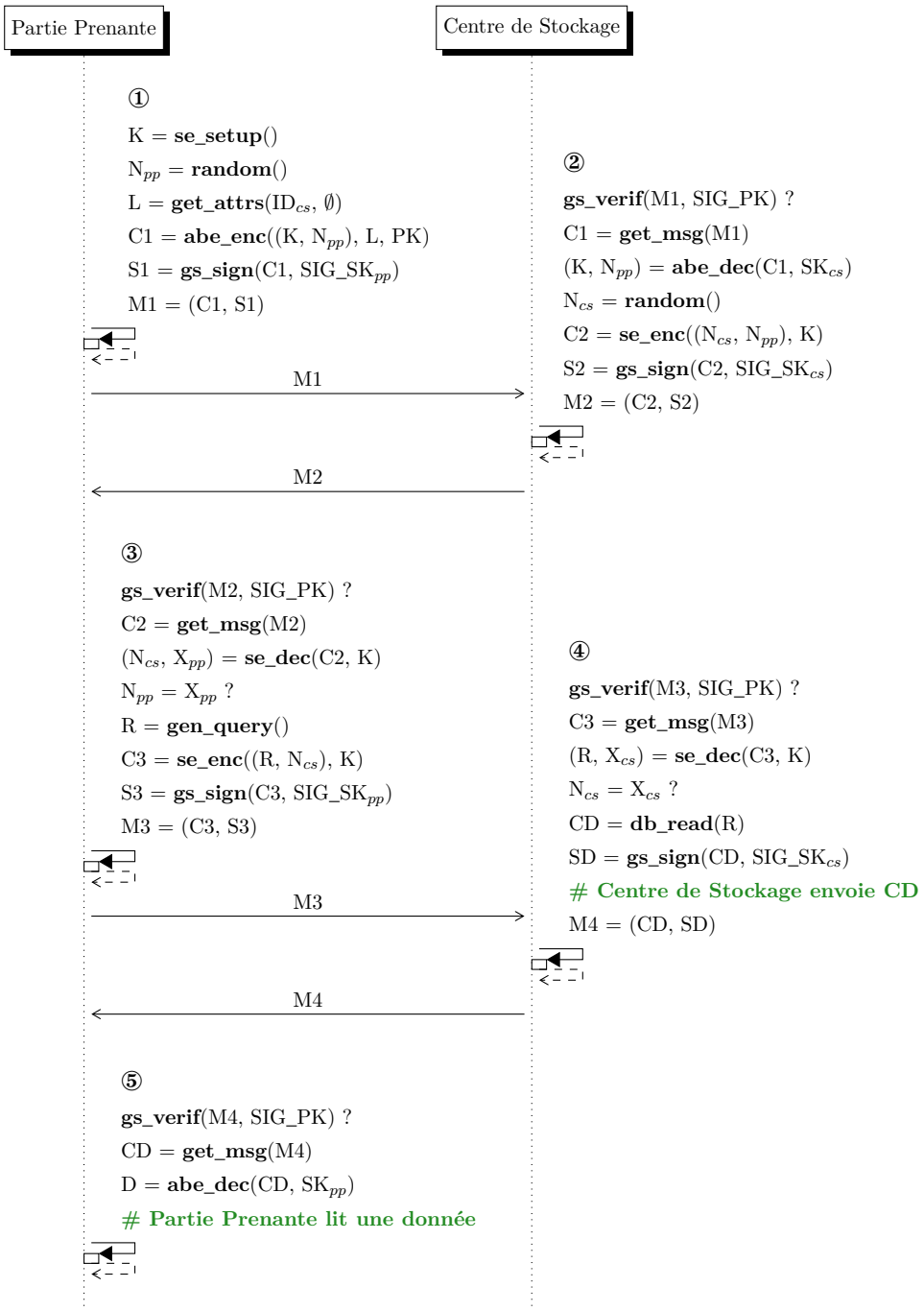

FIGURE 3.2 – Diagramme de séquence de lecture sécurisée des données par une partie prenante

TABLE 3.2 – Syntaxe des termes ProVerif

Termes	M, N : :=
Noms	a, b, c, k, m, n, s
Variables	x, y, z
Tuple	$(M_1, ..., M_k)$
Constructeur/Destructeur	$h(M_1, ..., M_k)$
Égalité de terme	M = N
Inégalité de terme	M <> N
Conjonction	M && N
Disjonction	M || N
Négation	**not**(M)

la Table 3.2 et la Table 3.3, ces tables sont issues de la documentation officielle de ProVerif.

La syntaxe des processus ProVerif permet de représenter un protocole selon des phases. Les *phases* permettent de décomposer un protocole en plusieurs parties pour simplifier sa représentation, elles peuvent être utilisées dans la formulation des propriétés vérifiées par ProVerif. Notre modélisation comporte quatre phases : 1) la première phase correspond à la génération et à la distribution des clés ; 2) la seconde phase correspond à la génération des verrous (utilisés dans la représentation ProVerif d'ABE) ; 3) la troisième phase correspond à la divulgation des paramètres publiques (les clés publiques, les attributs, arbres d'accès et verrous générés), à la génération de clés de déchiffrement ABE auxquelles l'attaquant peut accéder et aux fuites des clés d'un véhicule, du centre de stockage, et d'une partie prenante lorsqu'elles ont lieu ; 4) la quatrième et dernière phase correspond aux scénarios (S1, S2 et S3) du protocole.

ProVerif vérifie les propriétés des protocoles en s'appuyant sur deux notions : les événements et les requêtes. Les *événements* représentent le fait qu'une étape particulière d'un protocole a été atteinte. Pour les distinguer, si nécessaire, un événement peut avoir des paramètres. Par exemple, un événement généré lors de l'exécution de la partie véhicule du protocole peut avoir comme paramètre l'identité de ce véhicule. Les *requêtes* sont des propriétés élémentaires vérifiées par ProVerif. La vérification des requêtes peut entraîner plusieurs sorties. La mention *true* est renvoyée lorsqu'une requête est vérifiée. La mention *false* est renvoyée lorsqu'une requête est insatisfaite, ProVerif renvoie également une trace d'exécution illustrant comment la propriété est insatisfaite. La mention *cannot be proved* est renvoyée lorsque ProVerif n'arrive pas à déterminer si une requête est vraie et si une trace ne peut pas être trouvée prouvant que la requête est fausse, cependant nous verrons plus tard que cette sortie ne se produit pas dans notre cas. Les requêtes ProVerif utilisées pour prouver les propriétés de sécurité reposent sur cinq constructions

TABLE 3.3 – Syntaxe des processus ProVerif

Processus	P, Q : :=
Processus nulle	0
Composition parallèle	P | Q
Réplication	!P
Restriction de nom	**new** n : t ; P
Message en entrée	**in**(M, x : t) ; P
Message en sortie	**out**(M, N) ; P
Conditionnel	**if** M **then** P **else** Q
Évaluation de terme	**let** x = M **in** P **else** Q
Utilisation de macro	R(M_1, ..., M_k)
Changement de phase	**phase** t ; P

principales :

- `not (event X)` : le résultat de cette requête est vrai si l'événement `X` n'est jamais généré dans le protocole. Sinon, il peut être généré et ProVerif fournit une trace de son exécution. Ce type de requête est utile pour vérifier que toutes les parties intéressantes du protocole sont atteignables ;
- `not attacker.pN(V)` : le résultat de cette requête est vrai si l'attaquant n'a pas la possibilité de connaître la valeur `V`. L'élément `N` est utilisé pour indiquer une phase du protocole ;
- `secret V1(,Vi)*` : le résultat de cette requête est vrai si toutes les valeurs de `Vi` sont gardées secrètes. Cette requête est proche de la précédente. Elle est utilisée lorsque les valeurs `Vi` sont des valeurs internes d'un processus ProVerif ;
- `inj-event(X1) ==> inj-event(X2)` : le résultat de cette requête est vrai si chaque événement `X2` correspond à un événement distinct et précédant `X1` ;
- `S1 (&& Si)* ==> C1 (|| Ci)*` : le résultat de cette requête est vraie si l'une des conditions `Ci` est vraie lorsque toutes les sous-requêtes `Si` sont vraies.

La vérification des requêtes par ProVerif est réalisée en trois étapes (algorithme de *Resolution with selection*) : 1) traduction de la description du protocole en clauses de Horn, 2) saturation des clauses de Horn en cherchant un point-fixe et 3) vérification des requêtes vis-à-vis des clauses saturées.

3.3.2 Modélisation ProVerif d'ABE

Pour faciliter la compréhension de la modélisation du protocole dans ProVerif, cette section présente les adaptations réalisées pour modéliser ABE, puis la modélisation d'ABE dans ProVerif ainsi qu'un exemple illustrant son utilisation dans un protocole simple.

3.3.2.1 Principe

L'utilisation d'ABE peut amener à employer un nombre quelconque d'attributs. Une façon *naïve* de représenter cette possibilité dans ProVerif passe par l'usage d'une liste non bornée et non ordonnée (i.e., une structure de données récursive), avec un système de réécriture qui permet de décider l'égalité entre deux listes. Un arbre d'accès serait alors représenté par un ensemble de listes et vérifier que l'arbre de contrôle d'accès est satisfait par une liste d'attributs serait alors ramené à une égalité entre listes non ordonnées. Toutefois, cette stratégie ne permet pas à l'algorithme de ProVerif d'atteindre un point-fixe étant donné qu'il est toujours possible, par construction, de créer une liste arbitrairement longue. L'implémentation des attributs dans ProVerif doit donc être adaptée. En pratique, le nombre d'attributs et de listes d'attributs utilisés pour une trace finie est nécessairement fini. Raisonner, au niveau d'une trace, avec des ensembles finis permettrait à l'algorithme interne de ProVerif de converger. Cela impose alors, durant une phase préliminaire de ProVerif, de générer ces éléments (attributs et listes d'attributs) en nombre finis (mais quelconque), avant d'enchaîner avec le protocole lui-même.

Comme nous l'avons vu précédemment, ABE est un schéma de chiffrement permettant à une clé associée à un arbre d'accès de déchiffrer des chiffrés associés à une liste d'attributs permettant de satisfaire l'arbre. Dans notre modélisation, les listes d'attributs (resp. les arbres d'accès) sont abstraits avec un type élémentaire, non-récursif. L'association entre un arbre d'accès et les listes qu'il valide est représentée par des couples nommés dans la suite *verrous*.

Un *verrou* est donc un couple contenant un arbre d'accès et une liste d'attributs, ce couple signifie que l'arbre d'accès est satisfait par la liste d'attributs. Nous considérons que cette abstraction représente le comportement d'ABE, en effet plusieurs verrous peuvent associer le même arbre d'accès à des listes d'attributs différentes, illustrant ainsi qu'une clé de déchiffrement peut permettre de déchiffrer des chiffrés contenant différentes listes d'attributs. Inversement, plusieurs verrous peuvent associer la même liste d'attributs à des arbres d'accès différents, illustrant ainsi qu'un chiffré peut être déchiffré par des clés de déchiffrement contenant différents arbres d'accès.

Ainsi les listes d'attributs, les arbres d'accès et les verrous sont générés lors d'une première phase de génération et distribution des clés dans la modélisation ProVerif du protocole. Par la même occasion, les clés de déchiffrement ABE et les arbres d'accès sont distribués aux différentes entités du système, ce qui leur confère une capacité de déchiffrement en accord avec les verrous. Le centre de stockage possédant une clé associée uniquement à son attribut, un unique verrou est généré associant son arbre d'accès à cet attribut.

3.3.2.2 Modélisation ProVerif

Types Les types représentent les différents types d'objets cryptographiques manipulés dans ABE :

— *attrs* représente les listes d'attributs ;
— *accessp* représente les arbres d'accès ;
— *abe_mkey* représente les clés maîtresses ABE ;
— *abe_skey* représente les clés de déchiffrements ABE ;
— *abe_pkey* représente les clés publiques de chiffrements ABE.

```
type attrs.
type accessp.
type abe_mkey.
type abe_skey.
type abe_pkey.
```

Fonctions Les fonctions représentent principalement les algorithmes des schémas ABE :

— *ext_attrs* renvoie l'union des deux ensembles d'attributs donnés en entrée ;
— *abe_pkgen* génère une clé publique de chiffrement à partir d'une clé maîtresse ABE ;
— *abe_skgen* génère une clé de déchiffrement ABE à partir d'un arbre d'accès et d'une clé maîtresse ABE ;
— *abe_enc* renvoie un chiffré pour un message, une liste d'attributs et une clé de chiffrement ABE donnés ;
— *abe_lock* renvoie un verrou liant un arbre d'accès à une liste d'attributs.

Notons que la fonction *abe_lock* est déclarée privée. Dans ProVerif, les fonctions non privées sont connues de l'attaquant, ainsi si la fonction *abe_lock* n'était pas déclarée privée, l'attaquant serait en mesure de créer ses propres verrous ce qui pourrait lui permettre de déchiffrer par lui-même n'importe quel chiffré. La fonction étant déclarée privée, seule l'autorité peut définir les verrous et spécifier les ensembles d'attributs permettant de satisfaire les arbres d'accès.

```
fun ext_attrs(attrs, attrs): attrs.
fun abe_pkgen(abe_mkey): abe_pkey.
fun abe_skgen(accessp, abe_mkey): abe_skey.
fun abe_enc(bitstring, attrs, abe_pkey): bitstring.
fun abe_lock(accessp, attrs): bitstring [private].
```

Noms libres Les noms libres sont utilisés pour représenter la clé maîtresse ABE et le canal de communication :

— *abe_mk* représente une clé maîtresse ABE, cette clé est gardée secrète, elle ne doit pas être accessible à l'attaquant ;
— *c* représente un canal de communication public sur lequel les messages sont envoyés, tous les messages envoyés sur ce canal sont accessibles à l'attaquant.

```
free abe_mk: abe_mkey [private].
free c: channel.
```

Réductions Les réductions expriment essentiellement les relations entre les algorithmes ABE :

— *abe_attrs* renvoie l'ensemble d'attributs d'un chiffré valide, c'est-à-dire d'un chiffré généré à partir d'un message, d'une liste d'attributs et d'une clé publique générée via une clé maîtresse ABE ;

— *abe_dec* réalise le déchiffrement ABE en renvoyant le message contenu dans le chiffré ABE, si : 1) un chiffré ABE valide est donné en entrée ; 2) une clé de déchiffrement ABE valide est donnée en entrée, c'est-à-dire une clé générée avec la même clé maîtresse ABE que celle utilisée pour la génération de la clé publique utilisée lors du chiffrement ; 3) un verrou valide est donné en entrée, c'est-à-dire un verrou généré avec le même arbre d'accès que la clé de déchiffrement, et la même liste d'attributs que le chiffré.

```
reduc forall m: bitstring, mk: abe_mkey, at: attrs ;
  abe_attrs(abe_enc(m, at, abe_pkgen(mk))) = at.

reduc forall m: bitstring, mk: abe_mkey, at: attrs, sk: abe_skey,
    ap: accessp ;
  abe_dec(abe_enc(m, at, abe_pkgen(mk)),
          abe_skgen(ap, mk),
          abe_lock(ap, at)) = m.
```

Tables Les tables permettent de stocker des informations, elles ne sont pas accessibles à l'attaquant :

— *list_attrs* stocke les listes d'attributs considérées dans le protocole ;

— *list_locks* contient les verrous définis par l'autorité, indexés par l'arbre d'accès et l'ensemble d'attributs utilisés pour générer le verrou.

Notons que ces tables ne contiennent pas d'informations considérées confidentielles, leurs contenus devront être rendus publics durant l'exécution du protocole.

```
table list_attrs(attrs).
table list_locks(accessp, attrs, bitstring).
```

3.3.2.3 Exemple d'un protocole ABE simple

Dans cette section nous présentons un exemple simple, détaché du contexte de cette thèse, afin d'illustrer l'utilisation de la modélisation ProVerif d'ABE décrite précédemment.

Cas d'étude Considérons qu'Alice envoie un message chiffré à Bob en utilisant ABE. Une autorité de confiance génère les listes d'attributs utilisées dans le protocole, la clé de déchiffrement ABE de Bob et les verrous associant l'arbre d'accès de Bob aux listes d'attributs. La sécurité de ce protocole est évaluée en considérant les propriétés suivantes : les messages reçus par Bob doivent être ceux qu'Alice a envoyés, la clé maîtresse ABE doit être secrète et les messages échangés entre Alice et Bob doivent être confidentiels.

Évènements Comme nous l'avons vu précédemment, ProVerif vérifie la sécurité d'un protocole en utilisant des requêtes qui peuvent s'appuyer sur des événements. Les événements considérés dans cet exemple sont :

— *alice_send*, généré lorsque Alice envoie un message ;

— *bob_read*, généré lorsque Bob déchiffre un message avec succès.

```
1 event alice_send(bitstring, bitstring).
2 event bob_read(bitstring, bitstring).
```

Requêtes Les requêtes considérées sont les suivantes :

— *Q1* vérifie que l'événement *alice_send* peut être généré par le protocole, ProVerif renvoie une trace d'exécution le cas échéant ;

— *Q2* est similaire à *Q1* pour la génération de l'événement *bob_read* ;

— *Q3* vérifie que lorsque Bob réussit à lire un message, alors ce message a été émis par Alice ;

— *Q4* vérifie que l'attaquant ne possède pas la clé maîtresse ABE ;

— *Q5* vérifie que le message émis par Alice est secret, c'est-à-dire inconnu de l'attaquant ;

— *Q6* vérifie que le message reçu par Bob est secret également.

```
1  (*Q1*)  query ct: bitstring, msg: bitstring ;
2              event (alice_send(ct, msg)).
3  (*Q2*)  query ct: bitstring, msg: bitstring ;
4              event (bob_read(ct, msg)).
5  (*Q3*)  query ct: bitstring, msg: bitstring ;
6              event (bob_read(ct, msg))
7                  ==> event (alice_send(ct, msg)).
8  (*Q4*)  query attacker(abe_mk).
9  (*Q5*)  query secret alice_msg.
10 (*Q6*)  query secret bob_msg.
```

Processus Les processus représentent une séquence d'opérations. Ils peuvent contenir une barre verticale séparant deux sous-processus, signifiant que ces sous-processus peuvent être exécutés en parallèle. Ce qui suit est une description des processus représentant le protocole de l'exemple.

Le processus principal *process* exécute tous les processus du système.

```
process
    create_list_attrs() | create_alice()
  | create_bob() | do_public_leak()
```

Le processus *create_list_attrs* est exécuté par l'autorité, il génère les listes d'attributs utilisées dans le protocole et les insère dans la table des listes d'attributs. Le point d'exclamation représente un point de réplication signifiant que les opérations entre parenthèses peuvent être répétées.

```
let create_list_attrs() =
  !(
    new a: attrs ;
    insert list_attrs(a)
  ).
```

Le processus *create_alice* est exécuté par l'autorité, il génère la clé publique de chiffrement ABE et la transmet au processus *alice*.

```
let create_alice() =
  let abe_pk = abe_pkgen(abe_mk) in
  alice(abe_pk).
```

Le processus *alice* est exécuté par Alice, il génère un message aléatoire, récupère une liste d'attributs, chiffre le message en utilisant la fonction *abe_enc*, génère l'événement *alice_send* signifiant que le message a été envoyé, puis envoie le chiffré sur le canal public.

```
let alice(abe_pk: abe_pkey) =
  !(
    new alice_msg: bitstring ;
    get list_attrs(a) in
    let ct = abe_enc(alice_msg, a, abe_pk) in
    event alice_send(ct, alice_msg) ;
    out(c, ct)
  ).
```

Le processus *create_bob* est exécuté par l'autorité, il génère un arbre d'accès aléatoire puis une clé de déchiffrement ABE associée à cet arbre. Ensuite, ce processus peut associer plusieurs ensembles d'attributs à l'arbre d'accès de Bob, et il enregistre les verrous générés dans la table des verrous. Ce processus transmet l'arbre d'accès et la clé de déchiffrement ABE au processus *bob*.

```
let create_bob() =
  new bob_ap: accessp ;
  let bob_sk = abe_skgen(bob_ap, abe_mk) in
  !(
    get list_attrs(a) in
    insert list_locks(bob_ap, a, abe_lock(bob_ap, a))
  ) | (
    bob(bob_ap, bob_sk)
  ).
```

Le processus *bob* est exécuté par Bob, il récupère un message sur le canal public, extrait les attributs du chiffré, puis récupère de la table des verrous le verrou correspondant à l'arbre d'accès de Bob et à l'ensemble d'attributs du chiffré. Puis il déchiffre le message et émet l'évènement *bob_read* si le déchiffrement est un succès.

```
let bob(bob_ap: accessp, bob_sk: abe_skey) =
  !(
    in(c, ct: bitstring) ;
    let at = abe_attrs(ct) in
    get list_locks(=bob_ap, =at, l) in
    let bob_msg = abe_dec(ct, bob_sk, l) in
    event bob_read(ct, bob_msg)
  ).
```

Le processus *do_public_leak* est exécuté par l'autorité, il rend public la clé publique ABE, les listes d'attributs, les arbres d'accès et les verrous stockés dans *list_locks*, les listes d'attributs stockées dans *list_attrs*, ainsi que des arbres d'accès et la clé de déchiffrement ABE associée pour des arbres générés de manière aléatoire. Toutes ces informations sont donc accessibles à l'attaquant.

```
let do_public_leak() =
  out(c, abe_pkgen(abe_mk)) ;
  !(
    get list_locks(ap, at, l) in
    out(c, ap) ;
    out(c, at) ;
    out(c, l)
  ) | !(
    get list_attrs(a) in
    out(c, a)
  ) | !(
    new ap: accessp ;
    out(c, ap) ;
    out(c, abe_skgen(ap, abe_mk))
  ).
```

3.3.2.4 Vérification des requêtes de l'exemple

La vérification des requêtes en utilisant ProVerif renvoie le résultat suivant.

Les requêtes *Q1* et *Q2* sont insatisfaites, ce résultat est celui attendu puisque ProVerif arrive à générer les événements correspondants. La trace ProVerif suite à l'insatisfaction de la première requête est donnée à titre d'exemple en Figure 3.3.

La requête *Q3* est insatisfaite, en effet l'attaquant peut générer un message, le chiffrer en utilisant la clé publique ABE et l'envoyer à Bob, ainsi tous les événements *alice_send* ne précèdent pas les événements *bob_read*. La trace ProVerif correspondante est donnée en Figure 3.4. Pour résoudre ce problème, Bob doit pouvoir être en mesure de vérifier l'identité de l'émetteur en utilisant par exemple, un mécanisme de signature, ce qui empêcherait l'attaquant d'envoyer un message qu'il a forgé.

La requête *Q4* est vérifiée, ce qui est attendu étant donné que la clé maîtresse ABE ne fuite à aucun moment.

La requête *Q5* est vérifiée, cela prouve que l'attaquant n'est pas en mesure de lire le contenu d'un message envoyé par Alice.

En revanche, la requête *Q6* est insatisfaite, cela ne contredit pas le résultat de l'évaluation de la requête *Q5*. En effet les messages émis par Alice ne peuvent pas être lus par l'attaquant. Cependant, l'attaquant peut forger un message et l'envoyer à Bob. L'attaquant connaît donc le message reçu par Bob ce qui invalide cette requête. La trace ProVerif correspondante est similaire à celle donnée en Figure 3.4.

```
Verification summary:

Q1: Query not event(alice_send(ct_2,msg)) is false.

Q2: Query not event(bob_read(ct_2,msg)) is false.

Q3: Query event(bob_read(ct_2,msg))
    ==> event(alice_send(ct_2,msg)) is false.

Q4: Query not attacker(abe_mk[]) is true.

Q5: Query secret alice_msg is true.

Q6: Query secret bob_msg is false.
```

3.3.3 Modélisation ProVerif des scénarios du protocole

Revenons à présent sur le protocole que nous proposons dans nos travaux et sur sa modélisation ProVerif. Cette modélisation étant relativement longue, il serait fastidieux de la commenter entièrement dans ce manuscrit. Nous avons donc choisi de décrire et commenter uniquement une partie du scénario **S1** correspondant à la partie du processus du véhicule et du centre de stockage gérant le scénario **S1**. Pour les scénarios **S2** et **S3** nous décrivons uniquement des simplifications réalisées pour modéliser ces scénarios. Le reste de la modélisation ProVerif du protocole est disponible en annexe et sa compréhension ne présente pas plus de difficultés que la partie que nous présentons en détail dans cette section.

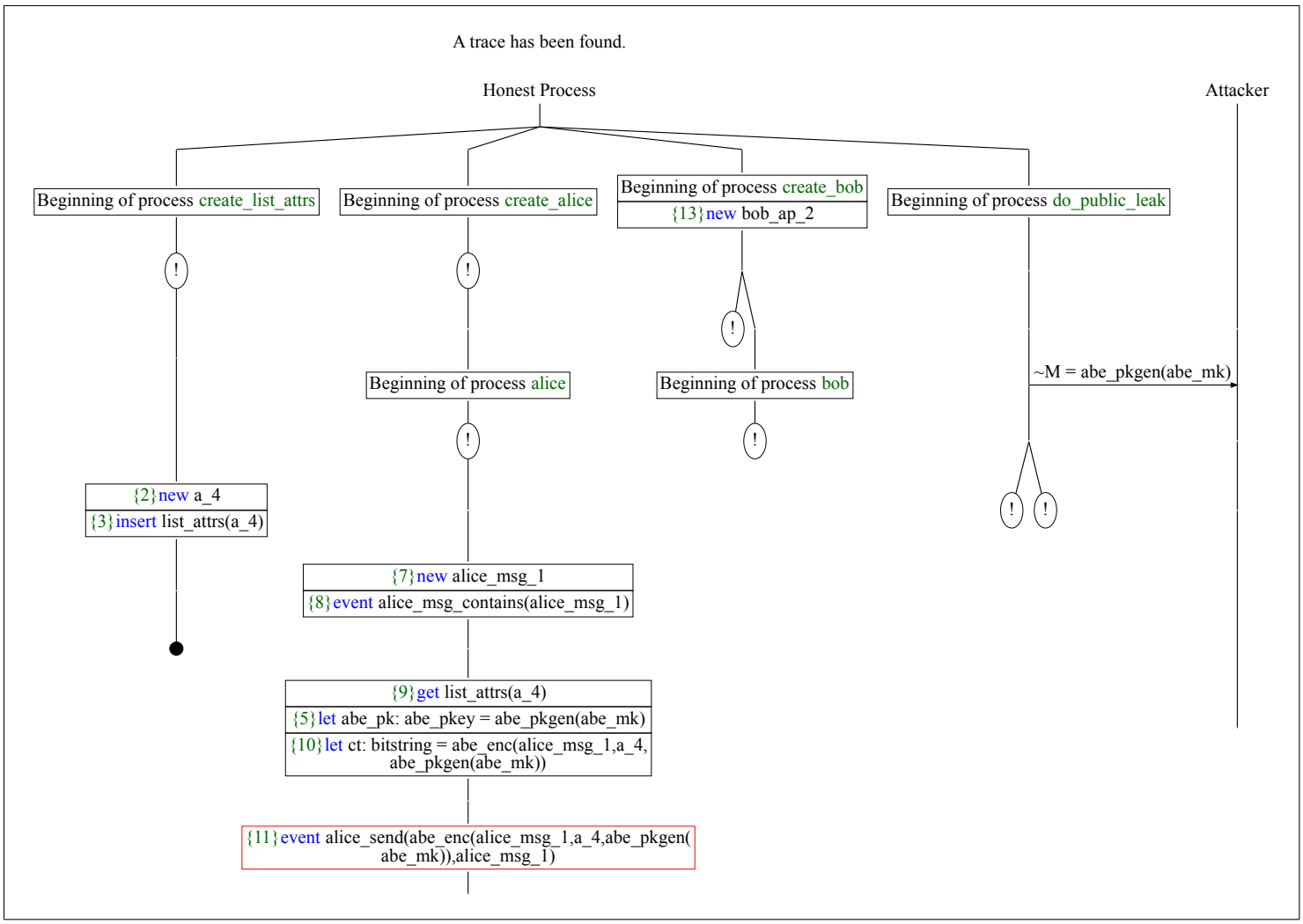

FIGURE 3.3 – Trace d'exécution ProVerif suite à la vérification de la requête Q1

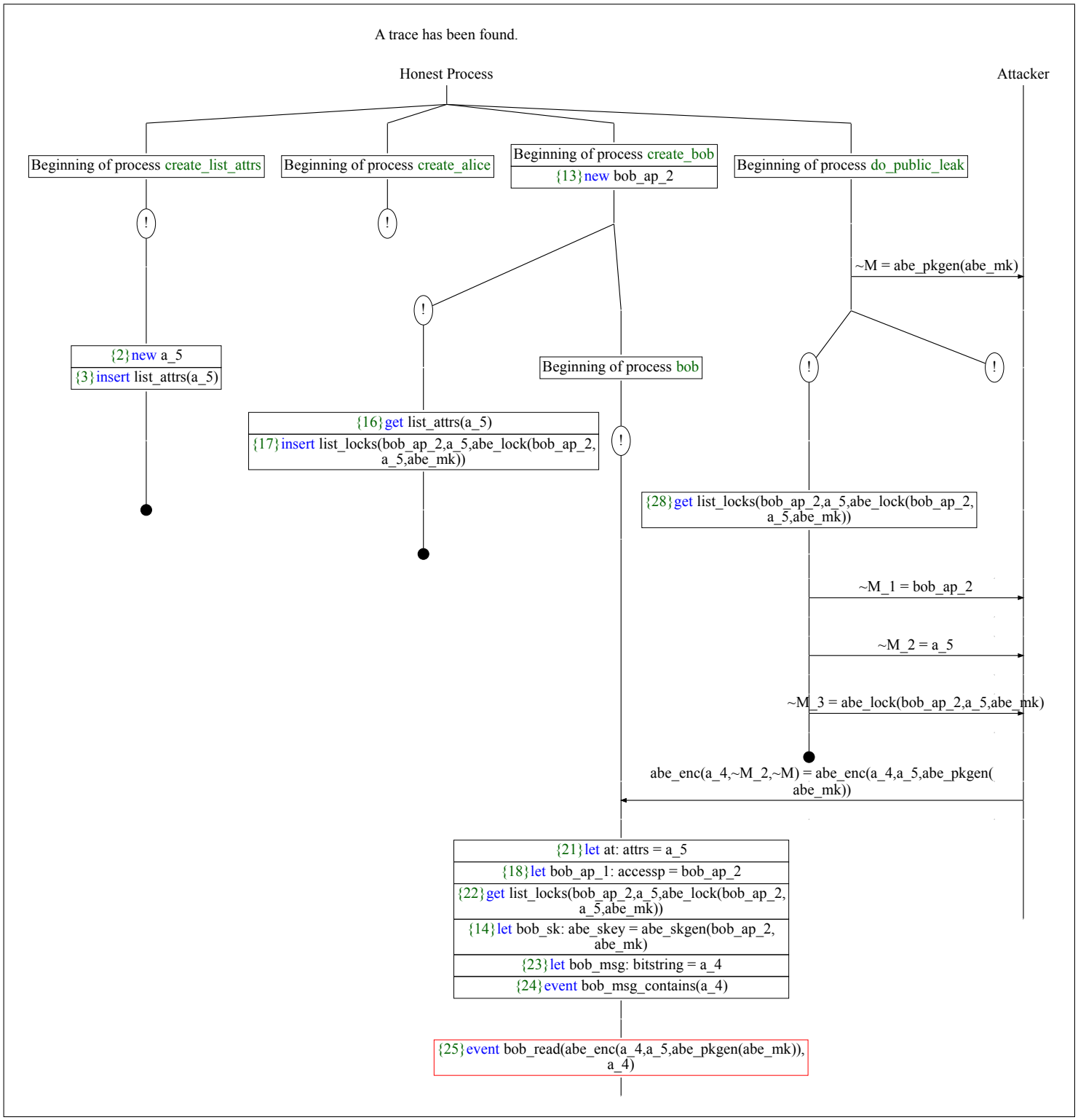

FIGURE 3.4 – Trace d'exécution ProVerif suite à la vérification de la requête Q3

Processus du véhicule Ce processus est représenté en Listing 3.1, il prend en entrée l'attribut, l'arbre d'accès, la clé de déchiffrement ABE et la clé de signature GS du véhicule, ainsi que la clé publique ABE et la clé publique GS. Ce matériel cryptographique est généré dans le processus parent par l'autorité de confiance. Le reste du processus correspond aux étapes 1 et 3 de la partie véhicule du scénario **S1** du protocole présentées en Section 3.2.2.2.

Étape 1 : une clé symétrique SE est générée aléatoirement, cela correspond à l'exécution de l'algorithme **se_setup** ; puis le véhicule génère un nonce aléatoirement. La clé symétrique SE et le nonce du véhicule sont chiffrés en utilisant **abe_enc** et l'attribut du centre de stockage. Le chiffré est ensuite signé en utilisant **gs_sign**, le résultat est envoyé sur le canal de communication public.

Étape 3 : un message est lu sur le canal de communication. La signature du message est vérifiée, puis le message est déchiffré en utilisant **se_dec** avec la clé symétrique SE précédemment envoyée. Le véhicule récupère le nonce du centre de stockage et un nonce qui doit être égal à celui qu'il a envoyé en étape 1. Puis une liste d'attributs est récupérée de la table des verrous en filtrant la table par l'arbre d'accès du véhicule. Un message est ensuite généré aléatoirement, ce qui correspond à l'exécution de **retreive_data**. Le message est chiffré avec la liste d'attributs récupérée en utilisant **abe_enc**. Le chiffré ABE résultat et le nonce du centre de stockage sont chiffrés en utilisant **se_enc** et signés en utilisant **gs_sign**. L'événement *vehicle_send* est ensuite généré, puis le chiffré signé est envoyé sur le canal de communication.

Processus du centre de stockage Ce processus est représenté en Listing 3.2, il prend en entrée la clé de déchiffrement ABE et la clé de signature GS du centre de stockage, ainsi que la clé publique GS. Ce matériel cryptographique est généré dans le processus parent par l'autorité de confiance. Le reste du processus correspond aux étapes 2 et 4 de la partie centre de stockage du scénario **S1** du protocole présentées en Section 3.2.2.2.

Étape 2 : un message est lu sur le canal de communication. Un verrou est récupéré de la table des verrous en filtrant la table par l'ensemble d'attributs et l'arbre d'accès du centre de stockage. La signature du message récupéré du canal de communication est vérifiée, puis le message est déchiffré en utilisant **abe_dec** avec la clé de déchiffrement ABE du centre de stockage et le verrou précédemment récupéré. Le centre de stockage récupère une clé symétrique SE et le nonce du véhicule. Il génère ensuite un nonce aléatoirement, puis chiffre le nonce du véhicule et son nonce en utilisant **se_enc** et la clé symétrique reçue, signe le chiffré résultant et envoie le chiffré signé sur le canal de communication.

Étape 4 : un message est récupéré du canal de communication. La signature du message est vérifiée, puis le message est déchiffré. Le centre de stockage récupère le chiffré ABE généré par le véhicule et un nonce qui doit être égal à celui envoyé en étape 1. Si le test d'égalité est un succès, l'événement *storage_write* est généré et le message est enregistré dans la table *list_msg*.

```
let handle_vehicle(
      va: attrs, vap: accessp,
      vehicle_abe_sk: abe_skey, vehicle_abe_pk: abe_pkey,
      vehicle_gs_sk: gs_skey, vehicle_gs_pk: gs_pkey) =
  !(
    (* Scenario S1 : envoi securise des donnees du vehicule *)

    (* Etape (1) *)
    new k: s_key ;
    new vehicle_nonce: bitstring ;
    let ct1 = abe_enc((s_key2bs(k), vehicle_nonce),
                      storage_attrs, vehicle_abe_pk) in
    out(c, gs_sign(ct1, vehicle_gs_sk)) ;

    (* Etape (3) *)
    in(c, response: bitstring) ;
    let (storage_nonce: bitstring, =vehicle_nonce)
        = s_dec(gs_msg(response, vehicle_gs_pk), k) in
    get list_locks(=vap, at, l) in
    new msg: bitstring ;
    let ct2 = abe_enc(msg, at, vehicle_abe_pk) in
    let ct3 = gs_sign(s_enc((ct2, storage_nonce), k),
                              vehicle_gs_sk) in
    event vehicle_send(va, at, ct2, msg) ;
    out(c, ct3)
  ) | !(
    (* Scenario S2 : lecture securisee des donnees *)
    ...
  ).
```

Listing 3.1 – Modélisation ProVerif du processus du véhicule

Modélisation des scénarios S2/S3 Pour simplifier le protocole, certains détails sont omis dans le modèle ProVerif : les étapes d'authentification (étapes 1, 2 et début de l'étape 3) précédant l'envoi de la requête au centre de stockage dans les scénarios S2 et S3 ne sont pas décrites dans ProVerif. En d'autres termes, les nonces et la clé symétrique ne sont pas utilisés pendant les scénarios de récupération de données. Ce choix a pour conséquence de rendre l'attaquant plus fort, mais il n'affecte pas la sécurité comme nous le verrons dans la prochaine section.

3.3.4 Vérification des propriétés du protocole

Cette section présente les événements du protocole sur lesquels s'appuient les requêtes ProVerif, et la vérification des propriétés de sécurité.

3.3.4.1 Évènements ProVerif

Dans le protocole, certains de ces événements correspondent aux commentaires en vert de la Figure 3.1 et de la Figure 3.2. Pendant la vérification du protocole, une fuite de clé de déchiffrement ABE est simulée pour analyser les possibilités de

```
let handle_storage(
      storage_abe_sk: abe_skey,
      storage_gs_sk: gs_skey,
      storage_gs_pk: gs_pkey) =
  !(
    (* Scenario S1 : envoi securise des donnees du vehicule *)

    (* Etape (2) *)
    in(c, store_request: bitstring) ;
    get list_locks(=storage_ap, =storage_attrs, l) in
    let (s_key2bs(k), vehicle_nonce: bitstring)
        = abe_dec(gs_msg(store_request, storage_gs_pk),
    storage_abe_sk, l) in
    new storage_nonce: bitstring ;
    out(c, gs_sign(s_enc((storage_nonce, vehicle_nonce), k),
    storage_gs_sk)) ;

    (* Etape (4) *)
    in(c, ct3: bitstring) ;
    let (ct2: bitstring, =storage_nonce)
        = s_dec(gs_msg(ct3, storage_gs_pk), k) in
    event storage_write(ct2) ;
    insert list_msg(ct2)
  ) | !(
    (* Scenario S2/S3 : lecture securisee des donnees *)
    ...
  ).
```

Listing 3.2 – Modélisation ProVerif du processus du centre de stockage

l'attaquant et des événements sont également générés lorsqu'une fuite de clé est forcée. La liste des événements considérés dans le protocole est la suivante :

- L'événement `vehicle_send` se produit lorsqu'un véhicule envoie, sur le canal public, un message chiffré contenant des données (voir l'étape ③ de la Figure 3.1) ;
- L'événement `storage_write` se produit lorsque le centre de stockage stocke un message chiffré dans sa base de données (voir l'étape ④ de la Figure 3.1) ;
- L'événement `storage_send` survient lorsque le centre de stockage envoie, sur le canal public, un message chiffré contenant des données (voir l'étape ④ de la Figure 3.2) ;
- L'événement `stakeholder_read` survient lorsqu'une partie prenante récupère et déchiffre avec succès un message contenant des données (voir l'étape ⑤ de la Figure 3.2) ;
- L'événement `vehicle_read` est identique à l'événement `stakeholder_read`, pour un véhicule ;
- L'événement `vehicle_leak` se produit lorsque la clé de déchiffrement ABE d'un véhicule est divulguée ;

— L'événement `storage_leak` se produit lorsque la clé de déchiffrement ABE du centre de stockage est divulguée ;

— L'événement `stakeholder_leak` se produit lorsque la clé de déchiffrement ABE d'une partie prenante est divulguée ;

— L'événement `stakeholder_attrs` se produit lorsque un verrou associé à l'arbre d'accès de la partie prenante et à une liste d'attributs est enregistré dans la table des verrous.

3.3.4.2 Vérification à l'aide de ProVerif

Cette sous-section présente d'abord les requêtes ProVerif considérées et leurs liens avec les propriétés, puis les scénarios de fuites envisagés, et enfin les résultats de la vérification en utilisant ProVerif.

La vérification des propriétés repose sur l'ensemble des requêtes ProVerif de la Table 3.4. Les requêtes **Q1** à **Q5** sont utilisées pour vérifier que le protocole est fonctionnel en assurant que tous les événements peuvent se produire. Les requêtes **Q6** à **Q10** garantissent le secret des clés maîtresses (pour l'autorité de confiance) et des clés de déchiffrement ABE (pour les véhicules, le centre de stockage et les parties prenantes). La requête **Q11** garantit que l'attaquant ne peut pas lire les données contenues dans un message. Les requêtes **Q12** et **Q13** (resp. **Q14** et **Q15**) garantissent que, si un véhicule (resp. une partie prenante) est capable de lire les données d'un message, alors ce message est nécessairement stocké dans le centre de stockage et ce message a été envoyé par un véhicule enregistré. La requête **Q16** garantit que, si le centre de stockage stocke un message, ce message a été envoyé par un véhicule enregistré. La requête **Q17** vérifie que, si un attaquant est capable de lire une donnée, cela implique nécessairement qu'une clé de déchiffrement ABE a été divulguée et que l'attaquant ne peut pas lire plus de données que cette clé le permet.

Ces requêtes participent ensemble à la vérification des propriétés présentées dans la Section 2.3. Les requêtes **Q12** à **Q15** vérifient que le destinataire légitime d'un message envoyé par un véhicule est en mesure de vérifier l'intégrité du message (**P1**). La requête **Q16** vérifie que le centre de stockage peut identifier si un message reçu provient d'un véhicule légitime (**P2**). Les requêtes **Q12** et **Q13** vérifient qu'un véhicule peut lire les données contenues dans les messages mais seulement ceux qu'il a envoyés (**P3**). Les requêtes **Q14** et **Q15** vérifient qu'une partie prenante peut seulement lire les messages pour lesquels elle a été autorisée à lire (**P4**).

Les propriétés **P5** et **P6** sont déduites de la requête **Q17**. Si la clé de déchiffrement ABE du centre de stockage a été divulguée, l'attaquant a les mêmes privilèges que le centre de stockage. Ainsi, en considérant le modèle de Dolev-Yao si l'attaquant n'est pas capable de lire les données, alors le centre de stockage n'est pas non plus capable de lire les données (**P5**). De plus, si l'attaquant est capable de lire un message, alors la clé de déchiffrement ABE du véhicule correspondant ou la clé de déchiffrement ABE d'une partie prenante capable de lire le message a été divulguée. Et donc

TABLE 3.4 – Propriétés de sécurité et résultats de la vérification

Identifiant de la requête	Requête	Résultat attendu	SANS_FUITE	FUITE_PARTIE_PRENANTE	FUITE_VEHICULE	FUITE_CENTRE_STOCKAGE
Q1	`not event(vehicle_send(va_2,conj_5,ct,msg_3))`	faux	●	●	●	●
Q2	`not event(vehicle_read(conj_5,ct,msg_3))`	faux	●	●	●	●
Q3	`not event(storage_write(ct))`	faux	●	●	●	●
Q4	`not event(storage_send(ct))`	faux	●	●	●	●
Q5	`not event(stakeholder_read(sap_2,ct,msg_3))`	faux	●	●	●	●
Q6	`not attacker_p4(abe_mk[])`	vrai	●	●	●	●
Q7	`not attacker_p4(gs_mk[])`	vrai	●	●	●	●
Q8	`secret storage_abe_sk_1,storage_abe_sk`	vrai	●	●	●	○
Q9	`secret stakeholder_abe_sk`	vrai	●	○	●	●
Q10	`secret vehicle_abe_sk_1,vehicle_abe_sk`	vrai	●	●	○	●
Q11	`secret msg_2,msg_1,msg`	vrai	●	○	○	●
Q12	`event(vehicle_read(va_2,ct,msg_3))` `==> event(vehicle_send(va_2,conj_5,ct,msg_3))`	vrai	●	●	●	●
Q13	`event(vehicle_read(va_2,ct,msg_3))` `==> event(storage_write(ct))`	vrai	●	●	●	●
Q14	`event(stakeholder_read(sap_2,ct,msg_3))` `==> event(vehicle_send(va_2,conj_5,ct,msg_3))`	vrai	●	●	●	●
Q15	`event(stakeholder_read(sap_2,ct,msg_3))` `==> event(storage_write(ct))`	vrai	●	●	●	●
Q16	`inj-event(storage_write(ct))` `==> inj-event(vehicle_send(va_2,conj_5,ct,msg_3))`	vrai	●	●	●	●

Q17	`attacker_p4(msg_3)`
	`&& event(storage_write(abe_enc(msg_3,ext_attrs(a_3,va_2),`
	`abe_pkgen(abe_mk[]))))`
	`==> event(vehicle_leak(va_2,abe_skgen(vap_2,abe_mk[])))`
	`|| (event(stakeholder_leak(sap_2,abe_skgen(sap_2,abe_mk[])))`
	`&& event(stakeholder_attrs(sap_2,a_3)))`
		vrai	●	●	●	●

● lorsque le résultat d'une requête est le même que celui de la colonne "Résultat attendu" et ○ lorsque le résultat d'une requête est différent de celui de la colonne "Résultat attendu"

si aucune clé de déchiffrement n'a été divulguée, l'attaquant ne peut pas lire les données (**P6**).

Quatre campagnes ont été réalisées : une sans fuite de clé de déchiffrement et une pour chaque type de fuite (une clé de déchiffrement ABE d'une partie prenante, une clé de déchiffrement ABE d'un véhicule et la clé de déchiffrement ABE du centre de stockage). Les résultats sont présentés dans la Table 3.4. La première colonne indique l'identifiant de la requête, la deuxième colonne détaille les requêtes et la troisième colonne indique le résultat attendu de l'évaluation des requêtes. Les dernières colonnes correspondent à l'évaluation des requêtes pour les différentes campagnes : un cercle plein indique que la requête est valide (correspond aux attentes) sinon le symbole est un cercle vide.

Cette table montre que, tout d'abord, sans aucune fuite, toutes les requêtes sont validées. Ainsi, toutes les propriétés sont satisfaites et le minimum attendu pour un tel protocole est fourni. De plus, il est formellement vérifié que le centre de stockage et l'attaquant ne sont pas en mesure de lire les données contenues dans un message chiffré. En cas de fuite, cinq requêtes sont invalidées. Dans le cas d'une fuite de la clé de déchiffrement ABE d'une partie prenante, le premier cercle vide est évident car il indique que l'attaquant a obtenu cette clé secrète. Le deuxième cercle vide indique que l'attaquant est capable de lire les messages, mais la dernière requête garantit que ces messages sont ceux que la partie prenante était déjà en mesure de lire. Dans le cas d'une fuite de clé de déchiffrement ABE d'un véhicule, les cercles vides ont une signification similaire à celle de la campagne précédente. Dans le cas d'une fuite de clé de déchiffrement ABE du centre de stockage, la seule information que l'attaquant est capable d'obtenir est la clé secrète du centre de stockage, l'attaquant n'est pas capable de lire les messages. Ainsi, cette dernière campagne nous donne la garantie que le centre de stockage n'est pas capable de lire les données.

3.4 Discussion

Plusieurs pistes d'amélioration sont nécessaires pour maintenir l'efficacité de l'approche dans les années à venir : intégrer efficacement la conformité au post-quantique, garantir l'anonymat des véhicules, empêcher les attaques par inférence et réutiliser la clé symétrique échangée au début des scénarios du protocole.

Conformité au post-quantique Les schémas ABE post-quantiques existants sont suffisamment expressifs pour construire des arbres d'accès requis pour l'application de la législation. Néanmoins, ils nécessitent encore des améliorations théoriques avant leur adoption dans la pratique. Comme indiqué dans la Section 3.1, pour parvenir à une sécurité adaptative, des contraintes doivent être appliquées aux arbres d'accès. Par exemple, la construction de Tsabary et al. [Tsabary 2019] exige que les arbres d'accès soient représentés en forme normale conjonctive. La transformation en FNC est possible, mais augmente la taille de l'arbre, ce qui a un impact sur la taille des textes chiffrés et des clés, ainsi que sur le temps de calcul. La même

discussion s'applique aux schémas ABE post-quantiques concurrents. Une autre limitation est le manque de constructions supportant un nombre d'attributs importants. Fondamentalement, pour permettre l'accès aux données récupérées par le véhicule lui-même, l'identité du véhicule est représentée sous la forme d'un attribut, ainsi il y aura autant d'attributs liés à l'identité du véhicule que de véhicules. Les schémas actuels ABE post-quantique ne s'adaptent pas bien à un univers contenant de nombreux attributs. Une fois encore, cette limitation a un impact sur la taille des textes chiffrés et des clés, ainsi que sur les temps de calcul. Le protocole reste entièrement compatible avec les schémas ABE basés sur les couplages, qui sont considérés comme matures mais vulnérables à un adversaire quantique.

Anonymat du véhicule L'anonymat du véhicule ne peut être garanti dans le protocole présenté précédemment car l'identité du véhicule est incluse dans la liste des attributs. Dans le schéma ABE actuellement utilisé dans le protocole, les attributs sont envoyés en clair et, par conséquent, l'identité du véhicule qui envoie les données n'est pas confidentielle. D'autres schémas ABE permettent de cacher les attributs contenus dans le chiffré (les schémas dits à politique cachée), mais ils ne sont pas nécessairement conformes aux normes post-quantiques, ces problématiques doivent encore être étudiées.

Requêtes et inférences Considérons qu'un schéma ABE permettant de cacher certains attributs du chiffré soit utilisé dans le protocole de telle sorte que les données chiffrées contiennent des attributs cachés tels que l'identité du véhicule émetteur, et des attributs publics ne révélant pas d'informations sur l'identité du véhicule tels que la région géographique d'émission de la donnée. Lorsqu'un conducteur voudra récupérer certaines de ses données, il soumettra une requête au centre de stockage et cherchera à récupérer uniquement les données qu'il sera en mesure de déchiffrer. Ainsi en répondant à la requête, le centre de stockage sera en mesure de lier les chiffrés au conducteur et donc d'inférer des informations sur le conducteur au moyen des attributs publics. Des contre-mesures empêchant le centre de stockage de faire ce lien doivent encore être étudiées.

Réutilisation de la clé symétrique Les premières étapes des scénarios du protocole consistent en un échange d'une clé symétrique et un échange de nonces. Lorsqu'un véhicule veut envoyer plusieurs données lors du scénario S1, pour chaque donnée à envoyer les premières étapes du protocole devront être répétées ce qui augmente le délai entre deux émissions de message contenant des données. Une amélioration consisterait à réutiliser la clé symétrique négociée pour plusieurs envois de données successifs. Néanmoins, comme nous le verrons dans le Chapitre 5, sans cette optimisation, les performances du protocole restent acceptables.

3.5 Conclusion

Dans ce chapitre, nous avons présenté un protocole sécurisé, formellement prouvé vis-à-vis de plusieurs propriétés de sécurité, permettant d'assurer la sécurité des données émises par des véhicules connectés. Ce protocole gère trois scénarios : un scénario d'envoi de données par un véhicule, un scénario de récupération de données par une partie prenante et un scénario de récupération de données par un véhicule. Le protocole est construit en utilisant trois schémas cryptographiques : le chiffrement basé attributs, le chiffrement symétrique et la signature de groupe. Ces schémas sont suffisamment génériques pour inclure une famille de schémas cryptographiques et notamment des schémas post-quantiques. Les propriétés du protocole sont vérifiées à l'aide d'un outil de vérification formelle automatique de propriétés de protocole, ProVerif. La vérification des propriétés a été réalisée dans plusieurs contextes : sans fuite de clé et en cas de fuite de clé de déchiffrement ABE de chaque entité légitime. En l'absence de fuite de clé, toutes les propriétés du protocole sont satisfaites. En cas de fuite d'une clé, un attaquant externe n'obtient pas plus de droits que l'entité initiale possédant la clé. Le protocole peut être amélioré en travaillant sur le protocole lui-même ou sur les schémas cryptographiques sous-jacents afin d'améliorer l'intégration des schémas post-quantiques dans le protocole, de garantir l'anonymat des véhicules, de réduire les inférences liées aux requêtes des conducteurs et de réduire le coût du protocole en réutilisant la clé symétrique échangée lors de plusieurs envois de données successifs. Le prochain chapitre se concentre sur une méthode permettant de mettre en place le contrôle d'accès et de définir les fonctions get_attrs et get_access_tree utilisées dans le protocole.

Chapitre 4

Contrôle d'accès

Sommaire

Ce quatrième chapitre est consacré au contrôle d'accès via la génération des attributs et des arbres d'accès utilisés dans les schémas ABE. Cette génération s'effectue en conformité avec les textes de lois tout en permettant au conducteur du véhicule de pouvoir paramétrer certains attributs selon ses propres choix. Ce chapitre débute par la présentation des attributs et arbres d'accès considérés puis détaille la méthode de génération du contrôle d'accès en deuxième section. Un cas d'utilisation illustrant la méthode est ensuite présenté en troisième section. Des améliorations de la méthode sont discutées en quatrième section puis une conclusion des travaux réalisés est enfin présentée en cinquième section.

4.1 Attributs et arbres d'accès

Le contrôle d'accès est réalisé en utilisant le chiffrement basé attributs qui repose sur les notions d'attributs et d'arbres d'accès. Dans ce qui suit, ces notions sont définies plus précisément, et les attributs et arbres d'accès considérés sont également présentés.

4.1.1 Définitions

Les notions d'attributs et d'arbres d'accès peuvent désigner plusieurs objets, elles peuvent désigner des valeurs utilisées dans les algorithmes ABE ainsi que des chaînes de caractères permettant de les représenter. Dans le reste de ce chapitre, les références à ces notions désignent uniquement leurs représentations sous forme de chaînes de caractères. Un *attribut* est donc une simple chaîne de caractères, plusieurs attributs peuvent être donnés en entrée à l'algorithme de chiffrement ABE sous la forme d'une liste d'attributs. Un *arbre d'accès* est une chaîne de caractères représentant un arbre dont les nœuds internes sont des connecteurs logiques binaires ET/OU, et dont les feuilles sont des attributs. Lorsqu'un arbre contient des nœuds ET et OU, des parenthèses sont utilisées pour indiquer la priorité des connecteurs. Une liste d'attributs *satisfait* un arbre d'accès lorsqu'il existe un sous-ensemble de branches de l'arbre dont les feuilles correspondent à un sous-ensemble d'attributs de la liste d'attributs. Lorsqu'un nœud de l'arbre est un connecteur logique de type OU, uniquement l'un des deux descendants du nœud est nécessaire pour satisfaire l'arbre. Lorsqu'un nœud de l'arbre est un connecteur logique de type ET, les deux descendants du nœud sont nécessaires pour satisfaire l'arbre. Par exemple, l'arbre suivant `(A OU B) ET C` peut être satisfait par trois listes d'attributs : `(A, C)`, `(B, C)` ou `(A, B, C)`. La satisfaction d'un arbre d'accès (associé à une clé de déchiffrement) par une liste d'attributs (associée à un chiffré) est le critère sur lequel repose l'algorithme de déchiffrement ABE pour accorder l'accès ou non au clair contenu dans le chiffré. Ainsi le reste de ce chapitre se concentre uniquement sur les arbres d'accès et les attributs pour définir le contrôle d'accès.

4.1.2 Structure des attributs et des arbres d'accès

Un *attribut* peut être un rôle, une identité, une information contextuelle ou une chaîne de caractères spécifiquement définie par une partie prenante :

— *un attribut rôle/identité* peut être l'identité d'une entité afin d'autoriser l'accès à une entité précise, cette identité est un pseudonyme généré par l'autorité de confiance, les identités considérées sont : l'identité du centre de stockage, l'identité d'une partie prenante ou l'identité d'un véhicule. Un attribut rôle/identité peut également être le rôle d'une partie prenante afin d'autoriser l'accès à plusieurs parties prenantes possédant le même rôle ;

— *un attribut contextuel* peut être la position d'un véhicule, la date d'envoi d'une donnée, le type d'une donnée (e.g. moteur, température) ou une situation spécifique (e.g. accident). Ces attributs permettent d'autoriser les parties prenantes à accéder aux données dans des conditions liées au contexte de la donnée (c'est-à-dire à une heure précise, dans un lieu spécifique ou en cas d'accident) ;

— *un attribut défini par une partie prenante* est une chaîne de caractères définie par une partie prenante en cas de délégation de clé de déchiffrement. Cette situation sera détaillée dans la suite de ce chapitre.

Une clé de déchiffrement contenant un *arbre d'accès* peut être détenue par le centre de stockage, un véhicule ou une partie prenante :

— *l'arbre d'accès du centre de stockage* : selon la propriété **P5** définie précédemment, le centre de stockage ne doit pas être en mesure d'accéder aux données contenues dans les messages envoyés par les véhicules. Néanmoins, le centre de stockage possède une clé de déchiffrement ABE pour déchiffrer les messages précédant l'envoi de données, lors de ses interactions avec un véhicule ou une partie prenante dans les scénarios du protocole. Son arbre d'accès comprend un unique attribut rôle/identité correspondant à son identité ;

— *l'arbre d'accès d'un véhicule* : selon la propriété **P3** définie précédemment, un véhicule doit pouvoir accéder aux données qu'il a précédemment émises, ainsi le véhicule doit posséder une clé de déchiffrement ABE lui permettant de déchiffrer ses propres données. Le véhicule n'étant pas sujet à des contraintes de contrôle d'accès supplémentaires, son arbre d'accès contient également un unique attribut rôle/identité correspondant à son identité ;

— *l'arbre d'accès d'une partie prenante* : les parties prenantes doivent pouvoir accéder aux données lorsqu'une donnée leur est directement adressée au moyen de leur identité, ou lorsqu'une donnée leur est accessible au moyen de leur rôle. Dans le second cas, des restrictions peuvent être spécifiées via des attributs contextuels indiquant dans quels contextes les parties prenantes sont autorisées à accéder aux données. Une partie prenante peut également accéder aux données lorsqu'une autre partie prenante lui délègue sa clé de déchiffrement. Ainsi, l'arbre d'accès d'une partie prenante intègre à minima un attribut rôle/identité correspondant à son identité et peut également intégrer des attributs correspondants à son rôle, au contexte, et en cas de délégation, des attributs définis par une partie prenante.

4.2 Génération du contrôle d'accès

Le contrôle d'accès doit être établi en prenant en compte les exigences **E1** à **E5**. Pour ce faire, la méthode de génération des attributs et des arbres d'accès permettant de mettre en place le contrôle d'accès est constituée de plusieurs étapes. La première étape est commune à la génération des attributs et des arbres d'accès, elle permet de prendre en compte la législation en générant des règles de sécurité dans le modèle de contrôle d'accès OrBAC à partir de la loi écrite en langage naturel. Cette étape étant difficile à automatiser, nous considérons qu'elle ne doit être effectuée manuellement qu'une seule fois par une personne compétente dans le domaine juridique. La génération des règles OrBAC semble être un compromis acceptable de part la simplicité de ce formalisme. Les étapes suivantes permettent de générer les attributs et les arbres d'accès à partir notamment des règles de sécurité OrBAC, elles peuvent facilement être automatisées en analysant ces règles. Les étapes de générations des attributs reposent sur les règles de sécurité OrBAC (**E1**), les contrats (**E2**) et le consentement du conducteur (**E3**). Les étapes de génération des

arbres d'accès reposent également sur les règles de sécurité OrBAC (**E1**), mais aussi sur la délégation (**E4**), et sur la prise en compte des situations exceptionnelles (**E5**). Cette approche en plusieurs étapes est proposée car générer en une seule étape les arbres d'accès et les attributs nécessiterait des compétences spécifiques qu'une personne compétente dans le domaine juridique n'est pas supposée avoir.

4.2.1 Génération des règles OrBAC

Cette première étape permet de prendre en compte la législation dans le contrôle d'accès en facilitant son exploitation via l'utilisation du modèle de contrôle d'accès OrBAC. Dans ce qui suit, des notions de modèles de contrôle d'accès sont introduites avant de détailler la génération des règles OrBAC à partir de la loi.

4.2.1.1 Modèle de contrôle d'accès OrBAC

Un *modèle de contrôle d'accès* permet de définir des règles de sécurité en s'appuyant sur les notions de sujet, d'objet et d'action pour définir les règles. Un *sujet* est une entité active interagissant avec le système (e.g. un utilisateur, un véhicule, un autre logiciel). Un *objet* est une ressource à laquelle un sujet souhaite accéder (e.g. un fichier, une donnée). Une *action* est une opération qu'un sujet souhaite réaliser (e.g. lire, écrire). Une *règle de sécurité* définit la permission d'un sujet à réaliser une action sur un objet. Le *contrôle d'accès basé sur les rôles* (en anglais *RBAC : Role-Based Access Control*) [Sandhu 1998] est un modèle de contrôle d'accès flexible s'appuyant sur la notion de *rôle*, cette notion permet d'abstraire et de regrouper des sujets. Dans ce modèle, les rôles sont associés aux sujets, les actions sont associées aux rôles et les sujets acquièrent des permissions en jouant des rôles. Le *contrôle d'accès basé sur les organisations* (en anglais *OrBAC : Organization-Based Access Control*) [Kalam 2003] est un modèle de contrôle d'accès plus général que RBAC. Ce modèle permet d'exprimer les règles de sécurité en utilisant uniquement des abstractions des notions de sujets, objets et actions. Dans OrBAC, une *organisation* est un groupe structuré d'entités actives, dans lequel les sujets jouent des *rôles* spécifiques. Une *activité* est un groupe d'une ou plusieurs actions, une *vue* est un groupe d'un ou plusieurs objets, et un *contexte* est une situation spécifique conditionnant la validité d'une règle. OrBAC comprend quatre règles de sécurité pour exprimer les relations entre les organisations, les rôles, les vues, les activités et les contextes : les obligations, les permissions, les interdictions et les recommandations. Ces règles de sécurité sont exprimées comme suit :

```
Obligation(O, R, V, A, C)
Permission(O, R, V, A, C)
Interdiction(O, R, V, A, C)
Recommandation(O, R, V, A, C)
```

Ces expressions signifient que dans le contexte `C`, l'organisation `O` accorde au rôle `R` l'obligation (ou la permission, l'interdiction, la recommandation) de réaliser

l'activité `A` sur la vue `V`.

4.2.1.2 Législation et règles OrBAC

Les lois considérées dans ces travaux traitent de l'accès aux données, elles décrivent des situations dans lesquelles certaines parties prenantes ont le droit ou non d'accéder aux données des véhicules connectés. Cependant, comme énoncé en Chapitre 1, ces lois sont encore au stade d'ébauche et ne sont quasiment pas exploitables en l'état, l'article 32 de la Loi d'Orientation des Mobilités [LOM 2019] en est un exemple. Nous considérons qu'avec le développement des véhicules connectés et des réflexions concernant la vie privée (le RGPD étant l'un des produits de ces réflexions), il est très probable qu'à l'avenir de telles lois existeront. L'un des objectifs de ces travaux est donc de proposer une méthode permettant d'établir un contrôle d'accès en anticipant la législation future.

Dans le cas où les lois présentent des conflits, tels que deux lois différentes autorisant et empêchant une partie prenante d'accéder à une même donnée, un juriste est considéré capable de résoudre ces conflits au cours de cette étape. Le processus de résolution des conflits n'entre pas dans le cadre de ces travaux. Dans le reste de ce chapitre, les règles OrBAC générées sont supposées ne pas présenter de conflits.

La génération des règles de sécurité OrBAC est réalisée par l'autorité de confiance, elle nécessite d'analyser chaque article de la loi manuellement afin d'identifier les informations pertinentes. Les articles de la loi stipulent des situations (utilisées pour définir le contexte `C`), dans lesquelles des données d'un certain type (utilisé pour définir la vue `V`), *peuvent* ou *ne peuvent pas* (utilisé pour définir le type de règles : `Permission` ou `Interdiction`) être accessibles à toutes les parties prenantes remplissant un certain rôle (utilisé pour définir le rôle `R`). Lorsqu'aucun contexte n'est mentionné, aucune restriction de contexte ne s'applique et le symbole * est écrit dans le champs `C` de la règle OrBAC générée. Chaque règle OrBAC définit également une organisation `O` et une activité `A`. Dans notre cas, l'organisation `O` correspond à l'unique organisation considérée dans ces travaux, l'Autorité de Confiance (AC), et l'activité `A` correspond à l'unique activité considérée, l'activité `lire`. Notons que la loi identifie les accès possibles ou interdits aux données, seules les relations `Permission` et `Interdiction` sont considérées dans la suite (les relations `Obligation` et `Recommandation` ne seront donc pas utilisées).

Illustrons cette démarche via l'article 32 de la Loi française d'Orientation des Mobilités, le paragraphe I.2 stipule :

> Rendre accessibles, **en cas d'accident de la route**, les **données des dispositifs d'enregistrement de données d'accident** et **les données d'état de délégation de conduite** enregistrées dans la période qui a précédé l'accident aux **officiers et agents de police judiciaire** aux fins de détermination des responsabilités ainsi qu'aux organismes chargés de l'enquête technique et de l'enquête de sécurité prévues à l'article L. 1621-2 du code des transports.

Considérons que les données des dispositifs d'enregistrement de données d'accident et les données d'état de délégation de conduite peuvent être regroupées sous le type de données *accident*, considérons également que les officiers et agents de police judiciaire peuvent être regroupés sous le rôle *police*. À l'issue de cette étape la règle OrBAC suivante est générée :

```
Permission(AC, police, accident, lire, accident)
```

Cette règle signifie donc que l'autorité de confiance permet aux parties prenantes possédant le rôle `police` de lire les données regroupées sous le type `accident` dans le contexte `accident`.

4.2.2 Génération des arbres d'accès

La génération des arbres d'accès des parties prenantes se déroule en deux étapes : l'extraction des arbres de la loi à partir des règles OrBAC de type `Permission`, puis la génération des arbres d'accès en utilisant la fonction `get_access_tree`. Étant donnée que la loi spécifie des autorisations et des interdictions en se basant sur le rôle des parties prenantes, le but de l'étape d'extraction des arbres de la loi est de générer des arbres d'accès en considérant uniquement les rôles des parties prenantes. Puis l'arbre d'accès d'une partie prenante est généré en sélectionnant l'arbre de la loi associé à son rôle. Des arbres d'accès peuvent être également générés dans deux situations supplémentaires : dans le cas d'une situation exceptionnelle, et suite à une délégation d'arbre d'accès.

4.2.2.1 Génération des arbres d'accès depuis la loi

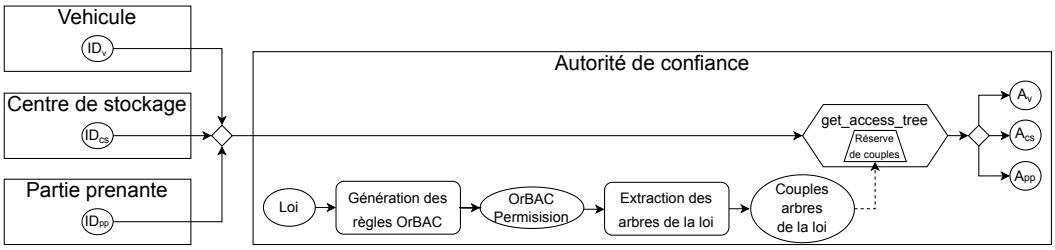

FIGURE 4.1 – Méthode de génération des arbres d'accès

La Figure 4.1 résume les principales étapes du processus de génération des arbres d'accès (les différentes étapes sont détaillées dans la suite de cette section). La légende de cette figure (ainsi que des figures 4.2, 4.3 et 4.4) est la suivante : un rectangle correspond à une entité (un véhicule, une partie prenante, le centre de stockage ou l'autorité de confiance) ; une ellipse correspond à une donnée ; un rectangle arrondi correspond à l'exécution d'un processus ; une flèche correspond au transfert d'une donnée ; un hexagone correspond à l'exécution d'une fonction spécifique ; un losange correspond à un choix exclusif sur ses flèches d'entrée ; un

trapèze correspond aux paramètres interne d'une fonction ; et une flèche en tiret correspond à un transfert de données utilisées en tant que paramètre interne d'une fonction.

OrBAC et arbres de la loi Cette étape est exécutée par l'autorité de confiance, elle correspond au processus *Extraction des arbres de la loi* de la Figure 4.1, elle prend en entrée les règles OrBAC de type `Permission`. Pour chaque règle, l'autorité de confiance extrait le rôle (champ `R` de la règle OrBAC), le type de données (champ `V` de la règle OrBAC) et les informations contextuelles (champ `C` de la règle OrBAC), puis elle génère un sous-arbre résultant du ET logique de ces champs : `C ET V ET R`. Ensuite, l'autorité regroupe les arbres contenant le même rôle, et forme un *arbre de la loi* résultant du OU logique des sous-arbres : `(C1 ET V1 ET R) OU ... OU (Cn ET Vn ET R)`. L'autorité génère en sortie de cette fonction une liste de couples associant à un rôle son arbre de la loi. La fonction `get_access_tree` repose sur cette liste de couples pour générer les arbres d'accès des parties prenantes.

Arbres d'accès finaux Cette étape est exécutée par l'autorité de confiance, elle correspond à la fonction `get_access_tree` de la Figure 4.1, elle prend une identité en entrée et se comporte comme suit. Si l'identité est celle d'un véhicule ou du centre de stockage, l'arbre d'accès résultant contient un seul nœud correspondant à l'identité donnée en entrée. Sinon l'identité est celle d'une partie prenante, les couples rôles-arbres sont filtrés afin de ne récupérer que l'arbre de la loi correspondant au rôle de la partie prenante. L'arbre d'accès final d'une partie prenante résulte du OU logique de son arbre de la loi et de son identité : `(C1 ET V1 ET R) OU ... OU (Cn ET Vn ET R) OU IDpp`. Notons que si le rôle d'une partie prenante n'apparaît dans aucun couple, l'arbre d'accès résultant contient un seul nœud correspondant à l'identité de la partie prenante.

4.2.2.2 Situation exceptionnelle

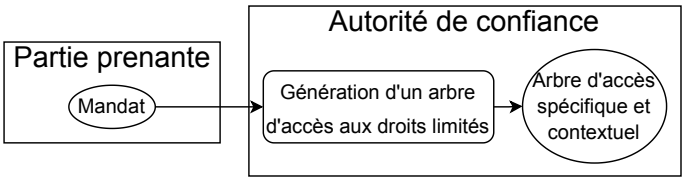

FIGURE 4.2 – Méthode de génération d'un arbre d'accès aux droits limités

Certaines parties prenantes spécifiques (e.g. police, justice) peuvent être assermentées par des organisations juridiques et ainsi obtenir des arbres d'accès donnant des droits limités dans des situations exceptionnelles, ce processus est illustré en Figure 4.2. Ces droits limités doivent permettre à la partie prenante d'accéder à des données auxquelles elle n'avait pas accès initialement, la partie prenante peut ainsi avoir accès à un nombre limité de données via la spécification de l'identité du

véhicule cible et des informations contextuelles (position, date, type de données et situation spécifique).

Le processus se déroule comme suit. Lorsqu'une partie prenante est assermentée, elle reçoit un mandat d'une organisation légale spécifiant les données auxquelles elle peut accéder. La partie prenante assermentée peut ainsi demander à l'autorité de confiance une clé de déchiffrement, incluant l'arbre d'accès souhaité, lui permettant d'accéder aux informations spécifiées dans le mandat. Cet arbre d'accès est un OU logique entre des sous-arbres, les sous-arbres sont des ET logique reliant des informations contextuelles telles qu'une date, un type de données et une position, avec l'identité d'un véhicule cible. L'autorité de confiance génère les clés de déchiffrement correspondantes et les délivre à la partie prenante assermentée via un canal sécurisé.

4.2.2.3 Délégation d'arbre d'accès

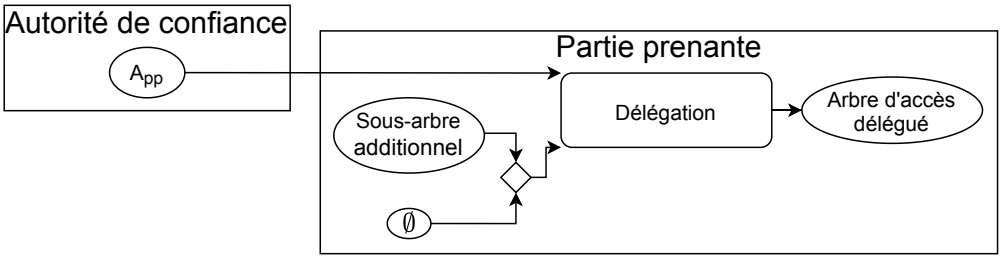

FIGURE 4.3 – Méthode de génération d'un arbre d'accès délégué

Une partie prenante peut souhaiter déléguer son accès à certaines données spécifiques à une autre partie prenante, cette situation est illustrée en Figure 4.3, la clé de déchiffrement résultante contient le même arbre d'accès ou un arbre d'accès plus restrictif que celui d'origine et peut inclure des attributs définis par la partie prenante. Une partie prenante est supposée être capable de déléguer sa clé de déchiffrement à une autre partie prenante avec laquelle elle a une relation légale (e.g. une filiale, un sous-traitant, une association). Deux cas de délégation de clé de déchiffrement sont considérés :

1. Ajout d'un nœud enfant à un nœud ET : des attributs supplémentaires sont nécessaires pour satisfaire le nœud. Le nœud enfant supplémentaire est un sous-arbre contenant des *attributs définis par une partie prenante* ;
2. Retrait d'un nœud enfant à un nœud OU : moins de combinaisons d'attributs permettent de satisfaire le nœud.

4.2.3 Génération des attributs

La génération des attributs utilisés lors du chiffrement est composée de trois étapes : l'extraction des attributs à partir de la loi, la sélection des attributs du conducteur, et l'exécution de la fonction `get_attrs`.

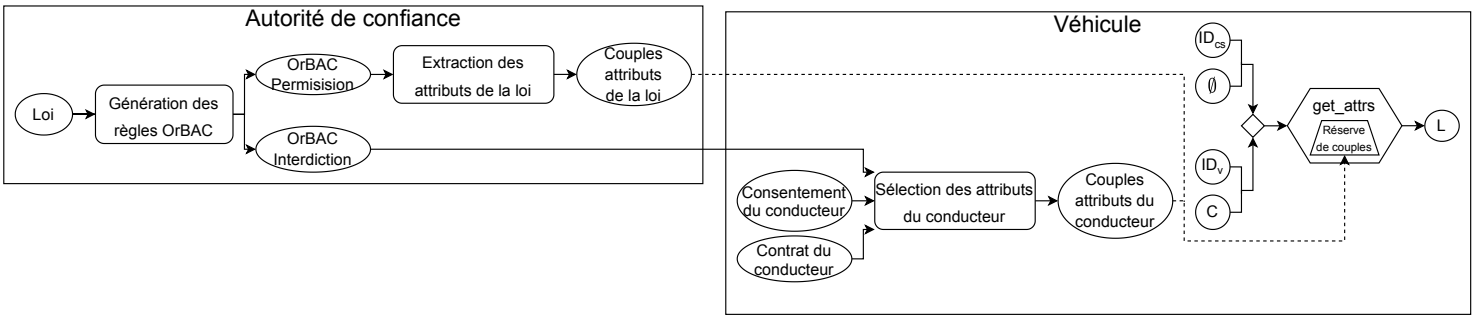

FIGURE 4.4 – Méthode de génération des attributs

4.2.3.1 OrBAC et attributs de la loi

Cette étape correspond au processus *Extraction des attributs de la loi* de la Figure 4.4. Les règles OrBAC de type `Permission` et `Interdiction` étant supposées ne pas être en conflit, seules les règles de type `Permission` sont utilisées dans cette étape par l'autorité de confiance pour générer les attributs de la loi. Cette étape se déroule comme suit. L'autorité crée une liste de couples vide. Pour chaque règle, le type de données (`V`), le contexte (`C`) et le rôle (`R`) sont extraits, l'autorité forme un couple associant à un contexte et un type de donnée le rôle d'une partie prenante : `(V, C) -> (R)`, ce couple est ajouté à la liste. L'autorité renvoie la liste des couples en sortie de cette étape. La fonction `get_attr` repose sur cette liste de couples pour générer les attributs utilisés lors du chiffrement, cette liste est incluse dans le véhicule lors de sa fabrication et doit être mise à jour lorsque les lois sont elles-mêmes modifiées. Le conducteur ne peut en aucun cas modifier cette liste.

4.2.3.2 Sélection des attributs du conducteur

Le conducteur peut ajouter des attributs pour le chiffrement des données pour autant qu'ils n'enfreignent pas une règle OrBAC de type `Interdiction`, cette étape est réalisée dans le véhicule et correspond au processus *Sélection des attributs du conducteur* de la Figure 4.4. L'ajout d'attributs par le conducteur peut se produire dans deux situations. Premièrement, le conducteur peut consentir à ouvrir ses données à d'autres parties prenantes, il peut ainsi ajouter des *attributs rôles/identités* correspondant aux identités des parties prenantes de son choix ou des *attributs définis par une partie prenante* pour partager ses données avec une partie prenante possédant une clé de déchiffrement déléguée. Deuxièmement, lorsqu'un conducteur a signé un contrat avec une partie prenante particulière, les attributs correspondants à ce contrat doivent être ajoutés lors du chiffrement. Ces attributs peuvent être simplement l'identité de la partie prenante correspondante ainsi que des *attributs définis par la partie prenante* stipulés dans le contrat. Dans les deux cas, des couples sont formés spécifiant un contexte et un type de données qui doivent être satisfaits pour générer les attributs correspondants. Dans le premier cas, le contexte et le type de données sont spécifiés par le conducteur, dans le second cas, ils sont spécifiés

dans le contrat. Ces couples sont également intégrés à la fonction `get_attrs`.

Attributs rôle/identité Lorsqu'un conducteur souhaite ajouter un attribut correspondant à l'identité d'une partie prenante, ou lorsqu'un contrat spécifie un attribut correspondant à une identité, cela doit être fait dans le respect de la loi. Si les données considérées ne sont couvertes par aucun article de loi et donc par aucune règle OrBAC, alors le conducteur peut utiliser n'importe quelle identité, et le contrat doit ajouter l'identité spécifiée dans le contrat. Dans le cas contraire, si les données considérées sont couvertes par une règle OrBAC de type `Interdiction` (c'est-à-dire correspondant au rôle, au type de données et au contexte), le conducteur/contrat ne peut ajouter que des attributs qui n'entrent pas en conflit avec cette règle. À cette fin, une fonction capable, à partir de l'identité d'une partie prenante, de fournir son rôle, est également intégrée au véhicule et est régulièrement mise à jour.

Attributs définis par une partie prenante Ces attributs ne concernent que les parties prenantes possédant une clé de déchiffrement déléguée, ils peuvent être ajoutés librement par un conducteur ou par le biais d'un contrat qui stipule la partie prenante déléguée ainsi que les attributs concernés. Les conducteurs sont supposés connaître les attributs nécessaires pour accorder l'accès à une partie prenante possédant une clé de déchiffrement déléguée.

4.2.3.3 Attributs finaux

Les attributs utilisés pour chiffrer les données à envoyer sont sélectionnés par le véhicule en utilisant la fonction `get_attrs`, représentée en Figure 4.4. Cette fonction prend en entrée une identité et un contexte (pour simplifier la définition, considérons que le contexte inclut le type de données) et se comporte comme suit. Une liste d'attributs vide est créée. Le contexte est utilisé pour filtrer la liste des couples, les couples résultants sont parcourus pour récupérer l'attribut associé, cet attribut peut être un *attribut rôle/identité* ou un *attribut défini par une partie prenante*, l'attribut est ensuite inséré dans la liste d'attributs. L'identité donnée en entrée est ensuite insérée dans la liste d'attributs. Enfin des attributs liés au contexte sont insérés dans la liste d'attributs, ces attributs sont : la date d'envoi du message, l'heure d'envoi du message, le type de données et des situations spécifiques supplémentaires si elles ont lieu telle que la détection d'un accident. La liste d'attributs renvoyée par la fonction est ensuite utilisée lors du chiffrement ABE. Lorsque le contexte donné en entrée est vide, les étapes de filtrage des couples et d'ajout des attributs contextuels sont ignorées, la fonction ne réalise que l'étape d'ajout de l'identité donnée en entrée à la liste d'attributs. En pratique, la fonction `get_attrs` est exécutée dans deux situations. Dans la première situation, l'identité du centre de stockage et un contexte vide sont donnés en entrée, la liste d'attributs renvoyée contient uniquement l'identité du centre de stockage. Dans la deuxième situation, l'identité du véhicule et un contexte non vide sont donnés en entrée, la liste d'attributs renvoyée contient à minima l'identité du véhicule et des attributs contextuels, puis si des couples sont

satisfaits par le contexte, la liste contient également des attributs rôle/identité ou des attributs définis par une partie prenante.

4.2.4 Satisfaction des exigences

Le contrôle d'accès satisfait les exigences **E1** à **E5** définies en Chapitre 2 par la génération d'attributs et d'arbres d'accès permettant de prendre compte chaque exigences.

L'exigence **E1** est satisfaite par 1) la génération de règle OrBAC de type `Permission` et `Interdiction` en utilisant la législation au cours du processus `Génération des règles OrBAC`; 2) par l'utilisation des règles OrBAC de type `Permission` pour générer des arbres d'accès lors du processus `Extraction des arbres de la loi`; et 3) par l'utilisation des règles OrBAC de type `Permission` pour générer des attributs lors du processus `Extraction des attributs de la loi`. La prise en compte des règles OrBAC de type `Interdiction` a lieu lors du processus `Sélection des attributs du conducteur`, en filtrant les attributs que le conducteur ou le contrat spécifie lorsqu'ils sont en conflit avec ces règles.

Les exigences **E2** et **E3** sont satisfaites par la prise en compte d'attributs, liés à l'identité d'une partie prenante ou à un attribut défini par une partie prenante, provenant du consentement du conducteur ou de ses contrats au cours du processus `Sélection des attributs du conducteur`. Ainsi que par l'ajout de l'identité de la partie prenante lors de la génération de son arbre d'accès au cours de l'exécution de la fonction `get_access_tree` et par la gestion du processus de délégation permettant notamment l'ajout d'attributs définis par une partie prenante.

L'exigence **E4** est satisfaite par le processus de délégation d'arbre d'accès, ainsi que par la prise en compte d'attributs définis par une partie prenante lors du processus `Sélection des attributs du conducteur`. Enfin, l'exigence **E5** est satisfaite par la génération d'une clé de déchiffrement associée à un arbre d'accès donnant des droit limités. Ces droits limités permettent de déchiffrer des messages en spécifiant un contexte et un type de données précis ainsi qu'une identité d'un véhicule. Cette clé est générée suite à l'obtention d'un mandat par une partie prenante assermentée lors du processus de situations exceptionnelles.

4.3 Cas d'utilisation

Un cas d'utilisation simple permettant d'illustrer la méthode de mise en place du contrôle d'accès est présenté dans ce qui suit. Ce cas d'utilisation commence par la présentation du contexte comprenant les parties prenantes et les données considérées. Il continue par la présentation de la spécification du contrôle d'accès incluant les lois considérées ainsi que le contrat et le consentement du conducteur. Puis la méthode de génération du contrôle d'accès précédemment définie est appliquée en prenant en compte la spécification. Le cas d'utilisation se termine en vérifiant si les mécanismes de contrôle d'accès implantés ont permis d'atteindre les objectifs définis lors de la spécification.

4.3.1 Contexte

Les parties prenantes considérées sont : un service météorologique (`meteo`), deux forces de police (`policeA` et `policeB`), un gestionnaire d'infrastructures routières (`infra`) possédant trois agences, deux dans la région A (`infraA1` et `infraA2`), et une dans la région B (`infraB1`), une assurance (`assur`) employant un sous-traitant (`st1`) traitant uniquement les données de vitesse et un sous-traitant (`st2`) traitant uniquement les données de position. Les rôles des différentes parties prenantes sont les suivants :

```
id:meteo, role:service_meteo
id:policeA, role:police
id:policeB, role:police
id:infra, role:infra_route
id:assur, role:assurance
id:st1, role:sous-traitant
id:st2, role:sous-traitant
```

Chaque partie prenante est considérée être une entité juridique. Si une partie prenante possède plusieurs agences, chaque agence dépend juridiquement de la société mère. Ceci s'applique aux agences `infraA1`, `infraA2` et `infraB1` qui dépendent de `infra`.

Un véhicule nommé `veh` est considéré, il est équipé de capteurs générant des données dont les types sont : `vitesse`, `pollution`, `position`, `temperature`, `route_endommagee` et `accident`. Une donnée d'accident est émise lorsque le véhicule lui-même a un accident et est capable de le détecter. Pour simplifier la représentation de la position, nous considérons que la carte du monde est divisée en tuiles et qu'une position correspond à une tuile spécifique, par exemple, `tuile5`. Six messages sont considérés, ils sont envoyés par `veh` le 22-07-2021 et présentés dans la Table 4.1.

Table 4.1 – Messages envoyés par veh le 22-07-2021

ID	Heure	Position	Type de donnée
M1	09:55:20	tuile5	pollution
M2	09:55:30	tuile5	position
M3	09:55:40	tuile6	temperature
M4	09:58:05	tuile6	route_endommagee
M5	09:59:00	tuile6	vitesse
M6	10:00:00	tuile7	accident

4.3.2 Spécification du contrôle d'accès

Les articles suivants sont inspirés de l'article 32 de la Loi française d'Orientation des Mobilités :

L1 *Les gestionnaires d'infrastructure routière peuvent accéder aux données relatives à l'état de la route* ;

L2 *Les forces de police ne sont pas autorisées à accéder aux données de vitesse* ;

L3 *En cas d'accident, les forces de police peuvent accéder aux données relatives à l'accident* ;

L4 *En cas d'accident, une force de police peut être assermentée et peut alors accéder aux données de position précédant l'accident.*

Il est considéré que :

— en utilisant L4, `policeA` obtient un mandat l'autorisant à accéder aux données de position précédant l'accident signalé dans M6 (c'est-à-dire les données de position de M2) ;

— le conducteur a signé un contrat de type pay-as-you-drive avec `assur`, le contrat stipule que ses données de vitesse doivent être accessibles par `assur` et `st1` ;

— le conducteur consent à partager ses données de position avec `infra` et ses agences dans la région A, ses données de température avec `meteo`, et ses données de vitesse avec `policeB` ;

— le conducteur doit avoir accès à toutes les données émises par son véhicule.

En résumé, les mécanismes de contrôle d'accès doivent assurer que :

1. les données de pollution de M1 ne doivent être accessibles par aucune partie prenante car aucune spécification de contrôle d'accès n'identifie une partie prenante autorisée à accéder aux données de pollution ;
2. les données de position de M2 doivent être accessibles par `policeA` puisqu'il a été assermenté, et par `infra`, `infraA1`, et `infraA2` puisque le conducteur consent à partager ses données de position avec `infra` et ses agences dans la région A ;
3. les données de température de M3 doivent être accessibles par `meteo` car le conducteur consent à les partager uniquement avec `meteo` ;
4. les données de routes endommagées provenant de M4 doivent être accessibles par `infra`, `infraA1`, `infraA2` et `infraB1` car L1 spécifie que chaque gestionnaire d'infrastructures routières peut accéder aux données de routes endommagées ;
5. les données de vitesse provenant de M5 doivent être accessibles par `assur` et `st1` car cela est spécifié dans le contrat pay-as-you-drive du conducteur ;
6. les données d'accident de M6 doivent être accessibles par `policeA` et `policeB` car L3 spécifie que les forces de police sont autorisées à accéder aux données d'accident ;

TABLE 4.2 – Matrice des droits d'accès représentant les objectifs du contrôle d'accès

	M1	M2	M3	M4	M5	M6
meteo	○	○	●	○	○	○
policeA	○	●	○	○	○	●
policeB	○	○	○	○	○	●
infra	○	●	○	●	○	○
infraA1	○	●	○	●	○	○
infraA2	○	●	○	●	○	○
infraB1	○	○	○	●	○	○
assur	○	○	○	○	●	○
st1	○	○	○	○	●	○
st2	○	○	○	○	○	○
veh	●	●	●	●	●	●

● représente un accès autorisé et ○ représente un accès interdit

7. toutes les données doivent êtres accessbiles par `veh`.

Notons que, même si le conducteur consent à partager ses données de vitesse avec `policeA` et `policeB`, étant donné que L2 interdit les forces de police d'accéder à ce type de données, `policeA` et `policeB` ne doivent pas avoir le droit d'y accéder. La matrice de contrôle d'accès correspondante est représentée en Table 4.2.

4.3.3 Mise en place du contrôle d'accès

4.3.3.1 Représentation des attributs

Chaque attribut est défini par un préfixe et une valeur. Dans ce qui suit nous présentons les valeurs considérées pour les différents types d'attributs.

Attributs rôles/identités des parties prenantes :

— le rôle d'une partie prenante préfixé par `pp_role`, les valeurs possibles sont `police` et `infra_route` (e.g. `pp_role:police`) ;

— l'identité d'une partie prenante préfixée par `pp_id`, les valeurs possibles sont `meteo`, `policeA`, `policeB`, `infra`, `assur`, `st1` et `st2` (e.g. `pp_id:meteo`).

Attributs contextuels :

— la date préfixée par `date` (e.g. `date:22-07-2021`) ;

— l'heure préfixée par `heure` (e.g. `heure:08-31`) ;

— la position préfixée par `position` (e.g. `position:tuile10`) ;

— le type de données préfixé par `type`, les valeurs possibles sont `pollution`, `position`, `temperature`, `route_endommagee`, `vitesse` et `accident` (e.g. `type:pollution`) ;

— un label contextuel préfixé par `label`, la valeur `accident` est la seule valeur considérée, l'attribut ainsi formé est `label:accident`.

Attributs définis par une partie prenante : Ces attributs sont précédés du préfixe `pp_attr`, les valeurs considérées sont :

— dans le cas de la délégation de `infra` : les valeurs sont `regionA` et `regionB` (e.g. `pp_attr:regionA`) ;

— dans le cas de la délégation de `assur` : les valeurs sont `vitesse` et `position` (e.g. `pp_attr:vitesse`).

Enfin, **l'identité du véhicule** est également un attribut, cet attribut est préfixé par `v_id`, l'attribut ainsi formé est `v_id:veh`.

4.3.3.2 Règles OrBAC

Trois règles OrBAC sont générées à partir des articles L1, L2 et L3 suite à l'exécution du processus `Génération des règles OrBAC` :

— À partir de la règle L1, "*Les gestionnaires d'infrastructures routières (R) peuvent (Permission) accéder aux données relatives à l'état de la route (V)*", la règle OrBAC suivante est générée : `Permission(AC, infra_route, route_endommagee, lire, *)` ;

— À partir de la règle L2, "*Les forces de police (R) ne sont pas autorisées (Interdiction) à accéder aux données de vitesse (V)*", la règle OrBAC suivante est générée : `Interdiction(AC, police, vitesse, lire, *)` ;

— À partir de la règle L3, "*En cas d'accident (C), les forces de police (R) peuvent (Permission) accéder aux données relatives à l'accident (V)*", la règle OrBAC suivante est générée : `Permission(AC, police, accident, lire, accident)`.

En résumé, les règles OrBAC ainsi générées sont :

```
Permission(AC, infra_route, route_endommagee, lire, *)
Interdiction(AC, police, vitesse, lire, *)
Permission(AC, police, accident, lire, accident)
```

Notons que la règle L4, "*En cas d'accident, une force de police peut être assermentée et peut alors accéder aux données de position précédant l'accident*", ne peut pas être utilisée pour générer une règle OrBAC car cette règle est dédiée au processus de situation exceptionnelle, elle est utilisée par l'autorité de confiance lors de la génération d'arbre d'accès aux droits limités.

4.3.3.3 Arbres d'accès

Couples arbres de la loi À partir des règles OrBAC de type `Permission`, les couples suivants sont générés par l'autorité de confiance en sortie du processus `Extraction des arbres de la loi` :

```
(role:infra_route)  ->  (pp_role:infra_route
                         ET type:route_endommagee)
(role:police)  ->  (pp_role:police ET type:accident
                                   ET label:accident)
```

Arbres d'accès des parties prenantes Suite à l'exécution de `get_access_tree` se basant sur les couples précédemment définis, les arbres d'accès suivants sont générés par l'autorité de confiance :

```
policeA:
    (pp_role:police ET type:accident ET label:accident)
    OU pp_id:policeA
policeB:
    (pp_role:police ET type:accident ET label:accident)
    OU pp_id:policeB
infra:
    (pp_role:infra_route ET type:route_endommagee) OU pp_id:infra
meteo:
    pp_id:meteo
assur:
    pp_id:assur
st1:
    pp_id:st1
st2:
    pp_id:st2
```

Notons que l'arbre d'accès des parties prenantes dont le rôle n'apparaît dans aucune règle OrBAC est uniquement composé de leur identité.

Arbres d'accès du véhicule L'autorité de confiance génère également l'arbre d'accès du véhicule en utilisant la fonction `get_access_tree`, cet arbre contient uniquement son identité :

```
veh:
    v_id:veh
```

Arbres d'accès suite à une délégation Deux cas de délégation sont considérés. Dans le premier cas, `infra` délègue sa clé de déchiffrement à ses agences en ajoutant un attribut correspondant à la région de l'agence sous la forme d'un nœud ET avec l'attribut identité. Dans le second cas, `assur` délègue sa clé de déchiffrement à ses

sous-traitants en ajoutant un attribut correspondant au type de données traité par le sous-traitant sous la forme d'un nœud ET avec l'attribut identité. Ainsi, les arbres d'accès résultant de l'exécution du processus `Délégation` sont :

```
infraA1/infraA2(delegue):
    (pp_role:infra_route ET type:route_endommagee)
    OU (pp_id:infra ET pp_attr:regionA)
infraB1(delegue):
    (pp_role:infra_route ET type:route_endommagee)
    OU (pp_id:infra ET pp_attr:regionB)
st1(delegue):
    pp_id:assur ET pp_attr:vitesse
st2(delegue):
    pp_id:assur ET pp_attr:position
```

Arbre d'accès suite à une situation exceptionnelle Étant donnée que `policeA` a obtenu un mandat (règle L4), l'autorité de confiance, en raison de la survenue d'un accident, lui fournit une clé de déchiffrement contenant l'arbre d'accès suivant :

```
policeA(exceptionnel):
    v_id:veh ET type:position ET position:tuile5
    ET date:22-07-2021 ET heure:09-55
```

4.3.3.4 Attributs

Couples attributs de la loi L'autorité de confiance exécute le processus `Extraction des attributs de la loi`, en utilisant les règles OrBAC, et génère les couples suivants :

```
(type:accident, label:accident)  ->  (pp_role:police)
(type:route_endommagee)  ->  (pp_role:infra_route)
```

Ces couples sont intégrés dans le véhicule lors de sa fabrication.

Couples attributs du conducteur Le véhicule exécute le processus `Sélection des attributs du conducteur` en utilisant le contrat et le consentement du conducteur :

Contrat À partir du contrat pay-as-you-drive, deux attributs sont ajoutés lorsqu'une donnée de vitesse est émise, l'un spécifiant l'identité `assur` et l'autre spécifiant l'attribut défini par la partie prenante `vitesse` :

```
(type:vitesse)  ->  (pp_id:assur)
(type:vitesse)  ->  (pp_attr:vitesse)
```

Consentement À partir du consentement du conducteur, `meteo` peut accéder aux données de `temperature` en utilisant son identité et les gestionnaires d'infrastructure de la région A peuvent accéder aux données de position en utilisant l'identité `infra` et l'attribut défini par la partie prenante `regionA` :

```
(type:temperature)  ->  (pp_id:meteo)
(type:position)  ->  (pp_id:infra)
(type:position)  ->  (pp_attr:regionA)
```

Notons qu'aucun attribut n'est sélectionné à partir du consentement du conducteur pour l'accès de l'identité `policeB` aux données de vitesse, car le véhicule filtre l'identité à l'aide de la règle OrBAC de type `Interdiction`.

Attributs utilisés lors du chiffrement Enfin, les attributs finaux utilisés lors du chiffrement sont générés en utilisant la fonction `get_attrs` reposant sur les couples définis précédemment. Pour les six événements d'envoi de données, de M1 à M6, les attributs suivants sont utilisés lors du chiffrement :

```
M1: (date:22-07-2021, heure:09-55, position:tuile5, type:pollution,
    v_id:veh)
M2: (pp_id:infra, pp_attr:regionA, date:22-07-2021, heure:09-55,
    position:tuile5, type:position, v_id:veh)
M3: (pp_id:meteo, date:22-07-2021, heure:09-55, position:tuile6,
    type:temperature, v_id:veh)
M4: (pp_role:infra_route, date:22-07-2021, heure:09-58,
    position:tuile6, type:route_endommagee, v_id:veh)
M5: (pp_id:assur, pp_attr:vitesse, date:22-07-2021, heure:09-59,
    position:tuile6, type:vitesse, v_id:veh)
M6: (pp_role:police, date:22-07-2021, heure:10-00, position:tuile7,
    type:accident, v_id:veh, label:accident)
```

4.3.4 Évaluation du contrôle d'accès

En tenant compte des attributs associés aux six événements et des arbres d'accès des différentes parties prenantes décrits ci-dessus, les objectifs du contrôle d'accès sont vérifiés :

1. les données de pollution de M1 ne sont accessibles par aucune partie prenante, en effet M1 ne contient pas d'attributs correspondant à l'identité ou le rôle d'une partie prenante. Par ailleurs, la clé exceptionnelle de `policeA` ne permet pas de déchiffrer ce message car l'arbre associé à cette clé ne contient pas l'attribut associé au type de donnée pollution :

```
M1: (date:22-07-2021, heure:09-55, position:tuile5, type:pollution,
    v_id:veh)
```

2. les données de position de M2 sont accessibles par `policeA` via sa clé exceptionnelle, par `infra` via l'attribut `pp_id:infra`, et par `infraA1` et `infraA2` via les attributs `pp_id:infra` et `pp_attr:regionA` :

```
M2: (pp_id:infra, pp_attr:regionA, date:22-07-2021,
    heure:09-55, position:tuile5, type:position, v_id:veh)

policeA(exceptionnel):
    v_id:veh ET type:position ET position:tuile5
    ET date:22-07-2021 ET heure:09-55
infra:
    (pp_role:infra_route ET type:route_endommagee) OU pp_id:infra
infraA1/infraA2(delegue):
    (pp_role:infra_route ET type:route_endommagee)
    OU (pp_id:infra ET pp_attr:regionA)
```

3. les données de température de M3 sont accessibles par `meteo` via l'attribut `pp_id:meteo` :

```
M3: (pp_id:meteo, date:22-07-2021, heure:09-55, position:tuile6,
    type:temperature, v_id:veh)

meteo:
    pp_id:meteo
```

4. les données de routes endommagées de M4 sont accessibles par `infra`, `infraA1`, `infraA2` et `infraB1` via les attributs `pp_role:infra_route` et `type:route_endommagee` :

```
M4: (pp_role:infra_route, date:22-07-2021, heure:09-58,
    position:tuile6, type:route_endommagee, v_id:veh)

infra:
    (pp_role:infra_route ET type:route_endommagee) OU pp_id:infra
infraA1/infraA2(delegue):
    (pp_role:infra_route ET type:route_endommagee)
    OU (pp_id:infra ET pp_attr:regionA)
infraB1(delegue):
    (pp_role:infra_route ET type:route_endommagee)
    OU (pp_id:infra ET pp_attr:regionB)
```

5. les données de vitesse de M5 sont accessibles par `assur` via l'attribut `pp_id:assur`, et par `st1` via les attributs `pp_id:assur` et `pp_attr:vitesse` :

```
M5: (pp_id:assur, pp_attr:vitesse, date:22-07-2021, heure:09-59,
    position:tuile6, type:vitesse, v_id:veh)

assur:
    pp_id:assur
st1(delegue):
    pp_id:assur ET pp_attr:vitesse
```

6. les données d'accident de M6 sont accessibles par `policeA` et `policeB` via les attributs `pp_role:police`, `type:accident` et `label:accident` :

```
M6: (pp_role:police, date:22-07-2021, heure:10-00, position:tuile7,
    type:accident, v_id:veh, label:accident)

policeA:
    (pp_role:police ET type:accident ET label:accident)
    OU pp_id:policeA
policeB:
    (pp_role:police ET type:accident ET label:accident)
    OU pp_id:policeB
```

7. toutes les données sont accessibles par `veh` via l'attribut `v_id:veh` contenu dans tous les messages ainsi que dans l'arbre d'accès associée à sa clé de déchiffrement.

Ces résultats concordent avec les objectifs représentés dans la matrice d'accès de la Table 4.2, ce qui montre que la méthode de génération d'attributs et d'arbres d'accès est pertinente pour faire respecter les droits d'accès décrits dans la matrice.

4.4 Discussion

L'approche proposée dans ce chapitre peut être étendue et améliorée en considérant les points suivants : l'évolution de la législation, l'évolution du consentement ainsi que des aspects plus pratiques.

Évolution de la législation Pour prendre en compte l'évolution de la législation, il est nécessaire de faire évoluer l'approche en considérant quatre situations :

— L'ajout d'un article de loi interdisant un accès. Cette situation entraîne la génération d'une nouvelle règle OrBAC de type `Interdiction` par l'autorité de confiance, cette règle doit être transférée au véhicule. Le véhicule doit ensuite vérifier si des conflits ont lieu entre les couples contenus dans la fonction `get_attrs` et la nouvelle règle `Interdiction`, et retirer les couples en conflit le cas échéant ;

— La suppression d'un article de loi interdisant un accès. Cette situation nécessite uniquement de supprimer la règle OrBAC de type `Interdiction` correspondante dans le véhicule, aucune modification supplémentaire n'est nécessaire ;

— L'ajout d'un article de loi autorisant un accès. Dans cette situation une nouvelle règle OrBAC de type `Permission` est créée, cette règle permet de générer un arbre d'accès donnant plus de droits aux parties prenantes possédant le rôle spécifié dans la règle, de nouvelles clés de déchiffrement doivent être générées par l'autorité de confiance et délivrées aux parties prenantes possédant le rôle ;

— La suppression d'un article de loi autorisant un accès. Cette situation réduit les permissions d'une partie prenante à accéder aux données par la génération d'une clé de déchiffrement associée à un arbre d'accès plus restrictif que la clé d'origine. Un mécanisme de révocation de clé doit être utilisé afin d'invalider les anciennes clés de déchiffrement. Des schémas ABE supportant un tel mécanisme doivent être utilisés dans le protocole.

Évolution du consentement Dans la méthode décrite précédemment, le conducteur peut consentir à partager ses données avec une partie prenante en spécifiant son identité. Le consentement peut être étendu en permettant au conducteur de consentir à partager ses données avec toutes les parties prenantes possédant un certain rôle. Cependant, cette amélioration n'est pas applicable en l'état. En effet une règle OrBAC de type `Permission` dans laquelle le rôle de la partie prenante est mentionné est nécessaire pour qu'une partie prenante possède un rôle dans son arbre d'accès. Par ailleurs, dans cette situation, l'arbre d'accès contient également, en supplément du rôle, un type de donnée et potentiellement un contexte ce qui restreint les listes d'attributs permettant de satisfaire l'arbre. Une solution consisterait à rajouter systématiquement dans les arbres d'accès des parties prenantes un attribut représentant le rôle de la partie prenante uniquement en cas de consentement pour le différencier de celui de la législation, par exemple `pp_role_consent:roleX`. Lors du consentement du conducteur, il est alors nécessaire de vérifier dans le véhicule que le rôle n'entre pas en conflit avec une règle OrBAC de type `Interdiction`.

Les aspects plus pratiques Plusieurs aspects peuvent être considérés :

— L'interface utilisateur. Elle doit être étudiée afin de la rendre facile d'utilisation au conducteur. Cette interface doit permettre au conducteur d'accéder aux contrats qu'il a signés, de spécifier son consentement pour partager ses données, ainsi qu'une liste des messages envoyés par son véhicule sur une certaine période donnée ;

— L'obtention d'une liste de parties prenantes en mesure d'accéder aux données. Le conducteur pourrait obtenir cette liste en demandant à l'autorité de confiance la liste des parties prenantes en mesure de déchiffrer certaines listes d'attributs. Le conducteur pourrait également demander au centre de stockage quelles parties prenantes ont récupéré ses messages ;

— L'implémentation des contrats. Des considérations de sécurité sont à prendre en compte lors de la mise en place des contrats dans le véhicule tel qu'empêcher un attaquant d'insérer de faux contrats.

4.5 Conclusion

Une méthode de génération du contrôle d'accès est présentée dans ce chapitre, cette méthode est vérifiée vis-à-vis d'exigences permettant de prendre en compte différentes sources de contrôle d'accès. La mise en place du contrôle d'accès repose sur la satisfaction d'arbres d'accès par des listes d'attributs. Les attributs peuvent représenter un rôle, une identité, un contexte ou être générés par une partie prenante en cas de délégation. Les arbres d'accès sont des arbres contenant des nœuds sous la forme de ET ou OU logique et des feuilles sous la forme d'attributs. La méthode de génération du contrôle d'accès utilise la législation, les contrats du conducteur et le consentement du conducteur pour générer les attributs et les arbres d'accès. Une partie prenante peut déléguer sa clé de déchiffrement pour simplifier son organisation. Lors de situations exceptionnelles, telles qu'un accident ou un crime, l'autorité de confiance peut générer des arbres donnant des droits limités à une partie prenante possédant un mandat, telle que les forces de police, l'autorisant à accéder aux données relatives à la situation exceptionnelle. Le contrôle d'accès est illustré au travers d'un cas d'étude conçu pour représenter une situation la plus réelle possible. Plusieurs améliorations peuvent être envisagées pour améliorer l'approche : l'ajout et la suppression de loi peut être pris en compte dans l'approche, le consentement peut être étendu en permettant l'ajout de rôles de parties prenantes, des aspects techniques peuvent être considérés tels que la conception de l'interface utilisateur, l'obtention des potentielles parties prenantes pouvant accéder aux données et la gestion des contrats. Le prochain chapitre présente une évaluation des performances du protocole en considérant plusieurs nombres d'attributs et différentes ressources.

Chapitre 5

Prototype du protocole

Sommaire

Ce cinquième chapitre est consacré à l'évaluation des performances du protocole en considérant le temps d'exécution, l'occupation mémoire et la taille des messages. Ce chapitre débute par la présentation d'un prototype du protocole utilisé pour évaluer ses performances. La méthode d'évaluation des performances est présentée en seconde section, et les résultats de l'évaluation sont présentés en troisième section. Des améliorations du prototype et de l'évaluation sont discutées en quatrième section, puis une conclusion de ce chapitre est proposée en cinquième section.

5.1 Vue globale du prototype

Le prototype du protocole [1] a été réalisé en langage C, il repose sur plusieurs bibliothèques cryptographiques pour implémenter les schémas cryptographiques utilisés dans le protocole.

Le schéma KP-ABE utilisé dans le prototype est celui fourni par la bibliothèque OpenABE [2]. Cette bibliothèque, développée par Zeutro en C++ principalement,

1. Code disponible sur le lien suivant : `https://gitlab.laas.fr/jicv_2022_security_protocol/security_protocol`
2. Bibliothèque disponible sur le lien suivant : `https://github.com/zeutro/openabe`

s'appuie sur la bibliothèque RELIC [Aranha 2022], développée en C principalement, pour effectuer les opérations cryptographiques de bas niveau. OpenABE fournit notamment une implémentation, basée sur les couplages, d'une construction d'un schéma KP-ABE à large univers [Goyal 2006]. La courbe utilisée est la courbe BN254, cependant d'autres courbes elliptiques peuvent être utilisées.

Le schéma de chiffrement symétrique utilisé dans le prototype est le schéma AES de la bibliothèque libcrypto d'OpenSSL, développée en C principalement. AES propose plusieurs modes d'opérations, nous avons choisi d'utiliser le mode CounTeR (CTR), permettant de réaliser un chiffrement à flot réduisant la taille du chiffré renvoyé. Néanmoins, d'autres modes d'opération, offrant diverses propriétés, peuvent être utilisés à la place du mode CTR. AES repose sur une clé AES et un vecteur d'initialisation pour réaliser le chiffrement et le déchiffrement. Cependant, pour simplifier le prototype, nous considérons que la clé symétrique échangée lors de l'étape 1 des scénarios du protocole est la combinaison de la clé AES et du vecteur d'initialisation. La taille de la clé AES est de 32 octets, des tailles de clé plus petites peuvent être choisies au détriment de la sécurité, la taille du vecteur d'initialisation est de 16 octets, la taille totale de la clé symétrique est donc de 48 octets.

En l'absence d'une implémentation d'un schéma de signature de groupe approprié, le schéma de signature de groupe du prototype repose sur un schéma de signature asymétrique classique de la bibliothèque libcrypto d'OpenSSL, le schéma Elliptic Curve Digital Signature Algorithm (ECDSA). Pour produire une signature, l'algorithme de signature effectue un condensat en utilisant une fonction de hachage, puis réalise la signature. Nous avons choisi la fonction de hachage SHA256 et la courbe BrainpoolP512r1 entraînant une signature de 136 octets, d'autres fonctions de hachage et courbes elliptiques peuvent êtres utilisées.

Lors des premières étapes du protocole, des nonces sont échangés afin d'initialiser la connexion, dans ce prototype leurs tailles sont fixées à 32 octets afin qu'elles soient équivalentes à celle d'une clé AES. Le reste de cette section est dédié à l'implémentation du protocole. Tout d'abord, des exemples de méthodes et fonctions fournis par les bibliothèques sont présentés. Puis, des exemples de fonctions du prototype appelant les méthodes et fonctions des bibliothèques sont donnés. Enfin, l'implémentation des étapes du protocole est décrite.

5.1.1 Bibliothèques cryptographiques

5.1.1.1 OpenABE - KP-ABE

La bibliothèque OpenABE fournit des méthodes de haut niveau permettant de manipuler les algorithmes ABE via la classe *OpenABECryptoContext*. Les prototypes de ces méthodes, issus du code source d'OpenABE (fichier d'en-tête `zcrypto_box.h`), sont donnés en Listing 5.1. OpenABE propose également un schéma CP-ABE, le constructeur permet de choisir le schéma ABE à utiliser via le paramètre *scheme_id*. Les méthodes de la classe *OpenABECryptoContext* implémentent donc les algorithmes KP-ABE définis en Chapitre 3 :

```
class OpenABECryptoContext : public OpenABECryptoContextBase {
public:
  ...
  OpenABECryptoContext(const string scheme_id,
                       bool base64encode = true);
  int generateParams();
  int keygen(const string &keyInput, const string &keyID,
             const string &authID = "", const string &GID = "");
  int encrypt(const string encInput, const string &plaintext,
              string &ciphertext);
  bool decrypt(const string &keyID, const string &ciphertext,
               string &plaintext);
  ...
};
```

Listing 5.1 – OpenABE : prototype des méthodes implémentant un schéma ABE

— *generateParams* correspond aux algorithmes de génération de clé maîtresse *abe_setup* et de clé publique *abe_pkgen* ;

— *keygen* correspond à l'algorithme de génération de clé de déchiffrement *abe_skgen*, la génération de la clé est réalisée à partir d'un arbre d'accès (*keyInput*) ;

— *encrypt* correspond à l'algorithme de chiffrement *abe_enc*, le chiffrement est réalisé à partir d'une liste d'attributs (*encInput*) ;

— *decrypt* correspond à l'algorithme de déchiffrement *abe_dec*, le déchiffrement est réalisé à partir d'une clé de déchiffrement (*keyID*) et d'un chiffré (*ciphertext*).

5.1.1.2 OpenSSL - Chiffrement symétrique

```
int EVP_EncryptInit_ex(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type,
                       ENGINE *impl, const unsigned char *key,
                       const unsigned char *iv);
int EVP_EncryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out,
                      int *outl, const unsigned char *in, int inl);
int EVP_EncryptFinal_ex(EVP_CIPHER_CTX *ctx, unsigned char *out,
                        int *outl);
```

Listing 5.2 – OpenSSL : prototype des fonctions implémentant un chiffrement symétrique

La bibliothèque libcrypto d'OpenSSL fournit des fonctions génériques permettant de choisir un schéma de chiffrement symétrique parmi plusieurs schémas lors du chiffrement et du déchiffrement. La génération de la clé symétrique n'est pas réalisée en utilisant des fonctions dédiées mais en utilisant des fonctions de générations d'aléa. À titre d'exemple, les fonctions permettant le chiffrement sont données en Listing 5.2, ces fonctions sont issues de la documentation d'OpenSSL [3] :

3. Documentation disponible sur le lien suivant : `https://www.openssl.org/docs/man3.0/`

— *EVP_EncryptInit_ex* initialise un contexte avec le schéma de chiffrement symétrique (*type*), la clé AES (*key*) et le vecteur d'initialisation (*iv*) à utiliser. Ce contexte est utilisé par les fonctions suivantes ;

— *EVP_EncryptUpdate* réalise le chiffrement symétrique ;

— lorsqu'un bourrage est nécessaire, *EVP_EncryptFinal_ex* réalise le chiffrement des données restantes à chiffrer, et complète le dernier bloc avec du bourrage.

5.1.1.3 OpenSSL - Signature

```
1 int EVP_DigestSignInit(EVP_MD_CTX *ctx, EVP_PKEY_CTX **pctx,
2                         const EVP_MD *type, ENGINE *e,
3                         EVP_PKEY *pkey);
4 int EVP_DigestSignUpdate(EVP_MD_CTX *ctx, const void *d,
5                           size_t cnt);
6 int EVP_DigestSignFinal(EVP_MD_CTX *ctx, unsigned char *sig,
7                          size_t *siglen);
```

Listing 5.3 – OpenSSL : prototype des fonctions implémentant une signature

De manière similaire, OpenSSL fournit des fonctions génériques permettant de générer une clé de signature, de signer et de vérifier un message. Le schéma de signature à utiliser est spécifié lors de la génération de la clé de signature, l'identifiant du schéma est contenu dans la clé. Cette clé contient également à la fois la clé secrète de vérification et la clé publique de signature, ainsi la même clé est donnée aux fonctions de signature et de vérification. À titre d'exemple, les fonctions permettant de signer un message sont données en Listing 5.3, ces fonctions sont également issues de la documentation d'OpenSSL :

— *EVP_DigestSignInit* initialise un contexte associé à une fonction de hachage (*type*) et une clé publique (*pkey*). Ce contexte est utilisé par les fonctions suivantes ;

— *EVP_DigestSignUpdate* produit le condensat du message en utilisant la fonction de hachage, puis le stocke dans le contexte ;

— *EVP_DigestSignFinal* réalise la signature à partir du condensat stocké dans le contexte.

5.1.2 Schémas cryptographiques

5.1.2.1 Schéma KP-ABE

Le prototype étant développé en C, les méthodes C++ fournies par OpenABE ne peuvent pas être appelées directement dans le prototype, il est alors nécessaire de réaliser des fonctions C encapsulant ces méthodes (des wrappers). Pour ce faire, un objet global *kpabe* est instancié en appelant le constructeur et en spécifiant

man3/

```
extern "C" {
  int abe_encrypt(const char *, const char *, const size_t,
                  char **, size_t *);
}

OpenABECryptoContext kpabe("KP-ABE", false);

int abe_encrypt(const char *atr_chr, const char *pt_chr,
                const size_t pt_len, char **ct_chr,
                size_t *ct_len) {
  string atr_str(atr_chr);
  string pt_str(pt_chr, pt_len);
  string ct_str;
  int ret = kpabe.encrypt(atr_str, pt_str, ct_str);
  FAIL_IF(ret != 1, err, "OpenABECryptoContext.encrypt failed\n")
  ret = 0;
  *ct_len = ct_str.length();
  *ct_chr = (char *)malloc((*ct_len+1) * sizeof(char));
  FAIL_IF(*ct_chr == NULL, err, "malloc failed\n")
  memcpy(*ct_chr, ct_str.c_str(), *ct_len+1);
  ret = 1;
err:
  return ret;
}
```

Listing 5.4 – Wrapper de la méthode encrypt d'OpenABE

le schéma KP-ABE en premier paramètre. Très peu d'opérations sont nécessaires dans les wrappers pour appeler les méthodes C++, ces méthodes manipulent des chaînes de caractères sous la forme d'objets `string` or les fonctions C manipulent des chaînes de caractères sous la forme du type `char*`, les opérations à réaliser consistent essentiellement à convertir les chaînes de caractères de type `char*` en objet `string` et inversement. À titre d'exemple, le wrapper de la méthode *encrypt* est donné en Listing 5.4, ce code est issu du fichier *crypto_abe.cpp*.

5.1.2.2 Schémas SE et GS

Les algorithmes des schémas SE et GS sont implémentés en appelant les fonctions d'OpenSSL permettant le chiffrement symétrique et la signature. À titre d'exemple, la fonction correspondant à l'algorithme de chiffrement symétrique du protocole *se_enc* est donnée en Listing 5.5, cette fonction est issue du fichier *crypto_ssl.c*. L'algorithme AES en mode CTR est spécifié à l'appel de la fonction *EVP_EncryptInit_ex* via le second paramètre *EVP_aes_256_ctr()*.

5.1.3 Étapes du protocole

La première étape d'initialisation (génération et distribution des clés), ainsi que les étapes des scénarios S1 (scénario d'envoi sécurisé des données du véhicule), S2 (scénario de lecture sécurisée des données par une partie prenante) et S3 (scénario

```c
int s_enc(const unsigned char *k, const unsigned char *iv,
          const unsigned char *pt, const size_t pt_len,
          unsigned char **ct, size_t *ct_len) {
  EVP_CIPHER_CTX *ctx = NULL;
  int ret = 0, len = 0;

  *ct = malloc(pt_len * sizeof(char));
  FAIL_IF(*ct == NULL, err, "malloc failed\n");

  ctx = EVP_CIPHER_CTX_new();
  FAIL_IF(ctx == NULL, err, "EVP_CIPHER_CTX_new failed\n")

  ret = EVP_EncryptInit_ex(ctx, EVP_aes_256_ctr(), NULL, k, iv);
  FAIL_IF(ret != 1, err, "EVP_EncryptInit_ex failed\n")

  ret = EVP_EncryptUpdate(ctx, *ct, &len, pt, pt_len);
  FAIL_IF(ret != 1, err, "EVP_EncryptUpdate failed\n")
  *ct_len = len;

  ret = EVP_EncryptFinal_ex(ctx, *ct + len, &len);
  FAIL_IF(ret != 1, err, "EVP_EncryptFinal_ex failed\n")
  *ct_len += len;
err:
  if (ctx != NULL) EVP_CIPHER_CTX_free(ctx);
  FAIL_FREE(ret != 1, *ct)

  return ret;
}
```

Listing 5.5 – Fonction réalisant le chiffrement symétrique via des appels aux fonctions d'OpenSSL

de lecture sécurisée des données par un véhicule) sont implémentées sous la forme d'une fonction. Étant donnée que les fonctions *get_access_tree* et *get_attrs* réalisent essentiellement une recherche dans un tableau, ces fonctions ont un faible coût de calcul, elles ne sont alors pas implémentées. Lorsqu'un appel à l'une de ces fonctions est réalisé, cet appel est remplacé soit par une constante, soit par une variable donnée en entrée à la fonction implémentant l'étape. Le stockage du message dans la base de données lors de l'étape 4 du scénario S1, et le traitement de la requête lors de l'étape 4 des scénarios S2 et S3 n'ont pas été implémentés également, les performances de ces opérations dépendant des outils utilisés. À titre d'exemple, la fonction correspondant à la première étape des scénarios du protocole est donnée en Listing 5.6, cette fonction est issue du fichier *protocole.c*. Elle réalise les opérations suivantes :

- la génération de la clé symétrique via la fonction *s_setup* ;
- la génération du nonce du véhicule de manière aléatoire via la fonction *gen_rand_bytes* ;
- la génération de l'attribut du centre de stockage, cependant la fonction *get_attrs* n'étant pas implémentée, une constante est alors utilisée ;

```
int p_send_1(EVP_PKEY *sig_v, AES_KEY *aes_v, unsigned char **nv,
             MSG *m1) {
  unsigned char *n1 = NULL, *c1 = NULL, *s1 = NULL, *L1 = NULL;
  size_t len = 0, n1_len = 0;
  int err_code = NO_ERR;

    // memory allocation
  n1 = malloc((AES_K_LEN + AES_IV_LEN + NONCE_LEN) * sizeof(char));
  FAIL_IF3(n1 == NULL, MALLOC_P1_ERR, "malloc failed (1): n1\n")

    // K = s_setup()
  int ret = s_setup(&aes_v->k, &len, &aes_v->iv, &len);
  FAIL_IF3(ret != 1, S_SETUP_ERR, "s_setup failed\n")
    // Nv = random()
  ret = gen_rand_bytes(nv, NONCE_LEN);
  FAIL_IF3(ret != 1, GEN_BYTES_P1_ERR, "gen_rand_bytes failed\n")
    // L1 = get_attrs(IDsc, empty)
  L1 = "sc_id:sc1";
    // C1 = abe_enc((K, Nv), L1, PK)
  memcpy(n1, aes_v->k, AES_K_LEN);
  n1_len = AES_K_LEN;
  memcpy(n1 + n1_len, aes_v->iv, AES_IV_LEN);
  n1_len += AES_IV_LEN;
  memcpy(n1 + n1_len, *nv, NONCE_LEN);
  n1_len += NONCE_LEN;

  ret = abe_encrypt(L1, n1, n1_len, (char **)&c1, &m1->c_len);
  FAIL_IF3(ret != 1, ABE_ENC_P1_ERR, "abe_encrypt failed\n")
    // S1 = gs_sign(C1, SIG_SKv)
  ret = gs_sign(sig_v, c1, m1->c_len, &s1, &m1->s_len);
  FAIL_IF3(ret != 1, GS_SIGN_P1_ERR, "gs_sign failed\n")
    // M1 = (C1,S1)
  m1->m = malloc((m1->c_len + m1->s_len) * sizeof(char));
  FAIL_IF3(m1->m == NULL, MALLOC_P1_ERR, "malloc failed (2): m1\n")

  memcpy(m1->m, c1, m1->c_len);
  m1->m_len = m1->c_len;
  memcpy(m1->m + m1->m_len, s1, m1->s_len);
  m1->m_len += m1->s_len;
err:
  FREE(n1)
  FREE(c1)
  FREE(s1)

  FAIL_FREE(err_code != NO_ERR, aes_v->k)
  FAIL_FREE(err_code != NO_ERR, aes_v->iv)
  FAIL_FREE(err_code != NO_ERR, *nv)
  FAIL_FREE(err_code != NO_ERR, m1->m)

  return err_code;
}
```

Listing 5.6 – Fonction implémentant la première étape des scénarios du protocole

— le chiffrement KP-ABE de la clé symétrique et du nonce via la fonction *abe_encrypt* à l'aide de l'attribut du centre de stockage ;

— le chiffré généré est ensuite signé en utilisant la fonction *gs_sign* ;

— le chiffré et la signature sont concaténés pour former le message M1.

5.2 Méthode d'évaluation

Le véhicule étant le composant le plus contraint du système, l'évaluation des performances du prototype a été réalisée uniquement sur une Raspberry Pi 3 B+ représentant le calculateur d'un véhicule. La Raspberry Pi est équipée d'un processeur Broadcom BCM2837B0 64 bits quadricœur ARM Cortex-A53 de 1,4 GHz et d'une mémoire vive de 1 Go. Les évaluations réalisées sont : l'évaluation du temps de calcul des étapes du protocole, l'évaluation de la mémoire consommée par chaque étape du protocole, et l'évaluation de la taille des messages.

5.2.1 Fonctions réalisant l'évaluation

```
1 int bench(int e_choice, char *st_ac, char *L2, char *m,
2           size_t m_len, char *R_m, size_t R_m_len);
```

Listing 5.7 – Prototype de la fonction bench

Les évaluations sont effectuées en utilisant une fonction nommée *bench*, cette fonction réalise les appels aux fonctions représentant chaque étape du protocole. Elle réalise ainsi les appels aux fonctions correspondant aux étapes de génération de clés, puis aux étapes du scénario d'envoi sécurisé des données du véhicule, et enfin aux étapes du scénario de lecture sécurisée des données. Le prototype de la fonction *bench* est donné en Listing 5.7, il est issu du fichier *bench_proto.c*. Cette fonction prend en entrée une valeur permettant de choisir le scénario de lecture sécurisée des données entre celui d'une partie prenante et celui d'un véhicule (`e_choice`), l'arbre d'accès utilisé pour générer la clé de la partie prenante (`st_ac`), la liste d'attributs à utiliser lors du chiffrement des données (`L2`), la donnée à chiffrer lors du scénario d'envoi (`m`), et la requête à effectuer lors du scénario de lecture (`R_m`).

Deux paramètres sont variables, le nombre d'attributs et la taille de la donnée, afin d'évaluer leurs impacts sur les performances du prototype. Ainsi, la fonction *bench* est appelée dans une fonction faisant varier ces paramètres, la fonction *launch_bench_at*. Cette fonction est illustrée en Listing 5.8, elle est issue du fichier *bench_proto.c*, son corps a été simplifié pour faciliter sa compréhension.

Cette fonction contient ainsi trois boucles imbriquées : la première boucle fait varier le nombre d'attributs de 1 à 31 par pas de 2, elle réalise donc 16 itérations ; la seconde boucle fait varier la taille de la donnée de 1 à 4096 par puissance de 2, elle réalise donc 13 itérations ; la troisième boucle permet de réaliser un certain nombre d'itérations pour chaque expérimentation dans laquelle on a fixé le nombre

```
int launch_bench_at(int read_scenario) {
  unsigned char *access_tree = NULL, *attributs = NULL,
                *data = NULL;
  char *request = "cd.kpabe";

  for (int nb_attr = 1; nb_attr <= 31; nb_attr += 2) {
    for (int data_len = 1; data_len <= 4096; data_len *= 2) {
      for (int iter = 1; iter <= NB_ITER; iter++) {
        gen_attr(&attributs, &access_tree, nb_attr);
        gen_data(&data, data_len);

        bench(read_scenario, access_tree, attributs, data,
              data_len, request, strlen(request));
      }
      write_to_file();
    }
  }
  ...
}
```

Listing 5.8 – Fonction launch_bench_at simplifiée

d'attributs et la taille des données. Ce nombre d'itérations varie en fonction de la mesure visée par l'expérimentation. Lorsqu'on réalise une expérimentation dans laquelle la taille des messages ou l'occupation mémoire sont mesurées (qui a priori, doivent être quasiment identiques pour chaque itération), nous avons fixé le nombre d'itérations à 16. En revanche, pour les expérimentations qui mesurent des durées d'exécution, nous avons fixé le nombre d'itérations à 1024 car ces durées sont plus susceptibles d'être variables. Ainsi, lorsque 1024 itérations sont réalisées, le nombre total d'itérations est de 212992, et lorsque 16 itérations sont réalisées, le nombre total d'itérations est de 3328.

Le corps de la dernière boucle imbriquée permet de générer la liste d'attributs, l'arbre d'accès et la donnée, puis de les utiliser lors de l'exécution de la fonction *bench*. La fonction *gen_attrs* génère la liste d'attributs utilisée lors du chiffrement de la donnée, et l'arbre d'accès d'une partie prenante en fonction du nombre d'attributs spécifié lors de la première boucle imbriquée. Cette fonction génère des attributs sous la forme "aXXXX" avec XXXX une valeur commençant à 0000 et augmentant jusqu'au nombre d'attributs désiré. Ainsi, lorsque quatre attributs sont requis, cette fonction renvoie une liste d'attributs sous la forme "a0000|a0001|a0002|a0003", et un arbre d'accès contenant uniquement des nœuds internes ET sous la forme "a0000 ET a0001 ET a0002 ET a0003". Ensuite, la fonction *gen_data* génère une donnée dont la valeur est aléatoire en fonction de la taille de la donnée spécifiée lors de la deuxième boucle imbriquée. Enfin la fonction *bench* est appelée, puis lorsque toutes les itérations sont effectuées, les résultats des mesures sont écrits dans un fichier de type Comma-Separated Values (csv) lors de l'exécution de la fonction *write_to_file*. Notons que la valeur et la taille de la requête ne varient pas, cependant nous verrons lors des résultats de l'évaluation que l'impact de la requête est minimal

sur les performances.

5.2.2 Évaluation du temps d'exécution des étapes

Le temps d'exécution en milliseconde de chaque étape du protocole a été évalué. Afin d'avoir la mesure la plus précise du temps d'exécution, les mesures ont été effectuées en utilisant le compteur de performance du processeur, correspondant au nombre de cycles d'horloge du processeur qui se sont déroulés, puis le temps d'exécution a été calculé en milliseconde en utilisant la fréquence du processeur.

Afin de réduire la variation du nombre de cycle d'horloge, le processus a été isolé sur un seul cœur via la commande *cset shield*. La fréquence du processeur a été réglée à son maximum pour également réduire la variation du nombre de cycles d'horloge ainsi que pour pouvoir calculer le temps d'exécution.

```
1 asm volatile("mrc p15,0,%0,c9,c13,0":"=r"(r));
```

Listing 5.9 – Instruction assembleur permettant de récupérer la valeur du compteur de performance du processeur (armv7)

Pour récupérer la valeur du compteur de performance, nous souhaitons avoir le moins de surcharge possible, ainsi une instruction assembleur récupérant la valeur de ce compteur est utilisée au lieu de fonctions fournies par des bibliothèques entraînant plus d'opérations, d'appels systèmes, etc. Le compteur de cycles du processeur est un compteur 32 bits, incrémenté à chaque cycle d'horloge, il peut également être remis à zéro suite à une réinitialisation. La valeur du compteur peut être récupérée via l'instruction `mrc`[4]. Cette instruction est représentée en Listing 5.9, elle est issue du fichier *arm_reg_cnt.h*.

5.2.3 Évaluation de la mémoire maximale consommée par les étapes

Les tailles maximales en kilo-octet des régions de mémoire du tas et de la pile de l'étape 1 et de l'étape 3 sont évaluées. Les mesures ont été effectuées à l'aide de l'outil Massif de Valgrind, cet outil profile les tailles du tas et de la pile.

```
valgrind --tool=massif --heap=yes --stacks=yes --time-unit=ms
  --peak-inaccuracy=0.0 --massif-out-file=massif.out <binaire> ;
  ms_print massif.out
```

Listing 5.10 – Profilage de la mémoire via l'outil Massif de Valgrind

L'outil Massif de Valgrind réalise régulièrement des captures de l'occupation de la pile et du tas. Ainsi, pour réaliser cette évaluation uniquement, le corps de la

4. Plus d'informations disponibles à la Section B4.1.113 du lien suivant :
`https://documentation-service.arm.com/static/5f8db1f7f86e16515cdba551`

fonction *bench* a été modifié en rajoutant un temps d'attente de 10 secondes avant les étapes 1 et 3, et un temps d'attente de 15 secondes après les étapes 1 et 3. De la sorte, nous pouvons déduire dans quelle étape du protocole Valgrind se situe. La commande à exécuter est donnée en Listing 5.10.

```
n  time(ms) total(B) useful-heap(B) extra-heap(B) stacks(B)
28 7,613    282,760  214,280        66,880        1,600
29 17,650   274,504  208,040        65,448        1,016
30 17,898   285,112  209,893        65,755        9,464
31 18,310   279,312  210,244        65,876        3,192
32 18,722   283,568  214,560        66,912        2,096
33 19,134   283,568  214,560        66,912        2,096
34 19,340   283,672  214,560        66,912        2,200
35 19,649   279,376  211,342        66,026        2,008
36 34,677   275,336  208,827        65,493        1,016
```

Listing 5.11 – Extrait du fichier de sortie de l'outil Massif de Valgrind

Enfin, une fois la sortie obtenue, nous pouvons l'analyser pour récupérer les valeurs maximales de la pile et du tas. À titre d'exemple, une partie des résultats correspondant à l'étape 1 est représentée en Listing 5.11. Nous pouvons ainsi clairement voir les sauts de 10 et 15 secondes délimitant l'étape 1, via les lignes 29 et 35, puis nous récupérons manuellement la taille maximale du tas via la colonne *useful-heap*, et la taille maximale de la pile via la colonne *stacks*.

Par ailleurs, pour cette évaluation, la taille de la donnée est fixée à 64 octets pour se simplifier la tâche, étant donnée que cette évaluation est réalisée manuellement. Ainsi cette évaluation a été faite sur 256 itérations.

5.2.4 Évaluation de la taille des messages

Bien que les échanges de messages sur le réseau ne sont pas strictement implémentés, l'impact sur les métriques du réseau (bande passante du réseau) est évalué. Pour cela, la taille en octets des messages produits après l'exécution de chaque étape est évaluée.

5.3 Résultats de l'évaluation

Plusieurs graphes sont présentés dans ce qui suit, dans lesquels un point représente une valeur mesurée ou calculée sur la base des valeurs mesurées, et une courbe représente la courbe déterminée par régression linéaire.

5.3.1 Évaluation du temps d'exécution des étapes

Les durées d'exécution des étapes 1 et 3 du scénario S1 sont d'abord présentées, ces étapes sont les plus critiques étant réalisées en ligne (c'est-à-dire lors d'un trajet d'un véhicule). Les durées d'exécution des étapes 3 et 5 du scénario S3 sont données à titre indicatif, ces étapes sont moins critiques étant réalisées hors ligne (c'est-à-dire

lorsque le véhicule ne réalise pas de trajet) et que le conducteur souhaite récupérer ses données.

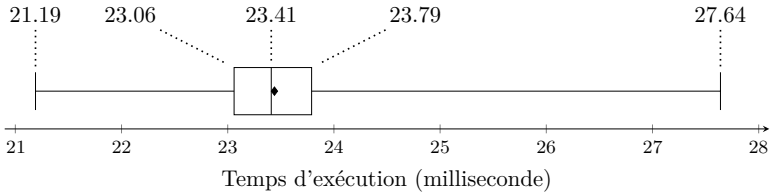

FIGURE 5.1 – Temps d'exécution de l'étape 1 (S1 et S3)

Scénario d'Envoi et de Lecture - Étape 1 Les opérations réalisées lors de cette étape consistent essentiellement à générer une clé symétrique et un nonce, puis à les chiffrer avec KP-ABE en utilisant l'attribut du centre de stockage, et enfin à signer le chiffré. Un seul attribut étant utilisé et les tailles de la clé symétrique et du nonce étant fixes, cette étape ne dépend donc pas du nombre d'attributs et de la taille de la donnée. Le temps d'exécution a été calculé sur la totalité des 212992 itérations, il est représenté sous forme d'un diagramme en boîte en Figure 5.1. Nous pouvons ainsi observer que le temps d'exécution de l'étape 1 peut être considéré constant et que le temps d'exécution absolu, de 21.19 ms au minimum à 27.64 ms au maximum, reste raisonnable dans un scénario de véhicule connecté.

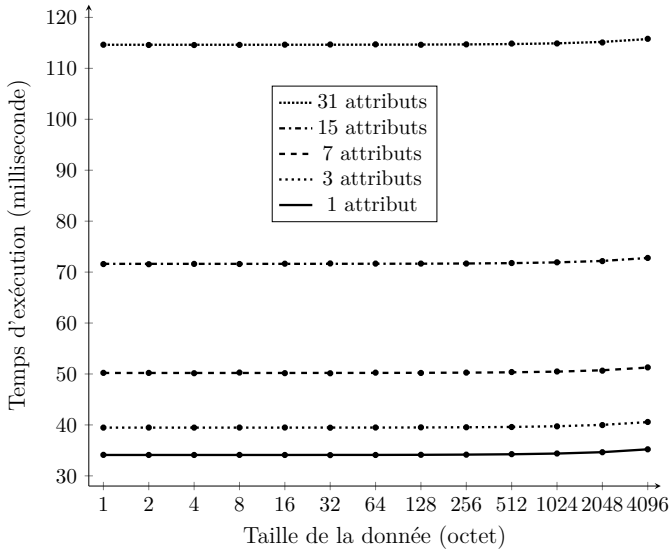

FIGURE 5.2 – Moyenne du temps d'exécution de l'étape 3 (S1) en fonction de la taille de la donnée à chiffrer pour 1, 3, 7, 15 et 31 attributs

Scénario d'Envoi - Étape 3 Cette étape consiste à vérifier la signature du centre de stockage, à déchiffrer le message reçu, à vérifier que le nonce du véhicule renvoyé

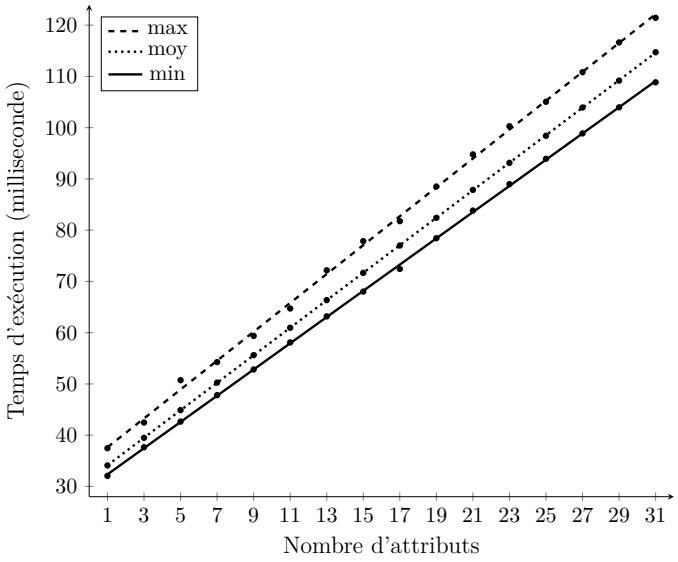

FIGURE 5.3 – Maximum, moyenne et minimum du temps d'exécution de l'étape 3 (S1) en fonction du nombre d'attributs pour une donnée à chiffrer de 64 octets

par le centre de stockage est le même que celui envoyé lors de l'étape 1, à chiffrer une donnée avec ABE en utilisant une liste d'attributs, puis à chiffrer le chiffré ABE et le nonce du centre de stockage avec AES, et enfin à signer le chiffré AES. Ainsi, le temps d'exécution de l'étape 3 varie selon le nombre d'attributs et la taille de la donnée. Cette étape effectuant régulièrement des chiffrements ABE, elle est la plus critique en temps d'exécution.

L'impact de la taille de la donnée sur le temps d'exécution de l'étape 3 est tout d'abord évalué. La moyenne du temps d'exécution de l'étape 3 en fonction de la taille de la donnée, dans le cas de 1, 3, 7, 15 et 31 attributs, est représentée en Figure 5.2. Nous pouvons observer que le temps d'exécution est une fonction quasi constante de la taille de la donnée à envoyer, la taille de la donnée n'a donc pas d'impact significatif sur le temps d'exécution de l'étape 3. Cela s'explique par le fait que les schémas ABE de la bibliothèque OpenABE implémentent un mécanisme d'encapsulation de clé : le chiffrement ABE consiste à chiffrer une clé AES, cette clé est utilisée pour chiffrer le clair. Le temps de chiffrement AES est faible par rapport au reste des opérations effectuées lors de l'étape 3, l'augmentation de la taille de la donnée n'a alors pas d'impact significatif sur le temps d'exécution total de l'étape 3. Dans ce qui suit, nous avons choisi de nous concentrer sur une donnée de 64 octets, ce qui est réaliste dans le contexte d'une donnée qui peut être régulièrement envoyée par des véhicules connectés, et sur une donnée de 4096 octets afin d'évaluer les performances dans une situation volontairement pessimiste.

L'impact du nombre d'attributs sur le temps d'exécution de l'étape 3 est ensuite évalué. Le minimum, le maximum, la moyenne et la médiane du temps d'exécution de l'étape 3, en fonction du nombre d'attributs, ainsi que la fréquence de l'étape 3 calculée à partir de la moyenne, respectivement dans le cas d'une donnée de 64 octets

TABLE 5.1 – Minimum, maximum, moyenne et médiane du temps d'exécution en milliseconde, et fréquence en hertz de l'étape 3 calculée à partir de la moyenne (S1), en fonction du nombre d'attributs pour une donnée à chiffrer de 64 octets et de 4096 octets

	64 octets					4096 octets				
Attribut	Min	Max	Moy	Med	Freq	Min	Max	Moy	Med	Freq
1	32.06	37.46	34.1	34.08	29	33.61	38.37	35.21	35.18	28
3	37.65	42.47	39.49	39.45	25	38.9	43.38	40.58	40.52	24
5	42.65	50.73	44.89	44.81	22	43.68	50.16	45.89	45.8	21
7	47.81	54.26	50.26	50.19	19	48.76	56.74	51.29	51.26	19
9	52.83	59.37	55.62	55.5	17	54.07	60.93	56.67	56.59	17
11	58.1	64.72	60.96	60.89	16	58.81	66.43	62.02	61.94	16
13	63.18	72.17	66.35	66.26	15	64.15	72.35	67.37	67.29	14
15	68.02	77.87	71.67	71.55	13	69.55	79.16	72.78	72.73	13
17	72.44	81.76	77.0	76.88	12	74.63	83.01	78.14	78.05	12
19	78.43	88.51	82.41	82.28	12	79.26	88.62	83.49	83.31	11
21	83.82	94.78	87.86	87.83	11	84.67	94.17	88.88	88.77	11
23	88.98	100.28	93.14	93.03	10	89.07	102.84	94.26	94.2	10
25	93.93	105.05	98.42	98.32	10	95.15	106.38	99.64	99.57	10
27	98.88	110.83	103.93	103.87	9	100.72	111.85	105.11	105	9
29	103.97	116.63	109.18	109.06	9	105.11	118.26	110.39	110.36	9
31	108.85	121.43	114.71	114.59	8	109.34	122.44	115.8	115.73	8

et 4096 octets sont donnés en Table 5.1. Le maximum, la moyenne et le minimum du temps d'exécution de l'étape 3 en fonction du nombre d'attributs dans le cas d'une donnée de 64 octets sont donnés en Figure 5.3. Cette table et cette figure montrent que le temps d'exécution de l'étape 3 augmente linéairement en fonction du nombre d'attributs, et que le temps d'exécution absolu de l'étape 3, de 32,06 ms au minimum pour une donnée de 64 octets et 1 attribut à 122,44 ms au maximum pour une donnée de 4096 octets et 31 attributs, reste également raisonnable dans un scénario de véhicule connecté. Dans le cas de 31 attributs et d'une donnée de 64 octets, le temps d'exécution de l'étape 3 est en moyenne de 114,71 ms, permettant environ 8 étapes par seconde, ce qui est parfaitement acceptable dans un contexte réel. En extrapolant sur ces données, pour atteindre un temps d'exécution d'une seconde, le nombre d'attributs devrait être fixé à 361. Ce nombre d'attributs est suffisamment important pour inclure les scénarios complexes d'une mise en œuvre de la législation dans un cas réel. Dans l'ensemble, ces expériences montrent que le temps d'exécution de l'étape 3 est acceptable et ne peut pas entraver de manière significative les performances du calculateur du véhicule réalisant les opérations.

Scénario de Lecture - Étape 3 Cette étape consiste à vérifier la signature du centre de stockage, à déchiffrer le message reçu, à vérifier que le nonce du véhicule renvoyé par le centre de stockage est le même que celui envoyé en étape 1, puis

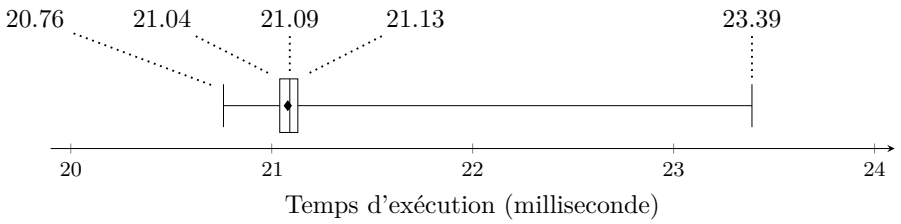

FIGURE 5.4 – Temps d'exécution en milliseconde de l'étape 3 (S3)

à chiffrer le nonce du centre de stockage et une requête avec AES, et à signer le chiffré AES. Le temps d'exécution de cette étape varie selon la taille de la requête, cependant le chiffrement réalisé étant un chiffrement AES, il n'a qu'un faible impact sur le temps d'exécution comme nous l'avons vu précédemment. Pour cette raison, la taille de la requête n'a pas été modifiée lors de l'évaluation de cette étape. Le temps d'exécution a été mesuré sur les 212992 itérations, le minimum, le maximum, la moyenne et la médiane du temps d'exécution sont donnés en Figure 5.4. Le temps d'exécution absolu, de 20.76 ms à 23.39 ms, est comparable à celui de l'étape 1.

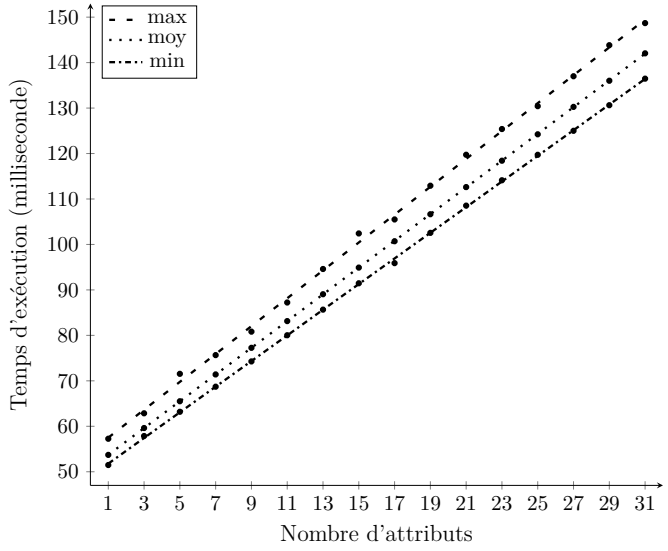

FIGURE 5.5 – Maximum, moyenne et minium du temps d'exécution de l'étape 5 (S3) en fonction du nombre d'attributs pour une donnée à chiffrer de 64 octets

Scénario de Lecture - Étape 5 Cette étape consiste à vérifier la signature du centre de stockage et à déchiffrer la réponse du centre de stockage. Pour cette étape, nous évaluons le temps d'exécution en fonction du nombre d'attributs uniquement. En effet, nous avons vu précédemment que l'implémentation d'ABE dans la bibliothèque OpenABE réalise un mécanisme d'encapsulation de clé, ainsi le déchiffrement est réalisé en deux parties, un déchiffrement ABE permettant de récupérer la clé AES encapsulée, suivi d'un déchiffrement AES des données. Ce dernier déchiffrement a

un coût très faible en temps de calcul, l'impact de la taille de la donnée sur le temps d'exécution est donc négligeable.

TABLE 5.2 – Minimum, maximum, moyenne et médiane du temps d'exécution en milliseconde de l'étape 5 (S3), en fonction du nombre d'attributs pour une donnée à chiffrer de 64 octets et de 4096 octets

	64 octets				4096 octets			
Attribut	Min	Max	Moy	Med	Min	Max	Moy	Med
1	51.48	57.26	53.71	53.67	54.07	58.68	55.8	55.8
3	57.86	62.86	59.62	59.6	59.84	64.72	61.66	61.64
5	63.19	71.54	65.51	65.42	65.28	71.64	67.52	67.45
7	68.71	75.66	71.41	71.35	70.75	78.55	73.43	73.35
9	74.27	80.82	77.26	77.17	76.56	83.3	79.32	79.27
11	80.03	87.23	83.14	83.03	82.18	89.72	85.17	85.12
13	85.66	94.6	89.05	88.96	87.5	96.1	91.04	91
15	91.46	102.42	94.91	94.81	93.29	103.16	96.97	96.91
17	95.89	105.5	100.71	100.6	99.12	108.1	102.87	102.77
19	102.55	112.92	106.65	106.56	104.19	113.83	108.73	108.58
21	108.54	119.73	112.62	112.56	110.26	119.78	114.61	114.57
23	114.12	125.39	118.42	118.33	115.26	128.79	120.51	120.45
25	119.72	130.42	124.23	124.15	121.64	132.85	126.4	126.32
27	125	137	130.26	130.22	127.95	138.81	132.39	132.27
29	130.63	143.83	136.01	135.9	132.9	145.95	138.19	138.15
31	136.47	148.69	142.04	141.95	137.44	150.89	144.13	144.02

Le minimum, le maximum, la moyenne et la médiane du temps d'exécution de l'étape 5, en fonction du nombre d'attributs, respectivement dans le cas d'une donnée de 64 octets et de 4096 octets sont donnés en Table 5.2. Le maximum, la moyenne et le minimum du temps d'exécution de l'étape 5 en fonction du nombre d'attributs dans le cas d'une donnée de 64 octets sont donnés en Figure 5.5. Cette table et cette figure montrent que le temps d'exécution de l'étape 5 augmente linéairement en fonction du nombre d'attributs, et que le temps d'exécution absolu de l'étape 5, de 51,48 ms au minimum pour une donnée de 64 octets et 1 attribut à 150,89 ms au maximum pour une donnée de 4096 octets et 31 attributs, est légèrement plus long tout en restant comparable au temps d'exécution de l'étape 3 du scénario S1. Cependant, étant donné que le véhicule déchiffre un message en utilisant une clé de déchiffrement contenant un seul attribut, nous pourrions nous attendre à un temps de d'exécution plus faible et quasiment constant. Cela s'explique par le fait que l'implémentation d'ABE de la bibliothèque OpenABE réalise des opérations supplémentaires pour garantir un niveau de sécurité élevé. L'une de ces opérations consiste, lors du déchiffrement, à rechiffrer la donnée en utilisant les attributs utilisés lors du chiffrement. Ainsi, l'augmentation linéaire du temps d'exécution de l'étape 5 en fonction du nombre d'attributs provient du temps de chiffrement tel que nous

avons pu l'observer lors de l'évaluation de l'étape 3 du scénario S1. Notons que cette étape est réalisée hors ligne, ce temps d'exécution impacte principalement l'attente du conducteur pour la récupération de ses données, nous jugeons alors ce temps très faible et acceptable pour le conducteur.

5.3.2 Évaluation de la mémoire maximale consommée par les étapes

Seule l'occupation mémoire des étapes 1 et 3 du scénario S1 est évaluée, ces étapes étant les plus critiques et devant être exécutées en ligne.

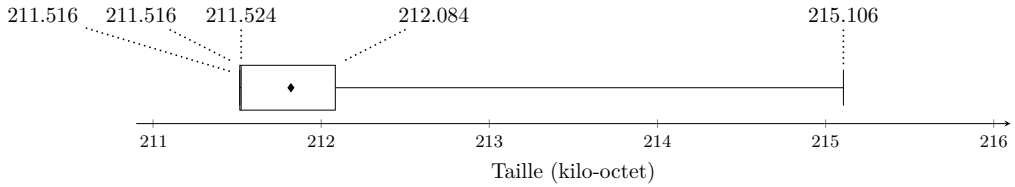

FIGURE 5.6 – Taille maximale du tas de l'étape 1 (S1 et S3)

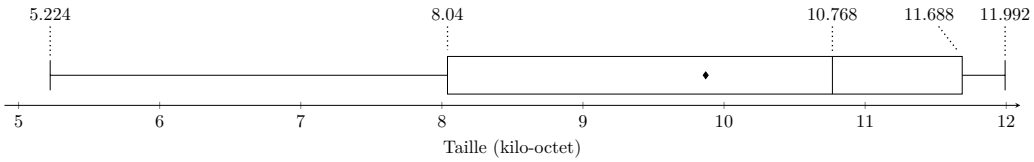

FIGURE 5.7 – Taille maximale de la pile de l'étape 1 (S1 et S3)

Scénario d'Envoi et de Lecture - Étape 1 Cette étape ne dépendant pas du nombre d'attributs, nous avons évalué les tailles maximales du tas et de la pile sur la totalité des 256 itérations. Le minimum, le maximum, la moyenne et la médiane des tailles maximales du tas et de la pile sont donnés en Figure 5.6 et en Figure 5.7. Nous pouvons observer que la taille maximale de la pile ne dépasse pas 20 kilo-octets, et que la taille maximale du tas ne dépasse pas 220 kilo-octets. Cette taille représente moins de 0,1% de l'ensemble de la mémoire vive disponible sur la Raspberry Pi, en faisant l'hypothèse que le calculateur du véhicule réalisant cette étape dispose d'au moins 1 gigaoctet de mémoire, nous pouvons considérer que la mémoire consommée est tout à fait acceptable.

Scénario d'Envoi - Étape 3 La Figure 5.8 représente la moyenne de la taille maximale du tas et de la pile de l'étape 3 en fonction du nombre d'attributs, dans le cas d'une donnée de 64 octets. Cette figure montre que la taille maximale de la pile est presque constante et ne dépasse pas 50 kilo-octets, et que la taille maximale du tas augmente linéairement en fonction du nombre d'attributs et ne dépasse pas 250 kilo-octets pour des attributs entre 1 et 31. De la même manière, nous jugeons

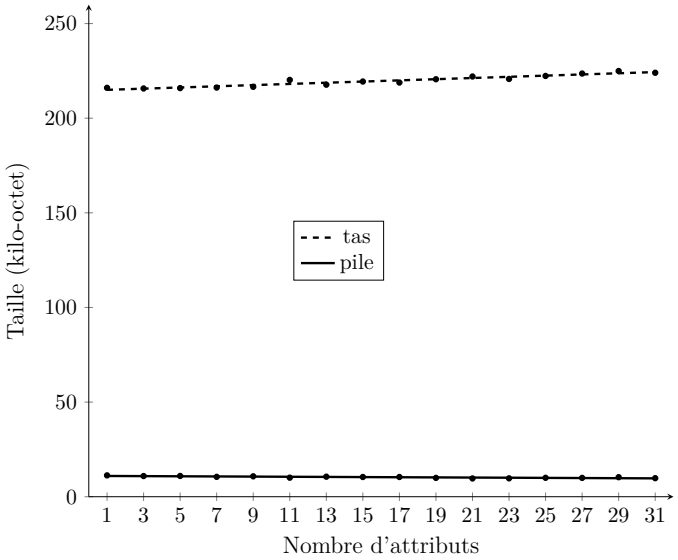

FIGURE 5.8 – Moyenne de la taille du tas et de la pile de l'étape 3 (S1) en fonction du nombre d'attributs pour une donnée de 64 octets

que la mémoire consommée est acceptable au regard de la mémoire totale de la Raspberry Pi.

5.3.3 Évaluation de la taille des messages

L'évaluation de la taille des messages est réalisée en considérant le message M1 émis en fin de l'étape 1 des scénarios S1 et S3, et le message M3 émis en fin de l'étape 3 du scénario S1.

Scénarios d'Envoi et de Lecture - Message M1 Le message M1 contient un chiffré ABE et une signature. Le chiffré ABE est associé à un attribut, le chiffré correspondant à une clé AES et un nonce. Ainsi, la taille du message ne varie pas, elle est de 707 octets.

TABLE 5.3 – Taille du message M3 en octet en fonction du nombre d'attributs pour une donnée de 64 octets et de 4096 octets

Attribut	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
64 octets	723	867	1007	1149	1292	1431	1573	1716	1856	1995	2140	2279	2420	2563	2703	2843
4096 octets	6104	6244	6388	6527	6667	6811	6951	7091	7235	7377	7516	7659	7799	7939	8083	8224

Scénarios d'Envoi - Message M3 Cette évaluation est dédiée à la taille des messages produits par l'étape 3, et, en conséquence directe, de la bande passante réseau nécessaire pour envoyer les messages. Le message M3 contient un chiffré AES

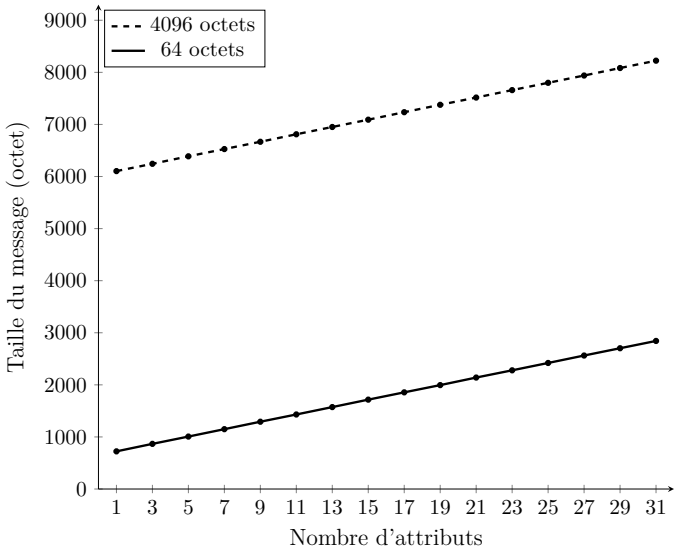

FIGURE 5.9 – Taille du message M3 en fonction du nombre d'attributs pour une donnée de 64 octets et de 4096 octets

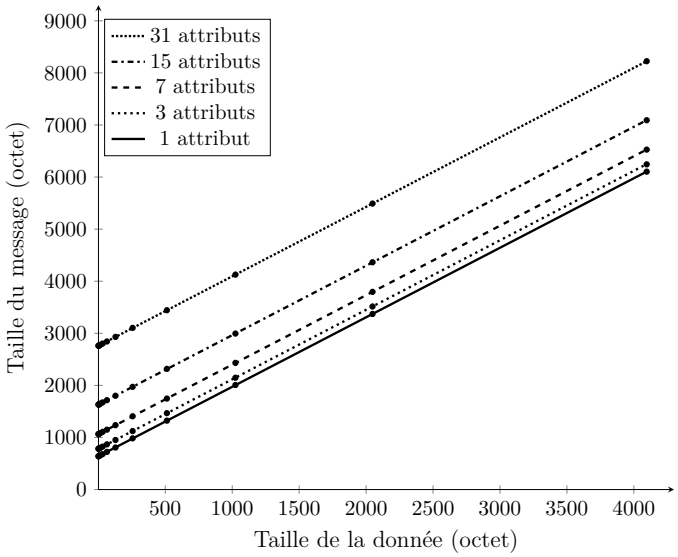

FIGURE 5.10 – Taille du message M3 en fonction de la taille de la donnée pour 1, 3, 7, 15 et 31 attributs

et une signature, le chiffré AES correspond à un nonce et à un chiffré ABE chiffré avec une liste d'attributs, ce chiffré ABE correspond à une donnée. La Table 5.3 et la Figure 5.9 représentent la taille du message produit par l'étape 3 en fonction du nombre d'attributs, dans le cas d'une donnée de 64 et 4096 octets, et la Figure 5.10 représente la taille du message produit par l'étape 3 en fonction de la taille de la donnée, dans le cas de 1, 3, 7, 15 et 31 attributs. Cette table et ces figures montrent que 1) la taille du message augmente linéairement en fonction du nombre

d'attributs ; 2) la taille du message augmente linéairement en fonction de la taille de la donnée ; et 3) pour une donnée de 64 octets, l'expansion du chiffré est de 11,3 pour 1 attribut, et de 44,4 pour 31 attributs. En considérant un chiffré de 2843 octets correspondant à une donnée de 64 octets avec 31 attributs, et en considérant, selon les évaluations précédentes, 8 exécutions de l'étape 3 par seconde, l'envoi des messages nécessite environ 181,95 kilobits par seconde de bande passante réseau. Le site web de l'Union Internationale des Télécommunications [ITU 2011] précise que la 3G fournit une vitesse minimale de 348 kilobits par seconde pour un véhicule en mouvement, ce qui signifie que la bande passante requise pour le protocole peut être supportée par les connexions de véhicules 3G, 4G et 5G.

Toutefois, avec un chiffré de 8224 octets correspondant à une donnée de 4096 octets et 31 attributs, le même envoi de messages nécessite une largeur de bande réseau d'environ 526,34 kilobits par seconde, ce qui ne devrait pas être supporté par une connexion 3G. Cette estimation a été réalisée en considérant que les données sont envoyées à la vitesse maximale, ce qui n'est probablement pas le cas pour des données de 4096 octets. Ces messages peuvent plutôt correspondre à des lots de données envoyés moins fréquemment. Si l'on considère un taux d'envoi plus faible, par exemple en exécutant l'étape 3 chaque seconde avec des données de 4096 octets et 31 attributs, la bande passante requise devient 65,79 kilobits par seconde, ce qui est conforme à toute connexion de véhicule 3G, 4G et 5G.

Pour finaliser l'évaluation de la taille des messages, on considère le pire cas dans lequel le véhicule émet l'ensemble des messages échangés sur le bus CAN. La bande passante du bus CAN étant de 1 mégabit par seconde [ISO 2003], on considère qu'un message de 1 mégabit avec 31 attributs doit être envoyé chaque seconde. La bande passante requise pour la connexion réseau est alors de 1,4 mégabits par seconde, ce qui ne devrait pas être supporté par une connexion 3G, mais peut l'être par des connexions 4G ou 5G.

5.4 Discussion

L'évaluation des performances peut être améliorée en considérant : une bibliothèque optimisée, la plateforme d'évaluation et les contraintes temps réels.

Bibliothèque optimisée L'implémentation du schéma KP-ABE repose sur la bibliothèque OpenABE qui n'est pas une bibliothèque optimisée. Cette bibliothèque utilise de nombreuses classes pour implémenter les algorithmes ABE, ce qui entraîne de nombreuses allocations dynamiques, et donc d'appels système, ainsi que de nombreux appels de fonctions pour exécuter les algorithmes ABE. Une bibliothèque plus optimisée reposerait uniquement sur de l'allocation statique et implémenterait les algorithmes en utilisant les fonctions bas niveaux directement (telles que les fonctions de la bibliothèque RELIC). Utiliser une telle bibliothèque permettrait de réduire le temps d'exécution et la mémoire maximale nécessaire à l'exécution des étapes du protocole.

Plate-forme d'évaluation N'ayant pas à notre disposition un calculateur du véhicule, l'évaluation a été réalisée sur une Raspberry Pi. Pour avoir une estimation plus fidèle du temps d'exécution des étapes, l'évaluation pourrait être réalisée sur l'un de ses calculateurs.

Contraintes temps réels Les applications exécutées sur les calculateurs des véhicules sont des applications temps réels, disposant d'un certain délai pour s'exécuter. Dans nos travaux nous n'avons pas pu considérer cette contrainte car elle n'était pas à notre disposition. Cependant, une évaluation plus précise nécessiterait de prendre en compte ce slot d'exécution.

5.5 Conclusion

Une évaluation des performances du protocole a été présentée dans ce chapitre, cette évaluation a été réalisée sur un prototype du protocole, elle s'est concentrée sur le temps d'exécution et l'occupation mémoire des étapes du protocole, ainsi que sur la taille des messages émis. Le prototype repose sur les bibliothèques OpenABE et OpenSSL pour implémenter les schémas du protocole. L'évaluation des performances du protocole est réalisée en faisant varier le nombre d'attributs et la taille de la donnée, puis plusieurs itérations sont réalisées afin d'observer la variabilité des mesures pour un des paramètres donnés. Le composant le plus contraint du système étant le calculateur du véhicule, l'évaluation a été réalisée sur une Raspberry Pi aux ressources semblables à celles d'un calculateur. Les résultats de l'évaluation ont montré que le temps d'exécution des étapes permet d'exécuter l'étape la plus critique, l'étape 3, environ 8 fois par seconde, ce qui nous semble acceptable dans un scénario d'envoi de données. En supposant que le calculateur du véhicule soit doté d'une mémoire de 1 gigaoctet, les résultats montrent que l'étape 3 ne dépasse pas 250 kilo-octets ce qui permet de respecter cette contrainte. Enfin, les résultats montrent que lorsqu'un message de grande taille (4096 octets) doit être envoyé régulièrement, les bandes passantes des réseaux actuels sont suffisamment larges pour supporter cet envoi, et dans un scénario pire cas dans lequel toutes les données devraient être envoyées, les réseaux sans fil un peu plus évolués seraient en mesure de supporter la charge. Dans l'ensemble, les résultats de l'évaluation montrent que la consommation de ressources, en termes d'occupation de la mémoire, de temps d'exécution et de bande passante, est acceptable dans le contexte d'un calculateur embarqué dans un véhicule. Les pistes d'améliorations de l'évaluation se concentrent sur l'utilisation d'une bibliothèque plus optimisée pour implémenter les schémas cryptographiques, l'utilisation d'une plateforme ciblant plus précisément les ressources d'un calculateur du véhicule, et la prise en compte des contraintes temps réel précises. Bien que le temps d'exécution des opérations des parties prenantes n'a pas été évalué, elles vont néanmoins devoir réaliser de nombreux déchiffrement KP-ABE lors de l'exécution du système. Ainsi, pour diminuer leur charge et le temps de calcul de leurs étapes du protocole, le prochain chapitre présente des optimisations du déchiffrement KP-ABE.

Chapitre 6

Optimisation du déchiffrement KP-ABE

Sommaire

Ce sixième chapitre est consacré à la présentation et à l'évaluation d'optimisations réalisées sur deux schémas KP-ABE : le schéma Large Universe de la bibliothèque OpenABE ainsi qu'un schéma Small Universe afin de déterminer si ce schéma est préférable au schéma Large Universe dans notre contexte. Ce chapitre débute par la présentation de préliminaires mathématiques nécessaires pour les sections suivantes, le contenu de cette section est issu de la documentation de la bibliothèque OpenABE. Les optimisations des schémas KP-ABE reposent sur des pré-calculs, ces pré-calculs

sont présentés en seconde section. L'application des pré-calculs aux schémas KP-ABE est présentée en troisième section. L'évaluation expérimentale des optimisations est présentée en quatrième section. Des optimisations supplémentaires sont discutées en cinquième section, puis une conclusion de ce chapitre est proposée en dernière section.

6.1 Notions préliminaires

6.1.1 Couplage bilinéaire

Soit $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$, trois groupes cycliques d'ordre premier p. Une application $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ est un **couplage bilinéaire** si elle satisfait les propriétés suivantes :

— bilinéarité : $\forall a, b \in \mathbb{Z}_p, \forall g_1 \in \mathbb{G}_1, \forall g_2 \in \mathbb{G}_2, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;

— non-dégénérescence : si g_1 et g_2 sont respectivement des générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$, alors $e(g_1, g_2)$ est un générateur de $\mathbb{G}_T$;

— efficacité : il existe une fonction calculable efficacement qui, $\forall g_1 \in \mathbb{G}_1, \forall g_2 \in \mathbb{G}_2$, calcule $e(g_1, g_2)$.

De la bilinéarité découle la propriété suivante :

$$\forall a_1, \ldots, a_n, b \in \mathbb{Z}_p, \prod_{i=1}^{n} e(g_1^{a_i}, g_2^b) = e(\prod_{i=1}^{n} g_1^{a_i}, g_2^b)$$

La preuve est donnée ci-dessous :

$$\prod_{i=1}^{n} e(g_1^{a_i}, g_2^b) = \prod_{i=1}^{n} e(g_1, g_2)^{a_i b} = e(g_1, g_2)^{b \sum_{i=1}^{n} a_i} = e(g_1^{\sum_{i=1}^{n} a_i}, g_2^b) = e(\prod_{i=1}^{n} g_1^{a_i}, g_2^b)$$

De manière similaire, la propriété suivante est également vraie :

$$\forall a_1, \ldots, a_n, b \in \mathbb{Z}_p, \prod_{i=1}^{n} e(g_1^b, g_2^{a_i}) = e(g_1^b, \prod_{i=1}^{n} g_2^{a_i})$$

6.1.2 Générateur admissible de couplage bilinéaire

L'algorithme **BSetup** prend en entrée un paramètre de sécurité 1^τ et génère les paramètres d'un groupe bilinéaire (p, g_1, g_2, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, e) tel que p est premier dans $O(2^\tau)$, $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ sont des groupes cycliques d'ordre p où g_1 génère $\mathbb{G}_1$, g_2 génère $\mathbb{G}_2$ et $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ est un couplage bilinéaire.

En particulier, dans le cas d'OpenABE, $|\mathbb{G}_1| < |\mathbb{G}_2| < |\mathbb{G}_T|$, cela impacte les performances des opérations réalisées sur des éléments de ces groupes, ainsi une exponentiation / multiplication / division dans $\mathbb{G}_1$ est en moyenne plus rapide que dans $\mathbb{G}_2$ qui est également en moyenne plus rapide que dans $\mathbb{G}_T$.

6.1.3 Arbre d'accès

Soit T un **arbre d'accès**. Chaque nœud interne de T représente une porte à seuil, décrite par ses enfants et une valeur seuil. Si num_x représente le nombre d'enfants d'un nœud interne x de T, et k_x représente sa valeur seuil, alors $1 \leq k_x \leq num_x$. Lorsque $k_x = 1$, la porte à seuil est une porte OU, lorsque $k_x = num_x$, la porte à seuil est une porte ET. Chaque nœud feuille x correspond à un attribut, associé à une valeur seuil $k_x = 1$. Pour faciliter l'utilisation des arbres d'accès, nous définissons les fonctions suivantes. Nous désignons le parent du nœud x par $parent(x)$. L'arbre d'accès T définit également un ordre entre les enfants de chaque nœud, c'est-à-dire que les enfants d'un nœud x sont numérotés de 1 à num_x. Nous désignons l'ordre d'un nœud x par $index(x)$. Nous désignons l'ensemble des enfants d'un nœud x par $child(x)$. Nous désignons l'ensemble des feuilles d'un arbre d'accès T par $leaf(T)$.

6.1.4 Satisfaction d'un arbre d'accès

Soit T un arbre d'accès possédant une racine r. Nous désignons par T_x le sous-arbre de T possédant une racine au nœud x. Ainsi, T et T_r sont équivalents. Si un ensemble d'attributs γ satisfait l'arbre d'accès T_x, ceci est dénoté par $T_x(\gamma) = 1$, le cas contraire est dénoté par $T_x(\gamma) = 0$. Un algorithme récursif calculant $T_x(\gamma)$ est implémenté dans OpenABE comme suit :

— Si x est un nœud interne, $T_{x'}(\gamma)$ est évalué pour tous les enfants x' du nœud x. $T_x(\gamma)$ retourne 1 si et seulement si au moins k_x enfants retournent aussi 1 ;

— Si x est un nœud feuille, alors $T_x(\gamma)$ retourne 1 si et seulement si $x \in \gamma$.

Dans cette implémentation, l'algorithme renvoie également un sous-ensemble minimal d'attributs de γ nécessaire pour satisfaire T.

6.1.5 Partage de secret

Un schéma de **partage de secret** permet de partager un secret en n parts de telle sorte qu'il est nécessaire de posséder k parts (avec $k \leq n$) pour récupérer le secret, et que la possession de $k-1$ parts ne divulgue aucune information sur le secret [Shamir 1979]. Nous définissons dans ce qui suit un schéma de partage de secret suivant un arbre d'accès T, ce schéma est composé de deux algorithmes :

— **computeSecretShares**$(s, T) \rightarrow \lambda$. Cet algorithme prend en entrée un secret $s \in \mathbb{Z}_p$, un arbre d'accès T et calcule les parts secrètes de s selon l'arbre d'accès T_r. Cet algorithme choisit en premier un polynôme q_x pour chaque nœud interne x de l'arbre d'accès T de manière récursive en commençant par le nœud racine r. Pour chaque nœud x de l'arbre, l'algorithme associe un degré d_x au polynôme q_x de tel sorte que d_x est inférieur de un à la valeur seuil k_x du nœud (i.e., $d_x = k_x - 1$). Pour le nœud racine, l'algorithme associe $q_r(0) = s$ et d_r autres points du polynôme q_r aléatoirement pour le définir complètement. Pour tous les autres nœuds x, l'algorithme associe $q_x(0) = q_{parent(x)}(index(x))$ et choisit aléatoirement d_x autres points pour définir complètement q_x. Nous

définissons $\lambda_x = q_x(0)$ pour chaque nœud feuille x de T. Ainsi, cet algorithme renvoie $\lambda = \{\lambda_x\}_{x \in leaf(T)}$, l'ensemble des parts secrètes de s pour chaque feuille de T ;

— **recoverCoefficients**$(T, \gamma) \rightarrow (\omega, S)$. Cet algorithme prend en entrée un arbre d'accès T, un ensemble d'attributs γ et calcule les coefficients de reconstruction de s si l'ensemble d'attributs γ satisfait l'arbre d'accès T (i.e., $T(\gamma) = 1$). Cet algorithme récupère d'abord un ensemble minimal d'attributs $S \subseteq \gamma$ satisfaisant l'arbre d'accès T. Pour chaque nœud x correspondant à un nœud feuille appartenant à S ou à un antécédent d'un nœud feuille appartenant à S à l'exception du nœud racine, l'algorithme calcule les coefficients de reconstruction $\omega_x = \omega_{parent(x)} \cdot \triangle_{i,V_x}(0)$, avec $i = index(x)$, $V_x = \{index(z)\}_{z \in child(parent(x))}$, et $\triangle_{i,V}(x)$ le coefficient de Lagrange défini de la manière suivante :

$$\triangle_{i,V}(x) = \prod_{j \in V, j \neq i} \frac{x - j}{i - j}$$

Pour le nœud racine r de T, l'algorithme associe $\omega_r = 1$. Ainsi, cet algorithme renvoie $\omega = \{\omega_x\}_{x \in S}$, l'ensemble des coefficients de reconstruction pour chaque attributs de l'ensemble minimal S, cet algorithme renvoie également S.

Ainsi, ces algorithmes permettent de garantir la propriété de reconstruction suivante, pour un secret $s \in \mathbb{Z}_p$, un arbre d'accès T, un ensemble d'attributs γ, un ensemble de parts secrètes λ obtenu en exécutant **computeSecretShares**$(s, T) \rightarrow \lambda$, un ensemble de coefficients de reconstruction ω et un ensemble minimal d'attributs S obtenus en exécutant **recoverCoefficients**$(T, \gamma) \rightarrow (\omega, S)$:

$$\sum_{x \in S} \lambda_x \cdot \omega_x = s$$

6.1.6 Fonctions de hachage

Soit $\mathbb{G}_1$ et $\mathbb{G}_2$, deux groupes cycliques d'ordre premier p, nous définissons une fonction de hachage d'une chaîne de caractères vers $\mathbb{G}_1$: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, nous définissons également une fonction de hachage d'une chaîne de caractères vers $\mathbb{G}_2$: $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_2$.

6.2 Pré-calculs

Les optimisations des schémas KP-ABE reposent sur des pré-calculs, lors du déchiffrement, qui peuvent être réutilisés dans certaines conditions lors des déchiffrements suivants.

6.2.1 Fixed-Argument Pairing

Le **Fixed-Argument Pairing (FAP)** [Scott 2011] permet de réduire le temps d'exécution des couplages. Ce pré-calcul consiste à réaliser les opérations de l'algorithme de couplage dépendant du paramètre de droite, et à stocker les valeurs

pré-calculées résultantes. Puis, lorsque le paramètre de gauche est connu, le couplage peut être calculé en réalisant les opérations de l'algorithme de couplage dépendant du paramètre de gauche et des valeurs pré-calculées. Ainsi, ces deux opérations permettent d'enlever le temps d'exécution des opérations de l'algorithme de couplage dépendant du paramètre de droite lorsqu'un même paramètre est réutilisé lors de plusieurs couplages.

Nous introduisons des fonctions permettant de représenter le FAP. Soit $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$, trois groupes cycliques d'ordre premier p, un couplage bilinéaire $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, et $\mathbb{C}$ un ensemble de valeurs pré-calculées, nous définissons les fonctions suivantes :

— le pré-calcul FAP, consistant à pré-calculer les opérations internes du couplage e dépendant uniquement du paramètre de droite, est dénoté par $e_R : \mathbb{G}_2 \to \mathbb{C}$;

— le calcul FAP, consistant à calculer le couplage en réalisant les opérations dépendant du paramètre de gauche et en utilisant les valeurs pré-calculées, est dénoté par $e_L : \mathbb{G}_1 \times \mathbb{C} \to \mathbb{G}_T$.

Ces deux fonctions respectent la propriété suivante, $\forall a \in \mathbb{G}_1, b \in \mathbb{G}_2, Q = e_R(b)$:

$$e_L(a, Q) = e(a, b)$$

Le gain apporté en temps de calcul et en taille des valeurs pré-calculées dépendent de l'algorithme réalisant le couplage, plus d'informations sont données en Section 6.4.

Ce pré-calcul permet donc de réduire le temps de calcul du couplage lorsque les valeurs pré-calculées sont réutilisées lors de plusieurs couplages. Par exemple, considérons un algorithme de déchiffrement dans lequel un couplage est réalisé à partir d'un élément de la clé de déchiffrement en paramètre de droite et d'un élément du chiffré en paramètre de gauche. Considérons également que l'on applique le pré-calcul FAP sur l'élément de la clé, si les valeurs pré-calculées sont réutilisées lors de plusieurs déchiffrements, alors le temps de calcul des couplages du déchiffrement diminue en ne nécessitant plus de réaliser le calcul de la partie droite du couplage. Cependant, selon la construction du schéma cryptographique, il est possible que l'algorithme de déchiffrement procède de manière inverse en réalisant un couplage dans lequel le paramètre de droite est un élément du chiffré et le paramètre de gauche est un élément de la clé de déchiffrement. Or, le chiffré étant généralement connu après la clé de déchiffrement et possédant généralement des valeurs obtenues de manière aléatoire à chaque chiffrement empêchant la réutilisation de valeurs pré-calculées, le pré-calcul FAP n'est souvent pas applicable. Néanmoins, il est possible d'inverser le groupe des éléments de la clé de déchiffrement et du chiffré pour pouvoir bénéficier du pré-calcul FAP sur les éléments de la clé de déchiffrement.

6.2.2 Pré-calcul Stream

Lors du déchiffrement des schémas KP-ABE de la section suivante, des exponentiations sont réalisées dans lesquelles les bases sont des éléments de la clé de déchiffrement et les exposants sont obtenus à partir de la clé de déchiffrement et

du chiffré. Dans cette situation, les valeurs résultantes ne sont généralement pas considérées pré-calculables du fait qu'elles dépendent du chiffré possédant des éléments générés aléatoirement dans un espace très grand. Cependant, dans les schémas KP-ABE, de nombreuses exponentiations sont réalisées en utilisant les coefficients de reconstruction renvoyés par la fonction **recoverCoefficients**. Ces coefficients dépendent de l'arbre d'accès de la clé de déchiffrement et de l'ensemble d'attributs du chiffré, ainsi lorsqu'un même arbre d'accès et un même ensemble d'attributs sont donnés en entrée de cette fonction, les coefficients renvoyés sont identiques. Or, une clé de déchiffrement pouvant déchiffrer un ensemble fini d'ensemble d'attributs, cet ensemble peut être plus ou moins grand selon la complexité de l'arbre d'accès, les valeurs résultantes des exponentiations peuvent être stockées et réutilisées. Le **pré-calcul Stream** consiste, lors du déchiffrement, à stocker les valeurs résultantes des exponentiations d'éléments de la clé de déchiffrement par des coefficients de reconstruction.

Le gain apporté par ce pré-calcul permet donc d'enlever le temps de calcul des exponentiations lorsqu'un même ensemble minimal d'attributs S est récupéré lors du déchiffrement après l'exécution de la fonction **recoverCoefficients**. En fonction du groupe de l'élément de la clé de déchiffrement, il est nécessaire de stocker un élément de $\mathbb{G}_1$ ou de $\mathbb{G}_2$ par valeur résultante de l'exponentiation.

6.2.3 Stream Fixed-Argument Pairing

Considérons un algorithme de déchiffrement dans lequel des exponentiations sont réalisées avec comme base un élément de la clé de déchiffrement appartenant à $\mathbb{G}_2$ et comme exposant un coefficient de reconstruction renvoyé par l'algorithme **recoverCoefficients**. Dans cette situation le pré-calcul Stream est applicable. Considérons maintenant que le résultat de cette exponentiation est utilisé dans un couplage, le FAP est alors applicable au couplage. Nous désignons cette situation comme étant le **Stream Fixed-Argument Pairing (Stream FAP)**.

Le gain apporté par ce pré-calcul combine le gain apporté par le pré-calcul Stream en enlevant le temps de calcul des exponentiations, et le FAP en réduisant le temps de calcul des couplages. En ce qui concerne le coût de stockage, cela dépend de la construction du schéma, un certain nombre de valeurs pré-calculées par le FAP est à stocker.

6.3 Optimisations des schémas KP-ABE

Les pré-calculs précédemment décrits sont appliqués au schéma KP-ABE Large Universe d'OpenABE ainsi qu'au schéma KP-ABE Small Universe proposé par Goyal-Pandey-Sahai-Waters [Goyal 2006]. Dans ce qui suit, une optimisation désigne l'application d'un pré-calcul à un algorithme d'un schéma KP-ABE. Par ailleurs, des changements de base lors des exponentiations ou un changement de groupe des éléments peuvent également être réalisés afin de réduire le temps de calcul ou de bénéficier de pré-calculs. Une forme désigne l'application de l'une de ces modifications

à un schéma. Chaque forme permet d'obtenir le même résultat au déchiffrement que la forme de base, en effet les éléments sur lesquels portent ces modifications sont utilisés dans des couplages à partir de deux éléments et d'une exponentiation sur l'un des deux éléments ou sur le résultat du couplage, la propriété de bilinéarité permet de garantir que le résultat de ces couplages est le même quelque soit la base de l'exponentiation parmi les deux paramètres du couplage ou le résultat du couplage. La dénomination *Forme.Optimisation* est utilisée pour faire référence aux formes et optimisations, dans laquelle *Forme* désigne une ou plusieurs modifications des opérations réalisées sur un schéma, et *Optimisation* désigne une ou plusieurs optimisations réalisées sur la structure du schéma. Lorsque plusieurs modifications ou optimisations sont réalisées, elles sont séparées via le symbole _. L'accent est mis sur l'optimisation de l'algorithme du déchiffrement car il est principalement exécuté en ligne (lors d'un trajet d'un véhicule) par les parties prenantes. L'algorithme de setup étant exécuté une seule fois à l'initialisation du système, et l'algorithme de génération des clés de déchiffrement étant exécuté hors ligne (lorsque le véhicule ne réalise pas de trajet) par l'autorité de confiance, nous considérons que ces algorithmes ne sont pas critiques et n'ont donc pas besoin d'être optimisés. Pour chaque forme et pour chaque optimisation, nous présentons les opérations réalisées, le coût et le gain théorique en temps de calcul, le coût théorique en espace de stockage, et dans quelles circonstances l'optimisation peut être appliquée dans notre contexte. Les symboles utilisés lors de l'évaluation des coûts et des gains théoriques sont représentés en Table 6.1.

TABLE 6.1 – Symboles utilisés lors de l'évaluation des coûts et des gains théoriques

Description	Symbole
Ensemble d'attributs utilisé lors du chiffrement	γ
Ensemble minimal d'attributs de γ satisfaisant l'arbre d'accès T lors du déchiffrement	S
Cardinalité de l'ensemble γ, S	$\|\gamma\|, \|S\|$
Taille d'un élément de $\mathbb{G}_1, \mathbb{G}_2$, des valeurs pré-calculées pour un couplage	$\mathcal{L}_{\mathbb{G}_1}, \mathcal{L}_{\mathbb{G}_2}, \mathcal{L}_{\mathbb{C}}$
Temps d'exécution d'une exponentiation dans $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	$\mathcal{E}_{\mathbb{G}_1}, \mathcal{E}_{\mathbb{G}_2}, \mathcal{E}_{\mathbb{G}_T}$
Temps d'exécution d'une multiplication dans $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	$\mathcal{M}_{\mathbb{G}_1}, \mathcal{M}_{\mathbb{G}_2}, \mathcal{M}_{\mathbb{G}_T}$
Temps d'exécution d'une division dans $\mathbb{G}_T$	$\mathcal{D}_{\mathbb{G}_T}$
Temps d'exécution d'une fonction de hachage d'une chaîne de caractères vers $\mathbb{G}_1, \mathbb{G}_2$	$\mathcal{H}_{\mathbb{G}_1}, \mathcal{H}_{\mathbb{G}_2}$
Temps d'exécution d'un couplage, du pré-calcul FAP, du calcul FAP	$\mathcal{C}, \mathcal{C}_R, \mathcal{C}_L$

LU.Base.Setup(τ) $\rightarrow$ (PK, MSK). Cet algorithme prend en entrée un paramètre de sécurité τ. L'algorithme exécute **BSetup**(1^τ) $\rightarrow (p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ et choisit aléatoirement un exposant $y \in_r \mathbb{Z}_p$. L'algorithme génère une clé publique PK et une clé maîtresse secrète MSK comme suit :

$$\text{PK} = (g_1, g_2, Y = e(g_1, g_2)^y) \qquad \text{MSK} = y$$

LU.Base.Keygen(PK, MSK, T) $\rightarrow$ SK. Cet algorithme prend en entrée une clé publique PK, une clé maîtresse secrète MSK et un arbre d'accès T. L'algorithme génère une clé de déchiffrement permettant à un utilisateur de déchiffrer un message chiffré avec un ensemble d'attributs γ, si et seulement si $T(\gamma) = 1$. L'algorithme calcule de manière aléatoire les parts de y en fonction de l'arbre d'accès T (i.e., **computeSecretShares**$(y, T) \rightarrow \lambda$) et, pour chaque feuille i de T, choisit aléatoirement un coefficient $r_i \in_r \mathbb{Z}_p$. L'algorithme génère une clé de déchiffrement SK comme suit :

$$\text{SK} = (T,\ (D_i = g_1^{\lambda_i} \cdot H_1(i)^{r_i},\ d_i = g_2^{r_i})_{i \in leaf(T)})$$

LU.Base.Encrypt(PK, γ) $\rightarrow$ (L, CT). Cet algorithme prend en entrée une clé publique PK et un ensemble d'attributs γ. L'algorithme choisit aléatoirement un exposant $s \in_r \mathbb{Z}_p$. L'algorithme génère un verrou L et un chiffré CT comme suit :

$$\text{L} = Y^s = e(g_1, g_2)^{ys} \qquad \text{CT} = (\gamma,\ E'' = g_2^s,\ (E_i = H_1(i)^s)_{i \in \gamma})$$

LU.Base.Decrypt(CT, SK) $\rightarrow$ L ou $\perp$. Cet algorithme prend en entrée un chiffré CT et une clé de déchiffrement SK. L'algorithme détermine d'abord si l'arbre d'accès T de SK est satisfait par l'ensemble d'attributs γ de CT (i.e., si $T(\gamma) = 1$). Si $T(\gamma) \neq 1$, l'algorithme renvoie $\perp$. Sinon, l'algorithme récupère les coefficients de Lagrange ω pour l'ensemble minimal d'attributs S nécessaire pour satisfaire T (i.e., **recoverCoefficients**$(T, \gamma) \rightarrow (\omega, S)$). Par conséquent, pour chaque attribut $i \in S$, les coefficients correspondants de ω (i.e., ω_i), les composants correspondants de CT (i.e., E'' et E_i) et les composants correspondants de SK (i.e., D_i et d_i), l'algorithme calcule :

$$\frac{e(\prod_{i \in S} D_i^{\omega_i}, E'')}{\prod_{i \in S} e(E_i^{\omega_i}, d_i)} = \frac{e(\prod_{i \in S} (g_1^{\lambda_i} \cdot H_1(i)^{r_i})^{\omega_i}, g_2^s)}{\prod_{i \in S} e((H_1(i)^s)^{\omega_i}, g_2^{r_i})} =$$

$$\frac{e(\prod_{i \in S} g_1^{\lambda_i \omega_i} \cdot H_1(i)^{r_i \omega_i}, g_2^s)}{\prod_{i \in S} e(H_1(i)^{s\omega_i}, g_2^{r_i})} = \frac{e(\prod_{i \in S} g_1^{\lambda_i \omega_i}, g_2^s) \cdot e(\prod_{i \in S} H_1(i)^{r_i \omega_i}, g_2^s)}{\prod_{i \in S} e(H_1(i)^{s\omega_i}, g_2^{r_i})} =$$

$$\frac{\prod_{i \in S} e(g_1, g_2)^{s\lambda_i\omega_i} \cdot \prod_{i \in S} e(H_1(i), g_2)^{s\omega_i r_i}}{\prod_{i \in S} e(H_1(i), g_2)^{s\omega_i r_i}} = \prod_{i \in S} e(g_1, g_2)^{s\lambda_i\omega_i} =$$

$$e(g_1, g_2)^{s \sum_{i \in S} \lambda_i \omega_i} = e(g_1, g_2)^{sy} = \text{L}$$

FIGURE 6.1 – Schéma KP-ABE Large Universe (issu de la documentation d'OpenABE)

6.3.1 Optimisation du schéma Large Universe

Le schéma KP-ABE Large Universe de la bibliothèque OpenABE [1] est représenté en Figure 6.1, ce schéma est une variante du schéma Goyal-Pandey-Sahai-Waters Large Universe [Goyal 2006, Pirretti 2006]. Ce schéma implémente un *mécanisme d'encapsulation de clé* (en anglais *Key Encapsulation Mechanism, KEM*), ainsi l'algorithme de chiffrement KP-ABE ne prend pas de message en entrée mais renvoie un verrou (L) qui est utilisé par un autre algorithme de chiffrement pour générer une clé symétrique. Cette clé symétrique est ensuite utilisée pour chiffrer un message. Ainsi, nous nous intéressons dans ce qui suit uniquement à l'algorithme de chiffrement KP-ABE renvoyant un chiffré (CT) permettant d'obtenir le verrou (L).

6.3.1.1 Forme LU et optimisations di et prodDi

Forme LU La coût de l'algorithme de chiffrement est le suivant $|\gamma| \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{H}_{\mathbb{G}_1}) + \mathcal{E}_{\mathbb{G}_T} + \mathcal{E}_{\mathbb{G}_2}$; le coût de l'algorithme de déchiffrement est le suivant $|S| \cdot (2\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$.

Optimisation di : FAP Le FAP étant applicable au paramètre de droite du couplage, il peut être ici appliqué au couplage au dénominateur du déchiffrement (il ne peut pas être appliqué au couplage du numérateur car le paramètre de droite est un élément du chiffré généré aléatoirement à chaque chiffrement). Ainsi, lorsqu'un élément d_i de la clé de déchiffrement est utilisé pour la première fois lors de l'exécution de l'algorithme de déchiffrement, le FAP est appliqué, puis les valeurs pré-calculées sont réutilisées lors des déchiffrement successifs :

LU.di.Decrypt(CT, SK).
$\forall i \in S$, si d'_i n'existe pas, $d'_i = e_R(d_i)$.
Renvoie : $\dfrac{e(\prod_{i \in S} D_i^{\omega_i}, E'')}{\prod_{i \in S} e_L(E_i^{\omega_i}, d'_i)}$

Cette optimisation réduit le temps du déchiffrement en réduisant le temps de calcul des couplages au dénominateur, mais elle nécessite de stocker les valeurs pré-calculées pour chaque élément d_i apparaissant pour la première fois dans S :

Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
$\lvert S\rvert \cdot (2\mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot \mathcal{C}_R$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$

Dans notre contexte, cette optimisation est donc applicable sans contrainte particulière à la totalité des éléments d_i de la clé de déchiffrement de chaque entité du système.

Optimisation prodDi : pré-calcul Stream Au déchiffrement, pour un ensemble minimal d'attributs S, les exponentiations des ω_i sur les D_i et le produit des $D_i^{\omega_i}$

1. Bibliothèque disponible sur le lien suivant : `https://github.com/zeutro/openabe`

au numérateur peuvent être calculés lors du premier déchiffrement et réutilisés lors des déchiffrements successifs lorsqu'un même ensemble d'attributs S est utilisé :

LU.prodDi.Decrypt(CT, SK).
Si D'_S n'existe pas, $D'_S = \prod_{i \in S} D_i^{\omega_i}$.
Renvoie : $\dfrac{e(D'_S, E'')}{\prod_{i \in S} e(E_i^{\omega_i}, d_i)}$

Cette optimisation réduit le temps du déchiffrement en enlevant le temps de calcul des exponentiations ainsi que du produit des $D_i^{\omega_i}$, mais elle nécessite de stocker un élément de $\mathbb{G}_1$ par ensemble d'attributs S :

Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{M}_{\mathbb{G}_1})$	$\mathcal{L}_{\mathbb{G}_1}$

L'application de cette optimisation dépend donc de la clé de déchiffrement et de l'ensemble d'attributs minimal S de γ. Cependant il est nécessaire que l'ensemble d'attributs S soit identique entre les différents déchiffrements pour réutiliser la valeur calculée D'_S. Lorsqu'un attribut de S est différent et si cette optimisation n'a jamais été calculée pour le nouvel ensemble S, les exponentiations des ω_i sur les D_i doivent être calculées et le produit des $D_i^{\omega_i}$ doit être calculé puis stocké. Dans notre contexte, cette optimisation est applicable à la clé de déchiffrement du véhicule et du cloud, leur clé contenant un unique attribut correspondant à leur identité, elle ne peut donc déchiffrer qu'un unique ensemble S contenant un attribut correspondant à leur identité, entraînant ainsi uniquement un élément de $\mathbb{G}_1$ à stocker. Cette optimisation est aussi applicable à la clé de déchiffrement des parties prenantes bien que l'ensemble S peut varier entre plusieurs déchiffrements. Pour une partie prenante donnée, la totalité des ensembles d'attributs que sa clé de déchiffrement permet de déchiffrer est finie et d'une taille peu conséquente. Prenons par exemple le cas d'utilisation donné en fin du Chapitre 4, la clé de déchiffrement de `infraB1` est la suivante :

```
infraB1(delegue):
    (pp_role:infra_route ET type:route_endommagee)
    OU (pp_id:infra ET pp_attr:regionB)
```

Cette clé de déchiffrement permet de déchiffrer des chiffrés contenant au minimum les ensembles d'attributs suivants :

```
1: pp_role:infra_route ET type:route_endommagee
2: pp_id:infra ET pp_attr:regionB
```

Ainsi, cette clé de déchiffrement permet de déchiffrer uniquement deux ensembles d'attributs, et donc uniquement deux valeurs dans $\mathbb{G}_1$ sont à stocker.

Combinaison des optimisations di et prodDi Ces deux optimisations peuvent être combinées afin de bénéficier du gain apporté par chaque optimisation.

6.3.1.2 Forme LU_di et optimisations di2 et di3

Forme LU_di : exponentiations sur les di Cette forme consiste à modifier l'algorithme de déchiffrement en réalisant les exponentiations au dénominateur des ω_i sur les d_i au lieu des E_i :

LU.Base.Decrypt(CT, SK). Renvoie : $\dfrac{e(\prod_{i\in S} D_i^{\omega_i}, E'')}{\prod_{i\in S} e(E_i^{\omega_i}, d_i)}$	$\Longrightarrow$	**LU_di.Base.Decrypt**(CT, SK). Renvoie : $\dfrac{e(\prod_{i\in S} D_i^{\omega_i}, E'')}{\prod_{i\in S} e(E_i, d_i^{\omega_i})}$

Cette forme entraîne un temps de déchiffrement plus long que la forme LU sans optimisation car les exponentiations sont réalisées sur des éléments de $\mathbb{G}_2$. Cette forme ne peut plus bénéficier de l'optimisation di. Cependant, elle peut bénéficier d'optimisations qui étendent l'optimisation di en appliquant le pré-calcul Stream au dénominateur (optimisation di2, détaillée ci-dessous) ou en appliquant le Stream FAP au dénominateur (optimisation di3, détaillée ci-dessous). Le déchiffrement modifiant uniquement le dénominateur, elle peut donc également bénéficier de l'optimisation prodDi. Ainsi, cette forme peut bénéficier des combinaisons des optimisations prodDi et di2 ou di3.

Optimisation di2 : pré-calcul Stream Au déchiffrement, pour un ensemble minimal d'attributs S, les exponentiations des ω_i sur les d_i peuvent être calculées lors du premier déchiffrement et réutilisées lors des déchiffrements successifs lorsqu'un même ensemble d'attributs S est utilisé :

LU_di.di2.Decrypt(CT, SK).
$\forall i \in S$, si $d'_{S,i}$ n'existe pas, $d'_{S,i} = d_i^{\omega_i}$.
Renvoie : $\dfrac{e(\prod_{i\in S} D_i^{\omega_i}, E'')}{\prod_{i\in S} e(E_i, d'_{S,i})}$

Cette optimisation réduit le temps du déchiffrement en enlevant le temps de calcul des exponentiations sur les d_i, mais elle nécessite de stocker un élément de $\mathbb{G}_2$ par attributs et par ensemble d'attributs S :

Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot \mathcal{E}_{\mathbb{G}_2}$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{G}_2}$

L'application de cette optimisation dans notre contexte est similaire à l'optimisation prodDi, cependant elle est plus coûteuse en stockage car elle nécessite de stocker une valeur par élément d_i de la clé de déchiffrement pour chaque ensemble minimal d'attributs S. Dans l'exemple du cas d'utilisation donné en fin du Chapitre

4, pour la partie prenante `infraB1`, sa clé de déchiffrement permet de déchiffrer deux ensembles minimaux d'attributs S, chaque ensemble contient deux attributs, ainsi quatre valeurs au total sont à stocker pour appliquer cette optimisation.

Optimisation di3 : Stream FAP Cette optimisation permet d'étendre l'optimisation di2 en appliquant Stream FAP sur les exponentiations des ω_i sur les d_i :

LU_di.di3.Decrypt(CT, SK).
$\forall i \in S$, si $d'_{S,i}$ n'existe pas, $d'_{S,i} = e_R(d_i^{\omega_i})$.
Renvoie : $\dfrac{e(\prod_{i\in S} D_i^{\omega_i}, E'')}{\prod_{i\in S} e_L(E_i, d'_{S,i})}$

Cette optimisation permet de réduire le temps du déchiffrement en réduisant le temps de calcul des couplages au dénominateur et en enlevant le temps de calcul des exponentiations sur les d_i :

Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{C}_R)$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$

L'application de cette optimisation a les mêmes contraintes que l'optimisation di2, en revanche cette optimisation est plus coûteuse en stockage que l'optimisation di2 car la taille d'un pré-calcul d'un couplage est plus grande que la taille d'un élément de $\mathbb{G}_2$.

6.3.1.3 Forme LU_Inv

La forme LU_Inv consiste à inverser les éléments dans $\mathbb{G}_1$ et $\mathbb{G}_2$.

Impact sur le chiffrement L'inversion des éléments lors du chiffrement entraîne les opérations suivantes :

LU.Base.Encrypt(PK, γ).
Renvoie : CT = (... $E'' = g_2^s$, $(E_i = H_1(i)^s)_{i\in\gamma}$)

$\Downarrow$

LU_Inv.Base.Encrypt(PK, γ).
Renvoie : CT = (... $E'' = g_1^s$, $(E_i = H_2(i)^s)_{i\in\gamma}$)

Cependant, cette forme ralentit le chiffrement car les exponentiations sont réalisées sur des éléments de $\mathbb{G}_2$.

Impact sur le déchiffrement L'inversion des éléments lors du déchiffrement entraîne les opérations suivantes :

LU.Base.Decrypt(CT, SK). Renvoie : $\frac{e(\prod_{i\in S} D_i^{\omega_i}, E'')}{\prod_{i\in S} e(E_i^{\omega_i}, d_i)}$	$\Longrightarrow$	**LU_Inv.Base.Decrypt**(CT, SK). Renvoie : $\frac{e(E'', \prod_{i\in S} D_i^{\omega_i})}{\prod_{i\in S} e(d_i^{\omega_i}, E_i)}$

Au dénominateur, les exponentiations peuvent toujours être réalisées sur les d_i ce qui n'affecte pas le coût du déchiffrement. Cependant, au numérateur, les exponentiations ne peuvent être réalisées que sur les D_i qui sont maintenant dans $\mathbb{G}_2$, cela ralentit donc le déchiffrement lorsqu'aucune optimisation n'est réalisée. En revanche, le Stream FAP peut être appliqué au numérateur, ce qui n'était pas possible pour les formes LU et LU_di. Cependant, bien que le dénominateur peut bénéficier de l'optimisation di2, il ne peut plus bénéficier du FAP et donc de l'optimisation di3.

Ainsi, ces optimisations ne permettent pas d'obtenir un gain plus important que les formes LU et LU_di optimisées car un seul couplage bénéficie du FAP dans la forme LU_Inv alors qu'en contrepartie tous les couplages au dénominateur bénéficient du FAP dans les formes LU et LU_di. Nous jugeons donc cette forme peu avantageuse, elle ne sera pas évaluée dans la suite.

6.3.2 Optimisation du schéma Small Universe

Le schéma KP-ABE Small Universe de Goyal-Pandey-Sahai-Waters [Goyal 2006] est représenté en Figure 6.2. L'algorithme de chiffrement du schéma original prenant un message en entrée, il a été adapté dans nos travaux pour implémenter un mécanisme d'encapsulation de clé, ainsi l'algorithme de chiffrement ne prend pas de message en entrée mais renvoie un verrou (L) qui est utilisé pour générer une clé AES.

6.3.2.1 Formes SU, SU_Ei, SU_Di et optimisation Di

Forme SU Le coût de l'algorithme de chiffrement est le suivant $|\gamma| \cdot \mathcal{E}_{\mathbb{G}_2} + \mathcal{E}_{\mathbb{G}_T}$; le coût de l'algorithme de déchiffrement est le suivant $|S| \cdot (\mathcal{E}_{\mathbb{G}_T} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$.

Forme SU_Ei : exponentiation sur les Ei Dans cette forme, au déchiffrement, les exponentiations des ω_i sont réalisées sur les E_i. Cette forme entraîne un temps de déchiffrement plus court que la forme SU car les exponentiations sont réalisées sur des éléments de $\mathbb{G}_2$.

Forme SU_Di : exponentiations sur les Di Dans cette forme, au déchiffrement, les exponentiations des ω_i sont réalisées sur les D_i. Cette forme entraîne un temps de déchiffrement plus court que la forme SU_Ei car les exponentiations sont réalisées sur des éléments de $\mathbb{G}_1$. Cette forme peut également bénéficier du pré-calcul Stream sur les exponentiations des D_i (optimisation Di).

SU.Base.Setup(τ, $U = \{1, 2, \ldots, n\}$) $\rightarrow$ (PK, MSK). Cet algorithme prend en entrée un paramètre de sécurité τ. L'algorithme exécute **BSetup**(1^τ) $\rightarrow$ $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ et choisit aléatoirement des exposants $y, v_1, \ldots, v_n \in_r \mathbb{Z}_p$. L'algorithme génère une clé publique PK et une clé maîtresse secrète MSK comme suit :

$$\text{PK} = (g_1, g_2, Y = e(g_1, g_2)^y, V_1 = g_2^{v_1}, \ldots, V_n = g_2^{v_n}) \qquad \text{MSK} = y, v_1, \ldots, v_n$$

SU.Base.Keygen(PK, MSK, T) $\rightarrow$ SK. Cet algorithme prend en entrée une clé publique PK, une clé maîtresse secrète MSK et un arbre d'accès T. L'algorithme génère une clé de déchiffrement permettant à un utilisateur de déchiffrer un message chiffré avec un ensemble d'attributs γ, si et seulement si $T(\gamma) = 1$. L'algorithme calcule de manière aléatoire les parts de y en fonction de l'arbre d'accès T (i.e., **computeSecretShares**$(y, T) \rightarrow \lambda$). L'algorithme génère une clé de déchiffrement SK comme suit :

$$\text{SK} = (T,\ (D_i = g_1^{\frac{\lambda_i}{v_i}})_{i \in leaf(T)})$$

SU.Base.Encrypt(PK, γ) $\rightarrow$ (L, CT). Cet algorithme prend en entrée une clé publique PK et un ensemble d'attributs γ. L'algorithme choisit aléatoirement un exposant $s \in_r \mathbb{Z}_p$. L'algorithme génère un verrou L et un chiffré CT comme suit :

$$\text{L} = Y^s = e(g_1, g_2)^{ys} \qquad \text{CT} = (\gamma,\ (E_i = V_i^s = g_2^{sv_i})_{i \in \gamma})$$

SU.Base.Decrypt(CT, SK) $\rightarrow$ L ou $\perp$. Cet algorithme prend en entrée un chiffré CT et une clé de déchiffrement SK. L'algorithme détermine d'abord si l'arbre d'accès T de SK est satisfait par l'ensemble d'attributs γ de CT (i.e., si $T(\gamma) = 1$). Si $T(\gamma) \neq 1$, l'algorithme renvoie $\perp$. Sinon, l'algorithme récupère les coefficients de Lagrange ω pour l'ensemble minimal d'attributs S nécessaire pour satisfaire T (i.e., **recoverCoefficients**$(T, \gamma) \rightarrow (\omega, S)$). Par conséquent, pour chaque attribut $i \in S$, les coefficients correspondants de ω (i.e., ω_i) et les composants correspondants de SK (i.e., D_i), l'algorithme calcule :

$$\prod_{i \in S} e(D_i, E_i)^{\omega_i} = \prod_{i \in S} e(g_1^{\frac{\lambda_i}{v_i}}, g_2^{sv_i})^{\omega_i} = \prod_{i \in S} e(g_1, g_2)^{s\lambda_i\omega_i} =$$

$$e(g_1, g_2)^{s\sum_{i \in S} \lambda_i\omega_i} = e(g_1, g_2)^{sy} = \text{L}$$

FIGURE 6.2 – Schéma KP-ABE Small Universe (adapté de [Goyal 2006])

Optimisation Di : pré-calcul Stream Au déchiffrement, pour un ensemble minimal d'attributs S, les exponentiations des ω_i sur les D_i peuvent être calculées lors du premier déchiffrement et réutilisées lors des déchiffrements successifs lorsqu'un même ensemble d'attributs S est utilisé :

SU_Di.Di.Decrypt(CT, SK).
$\forall i \in S$, si $D'_{S,i}$ n'existe pas, $D'_{S,i} = D_i^{\omega_i}$.
Renvoie : $\prod_{i \in S} e(D'_{S,i}, E_i)$

Cette optimisation réduit le temps du déchiffrement en enlevant le temps de calcul des exponentiations sur les D_i, mais elle nécessite de stocker un élément de $\mathbb{G}_1$ par attribut et par ensemble d'attributs S :

Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
$\lvert S\rvert \cdot (\mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot \mathcal{E}_{\mathbb{G}_1}$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{G}_1}$

Cette optimisation a les mêmes contraintes que l'optimisation di2, son coût en stockage est en revanche moins élevé car la taille d'un élément de $\mathbb{G}_1$ est plus petite que la taille d'un élément de $\mathbb{G}_2$.

6.3.2.2 Forme SU_Inv, optimisation Di2 et forme SU_Inv_Ei

Forme SU_Inv Cette forme consiste à inverser les éléments dans $\mathbb{G}_1$ et $\mathbb{G}_2$. L'inversion des éléments lors du chiffrement entraîne les opérations suivantes :

SU.Base.Encrypt(PK, γ).
Renvoie : CT = (... $(E_i = (g_2^{v_i})^s)_{i \in \gamma}$)

$\Downarrow$

SU_Inv.Base.Encrypt(PK, γ).
Renvoie : CT = (... $(E_i = (g_1^{v_i})^s)_{i \in \gamma}$)

Cette forme entraîne un chiffrement plus rapide que les formes SU, SU_Ei et SU_Di car les exponentiations sont réalisées sur des éléments de $\mathbb{G}_1$: $|\gamma| \cdot \mathcal{E}_{\mathbb{G}_1} + \mathcal{E}_{\mathbb{G}_T}$.

L'inversion des éléments lors du déchiffrement entraîne les opérations suivantes :

SU.Base.Decrypt(CT, SK).
Renvoie : $\prod_{i \in S} e(D_i, E_i)^{\omega_i}$

$\Longrightarrow$

SU_Inv.Base.Decrypt(CT, SK).
Renvoie : $\prod_{i \in S} e(E_i, D_i)^{\omega_i}$

Cette forme a le même coût au déchiffrement que la forme SU, cependant cette forme peut bénéficier du FAP (optimisation Di2).

Optimisation Di2 : FAP Au déchiffrement, le FAP est applicable sur les éléments D_i de la clé de déchiffrement SK. Lorsqu'un élément D_i est utilisé pour la première fois lors de l'exécution de l'algorithme de déchiffrement, le FAP est appliqué, puis les valeurs pré-calculées sont réutilisées lors des déchiffrements successifs :

SU_Inv.Di2.Decrypt(CT, SK).
$\forall i \in S$, si D'_i n'existe pas, $D'_i = e_R(D_i)$.
Renvoie : $\prod_{i \in S} e_L(E_i, D'_i)^{\omega_i}$

Cette optimisation réduit le temps du déchiffrement en réduisant le temps de calcul des couplages, mais elle nécessite de stocker les valeurs pré-calculées :

Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_T} + \mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot \mathcal{C}_R$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$

Cette optimisation a les mêmes contraintes que l'optimisation di.

Forme SU_Inv_Ei Cette forme consiste à réaliser les exponentiations des ω_i sur les E_i. Elle a le même coût au déchiffrement que la forme SU_Di, cependant elle peut également bénéficier du FAP (optimisation Di2).

6.3.2.3 Forme SU_Inv_Di et optimisation Di3

Forme SU_Inv_Di Dans cette forme, les exponentiations sont réalisées sur les D_i. Cette forme a le même coût au déchiffrement que la forme SU_Ei, cependant elle peut bénéficier de l'optimisation Di et de l'application du Stream FAP (optimisation Di3).

Optimisation Di3 : Stream FAP Cette optimisation permet d'étendre l'optimisation Di2 en appliquant le FAP :

SU_Inv_Di.Di3.Decrypt(CT, SK).
$\forall i \in S$, si $D'_{S,i}$ n'existe pas, $D'_{S,i} = e_R(D_i^{\omega_i})$.
Renvoie : $\prod_{i \in S} e_L(E_i, D'_{S,i})$

Cette optimisation réduit le temps du déchiffrement en réduisant le temps de calcul des couplages et en enlevant le temps de calcul des exponentiations sur les D_i :

Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
$\lvert S\rvert \cdot (\mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{C}_{\mathcal{R}})$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$

Cette optimisation a les mêmes contraintes que l'optimisation di3.

TABLE 6.2 – Récapitulatif des formes et optimisations possibles ainsi que des coûts et gains des schémas KP-ABE Large Universe et KP-ABE Small Universe pour le déchiffrement

Forme.Optimisation	Coût du déchiffrement	Gain par déchiffrement	Taille des pré-calculs
LU.Base	$\lvert S\rvert \cdot (2\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	×	×
LU.prodDi	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{M}_{\mathbb{G}_1})$	$\mathcal{L}_{\mathbb{G}_1}$
LU.di	$\lvert S\rvert \cdot (2\mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot \mathcal{C}_R$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$
LU.di_prodDi	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_R + \mathcal{M}_{\mathbb{G}_1})$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}} + \mathcal{L}_{\mathbb{G}_1}$
LU_di.Base	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	×	×
LU_di.prodDi	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{M}_{\mathbb{G}_1})$	$\mathcal{L}_{\mathbb{G}_1}$
LU_di.di2	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot \mathcal{E}_{\mathbb{G}_2}$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{G}_2}$
LU_di.di3	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T} + \mathcal{M}_{\mathbb{G}_1}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{C}_R)$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$
LU_di.prodDi_di2	$\lvert S\rvert \cdot (\mathcal{C} + \mathcal{M}_{\mathbb{G}_T}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{E}_{\mathbb{G}_1} + \mathcal{M}_{\mathbb{G}_1})$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{G}_2} + \mathcal{L}_{\mathbb{G}_1}$
LU_di.prodDi_di3	$\lvert S\rvert \cdot (\mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T}) + \mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_R + \mathcal{M}_{\mathbb{G}_1})$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}} + \mathcal{L}_{\mathbb{G}_1}$
SU.Base	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_T} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	×	×
SU_Ei.Base	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	×	×
SU_Di.Base	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	×	×
SU_Di.Di	$\lvert S\rvert \cdot (\mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot \mathcal{E}_{\mathbb{G}_1}$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{G}_1}$
SU_Inv.Base	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_T} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	×	×
SU_Inv.Di2	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_T} + \mathcal{C}_{\mathcal{L}} + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot \mathcal{C}_R$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$
SU_Inv_Ei.Base	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	×	×
SU_Inv_Ei.Di2	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_1} + \mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot \mathcal{C}_R$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$
SU_Inv_Di.Base	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	×	×
SU_Inv_Di.Di	$\lvert S\rvert \cdot (\mathcal{C} + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot \mathcal{E}_{\mathbb{G}_2}$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{G}_2}$
SU_Inv_Di.Di3	$\lvert S\rvert \cdot (\mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T})$	$\lvert S\rvert \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{C}_R)$	$\lvert S\rvert \cdot \mathcal{L}_{\mathbb{C}}$

× signifie une absence de pré-calcul

6.3.3 Conclusion

La liste des formes et optimisations possibles pour les schémas KP-ABE Large Universe et KP-ABE Small Universe est représentée en Table 6.2.

À partir de cette table, nous pouvons observer que pour la forme LU du schéma Large Universe, le temps de calcul est réduit par les optimisations di et prodDi, ces optimisations ont un impact significatif en enlevant le temps de calcul des exponentiations et des multiplications des D_i, ainsi qu'en réduisant le temps de calcul des couplages impliquant les d_i. Pour la forme LU_di, l'optimisation di ne peut plus être appliquée, cependant l'optimisation di2 est applicable, cette optimisation permet de réduire le temps de calcul du déchiffrement en enlevant les exponentiations sur les d_i. L'optimisation di3 de la forme LU_di permet de joindre le gain apporté par les optimisation di et di2 en enlevant les exponentiations sur les d_i et en réduisant le temps de calcul des couplages impliquant les d_i.

En ce qui concerne le schéma Small Universe, les changements de forme et l'optimisation Di permettent de réduire le temps de calcul en enlevant les exponentiations à réaliser. La forme SU_Inv_Di et son optimisation Di3 permettent d'étendre l'optimisation Di en réduisant le temps de calcul des couplages. Ainsi, ces formes et optimisations ont un impact significatif sur le temps de calcul du déchiffrement en réduisant le temps de calcul des exponentiations, en enlevant le temps de calcul des exponentiations, ou en réduisant le temps de calcul des couplages.

Le temps de calcul entre la forme LU_di.prodDi_di3 et la forme SU_Inv_Di.Di3 diffère d'un couplage et d'une division dans $\mathbb{G}_T$. Ainsi, pour un faible nombre d'attributs, la forme SU_Inv_Di.Di3 offre un gain plus important que la forme LU_di.prodDi_di3, puis ce gain devrait diminuer pour tendre vers une même valeur pour un grand nombre d'attributs. L'évaluation expérimentale permettra de déterminer à partir de combien d'attributs ce gain devient négligeable. En terme de stockage, SU_Inv_Di.Di3 entraîne un peu moins de stockage que LU_di.prodDi_di3 en ne nécessitant pas le stockage d'un élément de $\mathbb{G}_1$ par ensemble d'attributs S, cependant ce gain est négligeable.

En somme, l'optimisation di3 de la forme LU_di permet de réduire de manière significative le temps de calcul du déchiffrement du schéma Large Universe. L'optimisation Di3 de la forme SU_Inv_Di permet de réduire de manière significative le temps de calcul du déchiffrement du schéma Small Universe. Pour un faible nombre d'attributs, la forme SU_Inv_Di.Di3 est plus rapide que la forme LU_di.prodDi_di3. Puis à partir d'un certain nombre d'attributs qui sera déterminé lors de l'évaluation expérimentale, ce gain deviendra négligeable et le temps de calcul devrait être quasi équivalent.

6.4 Méthode d'évaluation et résultats

L'implémentation du FAP a été réalisée dans la bibliothèque RELIC [2], et l'implémentation du schéma Small Universe et des optimisations a été réalisée sur

2. Code disponible sur le lien suivant : `https://gitlab.laas.fr/radelin/relic_scott`

l'algorithme KP-ABE de la bibliothèque OpenABE [3], ces deux bibliothèques ont été présentées en Section 5.1 du chapitre précédent. L'évaluation des performances a été réalisée sur un ordinateur de bureau équipé d'un processeur AMD Ryzen 7 4800H 64 bits de 2.9 GHz et d'une mémoire vive de 16 Go.

6.4.1 Évaluation du FAP

La bibliothèque RELIC fournit deux fonctions pour calculer des couplages : la première réalise le couplage de deux éléments $e(a, b)$; la seconde réalise les couplages d'une liste d'éléments et le produit des couplages résultants $\prod_{i \in I} e(a_i, b_i)$, dans la suite nous appelons cette seconde fonction le **multi-couplage**. Le multi-couplage a la particularité d'être une fonction permettant d'obtenir le même résultat que la réalisation des couplages et du produit des couplages tout en ayant un temps de calcul réduit. La bibliothèque RELIC fournit également plusieurs algorithmes permettant de calculer un couplage et un multi-couplage, la bibliothèque OpenABE repose sur les fonctions de couplage et de multi-couplage de l'algorithme Optimal Ate Pairing. Ainsi, nous avons implémenté le FAP sur le couplage et le multi-couplage de l'algorithme Optimal Ate Pairing dans la bibliothèque RELIC.

Pour évaluer les performances de notre implémentation, nous avons mesuré le temps d'exécution en microseconde de la fonction du couplage, du pré-calcul FAP et du calcul FAP en faisant varier le nombre d'attributs de 1 à 1024 par puissance de deux, et pour chaque nombre d'attributs nous avons réalisé 1024 itérations.

Afin d'avoir la mesure la plus précise du temps d'exécution, les mesures ont été effectuées en utilisant le compteur de performance du processeur au moyen de l'instruction `rdtscp` représentée en Listing 6.1, puis le temps d'exécution a été calculé en microseconde en utilisant la fréquence du processeur. Le processeur a été réglé au préalable à sa fréquence maximale.

```
1 __asm__ volatile ("rdtscp\n" : "=&a" (t_low), "=&d" (t_high)
2                              :: "%rcx");
```

Listing 6.1 – Instruction assembleur permettant de récupérer la valeur du compteur de performance du processeur (x86)

La taille des pré-calculs FAP a été évaluée en analysant la définition des types définis dans RELIC, et en utilisant l'opérateur `sizeof`. De cette analyse, nous pouvons en tirer que la taille d'un pré-calcul FAP (un élément de $\mathbb{C}$) est de 17540 octets.

L'accélération du calcul FAP par rapport au multi-couplage en fonction du nombre de couplages est représentée en Figure 6.3. La moyenne de l'accélération, des temps d'exécution, et de la taille des données pré-calculées en fonction du nombre de couplages sont données en Table 6.3. De cette figure et de cette table, nous pouvons observer que l'accélération croit significativement en fonction du nombre

3. Code disponible sur le lien suivant : `https://gitlab.laas.fr/radelin/openabe_opti`

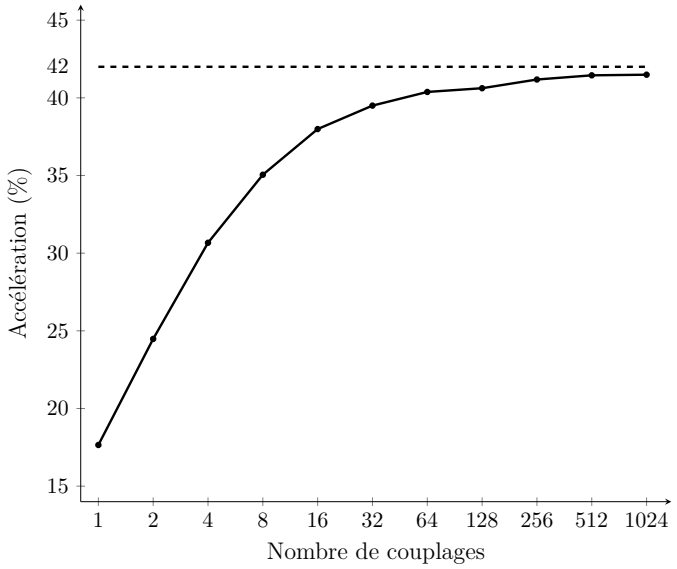

FIGURE 6.3 – Moyenne de l'accélération du calcul FAP par rapport au multi-couplage en fonction du nombre de couplages

TABLE 6.3 – Moyenne de l'accélération en pourcentage, du temps d'exécution en microseconde du multi-couplage, du pré-calcul FAP et du calcul FAP, et taille des données pré-calculées en kilo-octet en fonction du nombre de couplage

Couplage	Accélération	Multi-couplage	Pré-calcul	Calcul	Taille
1	17.92	240	39	197	18
2	24.62	333	78	251	36
4	30.71	521	156	361	71
8	35.09	892	313	579	141
16	38	1637	627	1015	281
32	39.48	3133	1256	1896	562
64	40.37	6123	2515	3651	1123
128	40.62	12111	5014	7191	2246
256	41.18	24366	10102	14332	4491
512	41.45	48860	20171	28609	8981
1024	41.49	97349	40259	56959	17961

de couplages, de 17.65% pour 1 couplage jusqu'à 35.05% pour 8 couplages. Puis, l'accélération croit beaucoup plus lentement pour atteindre une valeur limite ne dépassant pas 42%.

6.4.2 Méthode d'évaluation des optimisations

Les évaluations réalisées consistent en l'évaluation du temps d'exécution des optimisations, et en l'évaluation de la taille des pré-calculs.

```
int main() {
    string access_tree, attributs, ciphertext, k1, k2;
    bool result;
    OpenABEContextKPGPSW kpabe;

    for (int nb_attr = 1; nb_attr <= 1024; nb_attr *= 2) {

        gen_attr(attributs, access_tree, nb_attr);

        for (int iter = 1; iter <= 1024; iter++) {

            kpabe.setup();

            kpabe.encrypt(attributs, k1, ciphertext);

            kpabe.keygen(access_tree);

            kpabe.decrypt(ciphertext, k2);
            result = (k1 == k2);

            kpabe.decrypt(ciphertext, k2);
            result = (k1 == k2);
        }

        write_to_file();
    }
}
```

Listing 6.2 – Fonction réalisant l'évaluation des performances des optimisations

L'évaluation du temps d'exécution est effectuée en utilisant la fonction donnée en Listing 6.2. Cette fonction contient deux boucles imbriquées : la première boucle fait varier le nombre d'attributs de 1 à 1024 par puissance de deux ; la seconde boucle fait varier le nombre d'itérations de 1 à 1024 par pas de un.

Le corps de la première boucle imbriquée permet de générer la liste d'attributs et l'arbre d'accès. Cette génération est réalisée de la même manière qu'en Section 5.2.1 du chapitre précédent, à savoir les attributs sont spécifiés sous la forme "aXXX" avec XXXX une valeur commençant à 0000 et augmentant jusqu'au nombre d'attributs désiré. Lorsque quatre attributs sont requis, cette fonction renvoie une liste d'attributs sous la forme "a0000|a0001|a0002|a0003", et un arbre d'accès contenant uniquement des nœuds internes ET sous la forme "a0000 ET a0001 ET a0002 ET a0003".

Le corps de la dernière boucle imbriquée réalise les appels aux algorithmes du schéma KP-ABE. L'algorithme de déchiffrement est appelé deux fois car de nombreuses optimisations nécessitent de réaliser un pré-calcul lors du premier appel avant de pouvoir bénéficier de la réduction du temps de calcul par l'optimisation lors du second appel.

Avant et après chaque exécution des algorithmes KP-ABE, le temps d'exécution de l'algorithme est mesuré et stocké dans un tableau. Puis lorsque toutes les itérations sont effectuées, les résultats des mesures sont écrits dans un fichier de type Comma-

Separated Values (csv) lors de l'exécution de la fonction write_to_file.

Notons qu'OpenABE implémente des mécanismes fournissant un niveau de sécurité plus élevé au moyen d'algorithmes s'exécutant avant et après chaque algorithme du schéma KP-ABE, le protocole du chapitre précédent a été évalué en utilisant ces mécanismes. En revanche, dans ce chapitre, nous souhaitons évaluer uniquement le temps de calcul des algorithmes du schéma KP-ABE pour observer l'impact des optimisations sur le schéma en lui-même sans inclure les sur-coûts introduits par ces mécanismes. Ainsi, nous avons évalué les optimisations en utilisant les algorithmes du schéma KP-ABE directement tels qu'ils sont décrits dans la section précédente.

La taille des pré-calculs a été évaluée en analysant la définition des types définis dans OpenABE et RELIC, et en utilisant la fonction `sizeof`. De cette analyse, nous pouvons en tirer que la taille d'un élément de $\mathbb{G}_1$ est de 96 octets, et la taille d'un élément de $\mathbb{G}_2$ est de 192 octets.

6.4.3 Évaluation du déchiffrement

6.4.3.1 Schéma Large Universe

La moyenne du temps d'exécution du déchiffrement de la forme LU.Base, les moyennes de l'accélération des formes et optimisations du schéma Large Universe vis-à-vis de LU.Base sont données en Table 6.4.

De cette figure, nous pouvons observer que l'optimisation prodDi de la forme LU permet de gagner, selon le nombre d'attributs, de 1.42% pour 1 attribut à 30.89% pour 1024 attributs. Cette optimisation n'est en conflit avec aucune autre optimisation, elle peut être toujours réalisée.

L'optimisation di de la forme LU permet de gagner du temps de calcul pour un faible nombre d'attributs, de 5.02% pour 1 attribut à 25.05% pour 32 attributs, puis le gain diminue pour de plus grand nombre d'attributs pour atteindre 13.03% pour 1024 attributs.

La forme LU_di sans optimisation n'apporte pas de gains en temps de calcul, cela s'explique par des exponentiations sur des éléments de $\mathbb{G}_2$ dans la forme LU_di au lieu de $\mathbb{G}_1$ dans la forme LU.

Les optimisations di2 et di3 de la forme LU_di sont en conflit. Pour des attributs de 1 à 128, l'optimisation di2 permet de gagner moins de temps de calcul que l'optimisation di, 1.14% (-3.88%) pour 1 attribut, 9.92% (-15.13%) pour 32 attributs, et 18.63% (-0.62%) pour 128 attributs. Cependant, à partir de 256 attributs, cette optimisation permet de gagner plus de temps de calcul que l'optimisation di, à savoir 25.26% (+9.99%) pour 256 attributs, 28.05% (+13.95%) pour 512 attributs, et 28.89% (+15.86%) pour 1024 attributs. L'optimisation di3 permet de gagner plus de temps de calcul que les optimisations di et di2 quelque soit le nombre d'attributs, de 5.49% (+0.47% sur di) pour 1 attribut à 40.9% (+12.01% sur di2) pour 1024 attributs.

Ainsi, en combinant l'optimisation prodDi et l'optimisation di3 de la forme LU_di, nous avons la forme qui permet de réduire le plus le temps de calcul, de 6.91% pour

TABLE 6.4 – Moyenne du temps d'exécution en microseconde du déchiffrement de la forme LU.Base, moyennes de l'accélération en pourcentage des formes et optimisations du schéma Large Universe vis-à-vis de LU.Base

Attribut	LU.Base	LU.prodDi	LU.di	LU_di.Base	LU_di.di2	LU_di.di3	LU_di.prodDi_di3
1	511	1.42	5.02	-2.8	1.14	5.49	6.91
2	633	3.03	9.98	-9.75	2.26	11.8	14.83
4	885	5.17	15.68	-14.97	3.87	19.31	24.48
8	1388	7.39	20.8	-28.28	5.22	26.19	33.58
16	2438	9.64	24.29	-36.1	7.63	31	40.64
32	4664	12.15	25.05	-35.9	9.92	34.39	46.54
64	9691	15.69	23.09	-26.32	13.45	36.36	52.05
128	22043	20.64	19.25	-34.91	18.63	38.15	58.79
256	54981	26.23	15.27	-23.62	25.26	39.34	65.57
512	125146	29.59	14.1	-16.6	28.05	40.76	70.35
1024	266582	30.89	13.03	-13.73	28.89	40.9	71.79

1 attribut à 71.79% pour 1024 attributs. Pour des nombres d'attributs plus proches de notre contexte, de 1 à 64 attributs, la réduction du temps de calcul est également conséquente, jusqu'à 52.05% pour 64 attributs, ce qui est très avantageux dans notre contexte.

En ce qui concerne le coût de stockage, l'optimisation prodDi étant basée sur un pré-calcul Stream, elle dépend de la clé de déchiffrement et de l'ensemble d'attributs du chiffré. Cependant, une seule valeur dans $\mathbb{G}_1$ (96 octets) est à stocker par ensemble minimal d'attributs du chiffré permettant de satisfaire l'arbre d'accès de la clé de déchiffrement, ainsi cette optimisation nécessite peu de capacité de stockage. L'optimisation di étant basée sur le FAP, elle demande de stocker une valeur pré-calculée FAP par élément de la clé de déchiffrement, soit environ 18 kilo-octets par élément de la clé de déchiffrement. Les optimisations di2 et di3 sont des optimisations Stream, elles nécessitent de stocker un pré-calcul par élément de la clé de déchiffrement et par ensemble minimal d'attributs. L'optimisation di3 est plus coûteuse en stockage que l'optimisation di2 car elle nécessite de stocker des valeurs pré-calculées FAP dans la première, alors que la deuxième nécessite de stocker des éléments de $\mathbb{G}_2$ (192 octets).

6.4.3.2 Schéma Small Universe

La moyenne du temps d'exécution du déchiffrement de la forme SU.Base, les moyennes de l'accélération des formes et optimisations du schéma Small Universe vis-à-vis de SU.Base sont données en Table 6.5.

De cette figure, nous pouvons observer que, comme attendu, les formes SU_Ei.Base et SU_Inv_Di.Base permettent de réduire le temps de calcul à partir de 2 attributs, de 27.96% pour 2 attributs à 49.65% pour 1024 attributs, du fait des exponentiations réalisées sur des éléments de $\mathbb{G}_2$ au lieu d'éléments de $\mathbb{G}_T$.

TABLE 6.5 – Moyenne du temps d'exécution en microseconde du déchiffrement de la forme SU.Base, moyennes de l'accélération en pourcentage des formes et optimisations du schéma Small Universe vis-à-vis de SU.Base

Attribut	SU.Base	SU_Ei.Base	SU_Di.Base	SU_Di.Di	SU_Inv_Ei.Di2	SU_Inv_Di.Di3
1	251	-1.94	-1.72	0.43	10.58	12.71
2	559	27.96	34.71	37.12	47.19	49.56
4	1168	41.01	49.65	52.53	62.24	65.13
8	2608	46.78	60.32	63.41	71.86	74.97
16	5438	49.2	64.25	67.73	75.35	78.78
32	10892	50.56	64.86	69.17	76.03	80.3
64	21019	51.65	62.56	68.74	73.77	80.23
128	48882	50.4	65.02	73.36	74.27	82.53
256	103162	50.05	61.68	74.77	69.58	82.72
512	213033	49.82	59.49	75.42	67.08	83.04
1024	433560	49.65	58.27	75.5	65.74	82.95

Notons que le temps d'exécution de la forme SU_Inv.Base est le même que pour la forme SU.Base, l'accélération de la forme SU_Inv_Di.Base est la même que pour la forme SU_Ei.Base, l'accélération de la forme SU_Inv_Ei.Base est la même que pour la forme SU_Di.Base, et l'accélération de la forme SU_Inv_Di.Di est la même que pour la forme SU_Di.Di

TABLE 6.6 – Moyennes de l'accélération en pourcentage du déchiffrement des formes et optimisations Small Universe vis-à-vis de la forme LU.Base

Attribut	SU.Base	SU_Ei.Base	SU_Di.Base	SU_Di.Di	SU_Inv_Ei.Di2	SU_Inv_Di.Di3
1	50.82	49.86	49.97	51.03	56.02	57.07
2	11.75	36.42	42.38	44.51	53.39	55.48
4	-32.06	22.1	33.51	37.31	50.13	53.96
8	-87.87	0.02	25.45	31.26	47.13	52.97
16	-123.03	-13.3	20.28	28.02	45.01	52.67
32	-133.53	-15.45	17.93	27.99	44.02	54
64	-116.89	-4.86	18.79	32.21	43.11	57.13
128	-121.76	-9.99	22.43	40.93	42.95	61.27
256	-87.63	6.28	28.09	52.66	42.91	67.58
512	-70.23	14.58	31.05	58.16	43.97	71.13
1024	-62.64	18.11	32.13	60.15	44.28	72.27

Notons que l'accélération de la forme SU_Inv.Base est la même que pour la forme SU.Base, l'accélération de la forme SU_Inv_Di.Base est la même que pour la forme SU_Ei.Base, l'accélération de la forme SU_Inv_Ei.Base est la même que pour la forme SU_Di.Base, et l'accélération de la forme SU_Inv_Di.Di est la même que pour la forme SU_Di.Di

Par ailleurs, les formes SU_Di.Base et SU_Inv_Ei.Base permettent de réduire le temps de calcul vis-à-vis des formes SU_Ei.Base et SU_Inv_Di.Base à partir de 2 attributs également, de 34.71% (+6.75%) pour 2 attributs à 58.27% (+8.62%) pour 1024 attributs, du fait d'exponentiations réalisées sur des éléments de $\mathbb{G}_1$ au lieu d'éléments de $\mathbb{G}_2$.

Les optimisations Di, Di2 et Di3 permettent toutes de réduire le temps de calcul par rapport aux formes précédentes. Nous allons nous intéresser uniquement à l'optimisation Di3 de la forme SU_Inv_Di qui permet de réduire le plus le temps de calcul par rapport aux autres formes et optimisations. Cette optimisation permet de gagner 12.71% pour 1 attribut (+2.13% sur l'optimisation Di2 de la forme SU_Inv_Ei), jusqu'à 82.95% pour 1024 attributs (+7.45% sur l'optimisation Di de la forme SU_Di). Dans notre contexte, pour 64 attributs, cette optimisation permet de gagner 80.23% (+6.46% sur l'optimisation Di2 de la forme SU_Inv_Ei).

En ce qui concerne le coût de stockage, l'optimisation Di2 de la forme SU_Inv_Ei est une optimisation basée sur le FAP, donc 18 kilo-octets sont à stocker par élément de la clé de déchiffrement. L'optimisation Di de la forme SU_Di est une optimisation basée sur le pré-calcul Stream, les éléments à stocker sont dans $\mathbb{G}_2$ donc 192 octets sont à stocker. Enfin, l'optimisation Di3 est une optimisation basée sur le Stream FAP, donc 18 kilo-octets sont à stocker par éléments de la clé par ensemble minimal d'attributs.

6.4.3.3 Comparaison Large Universe et Small Universe

Les moyennes de l'accélération du déchiffrement des formes et optimisations Small Universes vis-à-vis de la forme LU.Base sont données en Table 6.6.

De cette table, nous pouvons observer que les formes SU.Base et SU_Inv.Base permettent de réduire le temps de calcul pour 1 attribut de 50.82% et pour 2 attributs de 11.75% sur la forme LU. Puis, pour un nombre plus grand d'attributs, ces formes ne permettent pas d'avoir un gain en temps de calcul, cela provient des exponentiations dans $\mathbb{G}_T$ qui prennent le pas sur le temps de calcul des autres opérations réalisées au déchiffrement de la forme LU.Base.

À l'exception de la forme SU_Ei.Base pour des attributs entre 16 et 128, les autres formes et optimisations permettent de réduire le temps de calcul par rapport à la forme LU. Nous allons nous intéresser à la forme la plus optimisée, à savoir la forme SU_Inv_Di.Di3. Cette optimisation possède des réductions du temps de calcul similaires à la forme LU_di.prodDi_di3 à partir de 256 attributs. De 67.58% pour SU_Inv_Di.Di3 contre 65.57% pour LU_di.prodDi_di3 pour 256 attributs. De 71.13% pour SU_Inv_Di.Di3 contre 70.35% pour LU_di.prodDi_di3 pour 512 attributs. De 72.27% pour SU_Inv_Di.Di3 contre 71.79% pour LU_di.prodDi_di3 pour 1024 attributs. En revanche pour des nombres d'attributs inférieurs à 256, SU_Inv_Di.Di3 permet d'avoir un plus grand gain en temps de calcul sur la forme LU_di.prodDi_di3 : +50.16% pour 1 attribut, +40.65% pour 2 attributs, +29.48% pour 4 attributs, +19.39% pour 8 attributs, +12.03% pour 16 attributs, +7.46% pour 32 attributs, +5.08% pour 64 attributs, et +2.48% pour 128 attributs.

Dans notre contexte, pour des attributs allant de 1 à 64, l'optimisation Di3 de la forme SU_Inv_Di permet de gagner entre 52.67% et 57.13% sur la forme LU.Base. Ces gains sont les plus importants vis-à-vis de toutes les autres formes et optimisations, dont la forme LU_di.prodDi_di3 avec +50.16% pour 1 attribut et +5.08% pour 64 attributs. Ainsi, l'utilisation du schéma Small Universe dans sa forme la plus optimisée est plus avantageuse en temps de calcul au déchiffrement vis-à-vis du schéma Large Universe dans sa forme la plus optimisée.

6.5 Discussion

Pour améliorer la proposition, nous pouvons : adapter le schéma KP-ABE Small Universe pour le protocole, étendre les optimisations à d'autres schémas KP-ABE dont le schéma KP-ABE with Fast Decryption, et optimiser le chiffrement KP-ABE.

6.5.1 Adapter le schéma KP-ABE Small Universe au protocole

L'algorithme setup du schéma KP-ABE Small Universe, décrit en Section 6.3, prend en entrée un ensemble d'entiers permettant de représenter l'univers des attributs du système. Ceci implique que l'univers des attributs doit être déterminé à l'exécution du setup et ne peut plus être changé lors de l'exécution du système. Cependant, nous considérons que ce choix a été fait pour simplifier la définition du schéma, en réalité les attributs peuvent être générés durant l'exécution du système. Dans notre contexte, nous pouvons même considérer qu'après fabrication, le véhicule ne possède que la représentation sous forme de chaînes de caractères des attributs utilisés lors du chiffrement. Puis, le véhicule récupérera les valeurs V_i associées à chaque attributs en communiquant avec l'autorité de confiance. Pour ce faire, le véhicule vérifie si l'attribut V_i est stocké dans son module de sécurité matériel, dans le cas contraire le véhicule envoie une demande à l'autorité de confiance. À la réception de cette demande, l'autorité de confiance vérifie si la valeur V_i est stockée dans sa clé publique. Le cas contraire, elle choisit aléatoirement un exposant v_i dans $\mathbb{Z}_p$, cet exposant est ajouté à sa clé maîtresse MSK, puis elle calcule l'exponentiation $V_i = g_2^{v_i}$, la valeur V_i est ajoutée à la clé publique PK, elle est renvoyée au véhicule et elle est rendue publique. En procédant de cette manière, le schéma KP-ABE Small Universe peut être transformé en un schéma KP-ABE Large Universe. Bien entendu, ces échanges de données entraînent un coût en temps d'exécution, cependant, nous considérons que le véhicule réalise tous ces échanges lors du premier démarrage du véhicule lors d'une phase d'initialisation. Puis, lorsque le conducteur modifie ses règles de contrôle d'accès dans le véhicule et s'il renseigne un attribut n'étant pas présent dans le véhicule, des échanges de données avec l'autorité de confiance sont réalisés, nous considérons que le conducteur ne peut pas modifier ses règles de contrôle d'accès en conduisant, mais uniquement avant ou après un trajet. Ainsi, le coût supplémentaire en temps d'exécution a lieu lorsque le système est hors-ligne (lorsque le véhicule ne réalise pas de trajets) et n'impacte donc pas les performances du protocole.

6.5.2 Étendre les optimisations à d'autres schémas KP-ABE

Les optimisations que nous avons réalisées peuvent s'étendre à d'autres schémas KP-ABE de la littérature, en particulier au schéma KP-ABE Small Universe with Fast Decryption [Hohenberger 2013]. Ce schéma a la particularité de ne nécessiter que deux couplages au déchiffrement, mais il nécessite bien plus de multiplications que les schémas précédent :

$$e(C', \prod_{i \in S}(D_i \prod_{i' \in S/i} Q_{i,i'})^{\omega_i}) / e(\prod_{i \in S} R_i^{\omega_i}, \prod_{i \in S} C_i)$$

avec C' et C_i appartenant au chiffré, D_i, $Q_{i,i'}$ et R_i appartenant à la clé de déchiffrement, les coefficients de reconstruction ω_i renvoyés en utilisant l'ensemble d'attributs du chiffré et l'arbre d'accès de la clé de déchiffrement, et S l'ensemble minimal d'attributs nécessaire pour réaliser le déchiffrement.

La complexité de cet algorithme est donc :

$$|S|^2 \cdot \mathcal{M}_{\mathbb{G}_2} + |S| \cdot (\mathcal{E}_{\mathbb{G}_2} + \mathcal{E}_{\mathbb{G}_1} + \mathcal{M}_{\mathbb{G}_2} + \mathcal{M}_{\mathbb{G}_1}) + 2\mathcal{C} + \mathcal{D}_{\mathbb{G}_T}$$

Plusieurs optimisations sont réalisables sur ce schéma :

— le couplage du numérateur peut être pré-calculé en utilisant le Stream FAP ;

— le pré-calcul Stream peut être appliqué au dénominateur.

On obtient ainsi le déchiffrement optimisé suivant :

$$e_L(C', D'_S) / e(R'_S, \prod_{i \in S} C_i)$$

avec $D'_S = e_R(\prod_{i \in S}(D_i \prod_{i' \in S/i} Q_{i,i'})^{\omega_i})$ le Stream FAP, et $R'_S = \prod_{i \in S} R_i^{\omega_i}$ le pré-calcul Stream.

Le complexité du déchiffrement optimisé est donc :

$$|S| \cdot \mathcal{M}_{\mathbb{G}_2} + \mathcal{C} + \mathcal{C}_L + \mathcal{D}_{\mathbb{G}_T}$$

Pour rappel, le déchiffrement le plus optimisé dans notre étude a le coût suivant :

$$|S| \cdot (\mathcal{C}_L + \mathcal{M}_{\mathbb{G}_T})$$

Le gain est significatif, les multiplications sont plus rapides et le nombre de couplages est de deux dont l'un est optimisé, il serait intéressant d'implémenter et d'évaluer expérimentalement cette optimisation.

6.5.3 Pré-calcul du chiffrement KP-ABE

Le chiffrement KP-ABE consiste essentiellement à obtenir une valeur aléatoire puis à réaliser des exponentiations des hashs des attributs pour le schéma Large Universe ou des valeurs associées aux attributs pour le schéma Small Universe. En considérant le contexte de nos travaux et la méthode de génération des attributs décrite dans le chapitre précédent, certaines exponentiations peuvent être pré-calculées. En effet, le véhicule va émettre régulièrement des données liées aux

capteurs du véhicule telles que des données de vitesse, de position, de température, ou de moteur. Pendant le trajet d'un véhicule, lorsqu'une donnée doit être chiffré, les attributs utilisés sont composés d'attributs statiques, qui sont établis avant le trajet du véhicule, et d'attributs dynamiques, qui changent à chaque chiffrement. Les attributs statiques consistent en les attributs issus des *couples attributs du conducteur*, les identités du centre de stockage et du véhicule, et le type de donnée. Les attributs dynamiques consistent en la date et l'heure d'envoi de la donnée. Le véhicule peut alors réaliser des pré-calculs des exponentiations des attributs statiques lorsque les calculateurs du véhicules ne sont pas occupés. Le véhicule peut réaliser la majorité du chiffrement en amont et réaliser les dernières exponentiations en lien avec les attributs dynamique lorsque la donnée à chiffrer est connue. Ainsi, en considérant un compromis entre l'espace de stockage et la temps de calcul, le temps de calcul du chiffrement peut être réduit à quelques exponentiations.

6.6 Conclusion

Des optimisations des schémas KP-ABE Large Universe et Small Universe ont été présentées dans ce chapitre. Ces optimisations sont réalisées à partir de pré-calculs de valeurs lors du déchiffrement, ces pré-calculs sont réutilisés lors des déchiffrements successifs. Les optimisations reposent également sur la modification des groupes des éléments du chiffré et de la clé, ainsi que la modification de l'ordre des opérations pour réduire le temps de calcul ou pour permettre de bénéficier de certains pré-calculs. De l'analyse théorique des optimisations, nous avons pu déduire que les temps de calcul du déchiffrement peuvent être réduits et qu'un schéma Small Universe peut être plus avantageux qu'un schéma Large Universe pour un faible nombre d'attributs. L'évaluation expérimentale a permis de confirmer que les optimisations apportent un gain non négligeable et que le schéma KP-ABE Small Universe dans sa forme la plus optimisée est plus avantageuse que le schéma KP-ABE Large Universe dans sa forme la plus optimisée pour des nombres d'attributs correspondant à notre contexte. Nous avons discuté également de l'adaptation du schéma KP-ABE Small Universe au protocole en permettant la génération des attributs au cours de l'exécution du système ce qui permet de le transformer en un schéma KP-ABE Large Universe. Nous avons discuté des optimisations du schéma KP-ABE Small Universe with Fast Decryption permettant dans sa forme la plus optimisée de gagner encore plus en temps de calcul que la forme la plus optimisée de notre étude. Enfin, nous avons discuté du pré-calcul du chiffrement KP-ABE en réalisant le pré-calcul des exponentiations des attributs statiques, ce qui permet de réduire le temps de calcul du chiffrement d'une donnée aux exponentiations des attributs dynamiques.

Conclusion

Bilan

L'architecture des véhicules est aujourd'hui composée de nombreux capteurs, actionneurs et d'équipements permettant la connectivité envoyant de nombreuses données vers un Cloud chargé du stockage et du partage des données avec de nombreuses parties prenantes.

Dans ce contexte se posent des problématiques de sécurité et de vie privée. Tout d'abord, le Cloud peut être victime d'attaques pouvant entraîner une fuite des données. Par ailleurs, pour protéger leur vie privée, les conducteurs peuvent ne pas avoir suffisamment confiance dans un Cloud pouvant accéder aux données en clair, ils ont le besoin de vouloir contrôler l'accès à leurs données d'autant plus depuis la mise en application du RGPD. Enfin, les autorités souhaitent réglementer l'accès aux données des parties prenantes via la législation, les solutions techniques doivent être adaptées pour la prendre en compte.

Ainsi, il est nécessaire de mettre en place des mécanismes cryptographiques permettant de protéger la confidentialité des données lors de leur transfert et de leur stockage contre toute entité honnête mais curieuse et contre un attaquant externe. Il est également nécessaire de mettre en place un contrôle d'accès intégré au mécanisme cryptographique, ce contrôle d'accès doit pouvoir prendre en compte les choix de partage du conducteur ainsi que la législation.

Les mécanismes cryptographiques classiques (e.g. chiffrement asymétrique classique, chiffrement de groupe) ne sont pas adaptés pour établir un contrôle d'accès car ils nécessitent de stocker de nombreuses clés, de chiffrer plusieurs fois une même donnée, ou ne sont pas assez flexibles. Un mécanisme adapté est le chiffrement basé attributs (ABE) reposant sur les notions d'attributs et d'arbres d'accès pour mettre en place le contrôle d'accès. Ce chiffrement est généralement considéré coûteux, notamment au déchiffrement de part la réalisation de nombreuses exponentiations et couplages. La plupart des travaux de la littérature autour de l'application du chiffrement basé attributs dans la protection des données des véhicules connectés se concentre sur la proposition de nouveaux schémas cryptographiques intégrant de nouvelles fonctionnalités ou de nouvelles propriétés mais très peu proposent un protocole bien défini détaillant les échanges entre les différentes entités du système. De plus, dans ces travaux, la sécurité du protocole est généralement évaluée via une discussion et n'est donc pas évaluée via une méthode plus robuste telle que l'utilisation d'outils de vérification formelle automatique des propriétés de protocole. Par ailleurs, les travaux de la littérature autour des méthodes de génération du contrôle d'accès sont également limités, ils ne prennent pas en compte la législation permettant de définir des règles d'accès autorisé ou interdit aux données sous la forme d'un contrôle d'accès cryptographique. Enfin, les travaux de la littérature autour des optimisations du déchiffrement KP-ABE peuvent être améliorés en pré-calculant

des éléments déterministes issus du chiffré.

Pour faire face à ces besoins, cette thèse s'articule autour de trois contributions. La première contribution consiste en un protocole cryptographique s'appuyant sur le chiffrement basé attributs afin de permettre la mise en place d'un contrôle d'accès flexible et personnalisable, c'est-à-dire que le contrôle d'accès repose sur l'utilisation d'attributs utilisés au chiffrement qui peuvent évoluer dans le temps. La sécurité du protocole est évaluée en considérant des propriétés sur la confidentialité, l'intégrité, l'authenticité, et la disponibilité des messages. La vérification des propriétés est réalisée via l'outil de vérification automatique Proverif et cette évaluation a montré que l'ensemble des propriétés sont satisfaites. Des scénarios de fuites des clés de déchiffrement ABE de chaque entité ont également été envisagés. L'évaluation a montré que le Cloud n'est pas en mesure de déchiffrer les données, que le véhicule peut déchiffrer uniquement les données qu'il a émises, et que les parties prenantes ne peuvent déchiffrer que les données auxquelles elles ont accès.

La deuxième contribution consiste, à partir de la législation, en une méthode de génération du matériel cryptographique nécessaire au chiffrement basé attributs permettant de mettre en place le contrôle d'accès. Cette méthode consiste à traduire la législation dans le formalisme de contrôle d'accès OrBAC, ce formalisme est ensuite utilisé pour générer les arbres d'accès de chaque entité ainsi que les attributs qui doivent être utilisés au chiffrement. Par ailleurs, cette méthode permet également de générer des attributs utilisés au chiffrement à partir des contrats du conducteur et de son consentement à partager ses données. La méthode est illustrée via un cas d'étude concret qui montre la faisabilité de notre approche.

Nous proposons également un prototype open source implémentant le protocole cryptographique. Nous avons évalué les performances des étapes du protocole coté véhicule en faisant varier le nombre d'attributs et la taille de la donnée. Les performances considérées sont le temps de calcul, la taille en mémoire et la taille des messages. Les résultats de l'analyse des performances montrent que ce protocole est réaliste et peut être implémenté dans les calculateurs des véhicules connectés.

Enfin, pour accélérer le traitement des données par les parties prenantes qui devront traiter une grande quantité de données issus de nombreux véhicules, nous avons défini des optimisations du déchiffrement KP-ABE se basant sur des pré-calculs du couplage et sur l'ordre des opérations. Le bénéfice en temps de calcul se fait au dépend de valeurs à stocker de la partie droite du couplage dépendent à la fois de la clé de déchiffrement et des attributs utilisés lors du chiffrement, soit 18 kilo-octets à stocker par valeur de la partie droite du couplage. L'évaluation des performances des optimisations nous montre que pour le schéma KP-ABE Large Universe, sa forme la plus optimisée permet un gain en temps de calcul allant de 6.91% pour 1 attribut à 52.05% pour 64 attributs. Pour le schéma KP-ABE Small Universe le gain est de 12.71% pour 1 attribut à 80.23% pour 64 attributs. Les optimisations permettent ainsi de réduire significativement le temps de calcul du déchiffrement.

Globalement, nos travaux ont montré que :

1. dans le cadre des véhicules connectés, il est possible de réaliser des protocoles cryptographiques vérifiés formellement s'appuyant sur des schémas cryptographiques

génériques ;
2. la législation peut être intégrée dans la génération des règles de contrôle d'accès et être appliquée via le protocole cryptographique précédemment défini ;
3. les contrats et les souhaits des conducteurs peuvent être pris en compte dans la génération des règles de contrôle d'accès ;
4. le temps de calcul du protocole peut être réduit en prenant en compte les optimisations précédemment identifiées.

Perspectives

Plusieurs perspectives à ces travaux sont envisageables, la plupart sont directement liées aux défis identifiés dans les sections "Discussion" des chapitres précédents.

Premièrement, le protocole peut être amélioré en permettant l'anonymat des véhicules. En effet, les attributs utilisés au chiffrement dans la plupart des schémas ABE sont accessibles en clair par n'importe quelle entité ayant accès au message. Or, l'identité du véhicule est intégrée sous forme d'attributs systématiquement lors du chiffrement d'une donnée. Ainsi, une entité pouvant accéder aux messages serait en mesure de regrouper tous les chiffrés envoyés par le même véhicule, et d'obtenir des informations sur ce véhicule en exploitant les autres attributs utilisés pour chiffrer la donnée. Des schémas ABE permettant de cacher les attributs, en totalité ou partiellement, lors du chiffrement peuvent être utilisés pour pallier à ce problème.

Deuxièmement, la génération du contrôle d'accès peut être améliorée en prenant en compte l'évolution de la législation lors de l'ajout ou de la suppression d'articles de la loi autorisant ou interdisant des accès aux données. Par exemple, un début de réflexion peut amener aux règles suivantes, mais nécessiterait d'être étudié en profondeur :

— l'ajout d'un article de loi interdisant un accès nécessite une nouvelle règle Interdiction dans le véhicule et une vérification lors de la mise à jour du véhicule afin de supprimer les couples d'attributs de la fonction *get_attrs* en conflit avec la règle ;

— la suppression d'un article de loi interdisant un accès nécessite uniquement une mise à jour du véhicule sans entraîner de vérification ;

— l'ajout d'un article de loi autorisant un accès nécessite de générer une règle de type Permission et de mettre à jour les clés de déchiffrement des parties prenantes pour les autoriser à accéder aux données spécifiées dans la règle ;

— la suppression d'un article de loi autorisant un accès nécessite de générer de nouvelles clés de déchiffrement plus restrictives ainsi que de révoquer les anciennes clés de déchiffrement.

Troisièmement, une évaluation à plus grande échelle de notre proposition doit être réalisée. En particulier, il serait intéressant de réaliser une évaluation du prototype en considérant le Cloud et les parties prenantes. Cette évaluation pourrait également être réalisée sur les calculateurs d'un véhicule en prenant en compte les contraintes réelles

de performances et de temps réel du calculateur. En effet, il n'a malheureusement pas été possible dans le cadre de cette thèse de réaliser une expérimentation sur un cas d'étude concret, sur des véhicules réels connectés à un cloud avec de réelles parties prenantes, même si nos expérimentations ont été réalisées afin de correspondre au mieux à la réalité. Un cas d'utilisation concret serait toutefois nécessaire pour évaluer au mieux la taille de stockage requise et le temps de calcul des étapes du protocole.

Quatrièmement, de multiples pistes sont encore à explorer pour les optimisations des différents schémas ABE. En effet, les optimisations définies peuvent être étendues à d'autres schémas KP-ABE tels que le schéma "KP-ABE with Fast Decryption". Par ailleurs, nous avons uniquement réalisé des optimisations sur le déchiffrement dans le cadre de cette thèse, par manque de temps essentiellement et parce que les idées des optimisations que nous avons eues concernaient le déchiffrement. Mais il est bien évident que le chiffrement, réalisé par les véhicules, qui vont nécessairement disposer de ressources limitées, doit également être étudié à des fins d'optimisations. Les optimisations peuvent être réalisées sur le chiffrement KP-ABE en considérant la politique de contrôle d'accès qui repose majoritairement sur des attributs statiques lors d'un trajet d'un véhicule.

Une cinquième perspective à ce travail concerne l'instanciation réelle de la méthode proposée pour dériver les attributs et arbres d'accès depuis la législation. En effet, la législation est en pleine évolution sur le sujet et nous n'avons pas pour le moment travaillé sur des cas concrets réels, puisqu'ils ne sont pas suffisamment explicites aujourd'hui. Il serait intéressant, lorsque ces lois seront promulguées de tester cette méthode sur ces cas d'usages précis. Par ailleurs, il faut également vérifier qu'un spécialiste de la loi est à même de réaliser ce travail, au moins pour la première partie de notre méthode, car elle doit être réalisée par des gens compétents.

Une autre perspective à ce travail peut être envisagée dans laquelle le temps de calcul des algorithmes ABE pourrait être réduit à l'aide de moyens matériels dédiés. Il serait par exemple intéressant d'étudier la réalisation d'un accélérateur matériel pour le chiffrement ABE et de l'évaluer dans une étude de cas d'un véhicule connecté. Pour les schémas KP-ABE Small Universe et Large Universe, le chiffrement consiste essentiellement en la réalisation d'exponentiations, l'accélérateur matériel devra principalement fournir une implémentation matériel d'une exponentiation.

Enfin, le protocole cryptographique est détaillé dans le contexte du véhicule connecté, mais ce protocole établit des scénarios d'envoi de données génériques permettant de prendre en compte des règles de contrôle d'accès. Il serait intéressant d'étudier l'application de ce protocole à d'autres domaines tels que celui de l'Internet des Objets pour lequel de nombreuses données contenant des informations personnelles sont également envoyées dans des clouds et dont les objets sont contraints en terme de performances.

Bibliographie

[Adelin 2022] Rémi Adelin, Cyrius Nugier, Éric Alata, Vincent Nicomette, Vincent Migliore et Mohamed Kaâniche. *Facing emerging challenges in connected vehicles : a formally proven, legislation compliant, and post-quantum ready security protocol.* Journal of Computer Virology and Hacking Techniques, vol. 18, no. 4, pages 425–452. Springer, 2022. (Cité en page 3.)

[Agarwal 2018] Sushant Agarwal, Simon Steyskal, Franjo Antunovic et Sabrina Kirrane. *Legislative compliance assessment : framework, model and GDPR instantiation.* Dans Privacy Technologies and Policy : 6th Annual Privacy Forum (APF), pages 131–149. Springer, 2018. (Cité en page 35.)

[Aranha 2022] Diego F. Aranha, Conrado P. L. Gouvêa, Tobias Markmann, Riad S. Wahby et Kevin Liao. *RELIC is an Efficient LIbrary for Cryptography.* `https://github.com/relic-toolkit/relic`, 2022. (Cité en pages 39 et 94.)

[Armando 2005] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani *et al. The AVISPA tool for the automated validation of internet security protocols and applications.* Dans Computer Aided Verification : 17th International Conference (CAV), pages 281–285. Springer, 2005. (Cité en page 49.)

[Avizienis 2004] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell et Carl Landwehr. *Basic concepts and taxonomy of dependable and secure computing.* IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pages 11–33. IEEE, 2004. (Cité en page 9.)

[Bartolini 2019] Cesare Bartolini, Said Daoudagh, Gabriele Lenzini et Eda Marchetti. *GDPR-based user stories in the access control perspective.* Dans Quality of Information and Communications Technology : 12th International Conference (QUATIC), pages 3–17. Springer, 2019. (Cité en pages xi, 36 et 37.)

[Baudet 2013] Mathieu Baudet, Véronique Cortier et Stéphanie Delaune. *YAPA : A generic tool for computing intruder knowledge.* ACM Transactions On Computational Logic (TOCL), vol. 14, no. 1, pages 1–32. ACM, 2013. (Cité en page 49.)

[Belguith 2018] Sana Belguith, Nesrine Kaâniche, Maryline Laurent, Abderrazak Jemai et Rabah Attia. *Phoabe : Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot.* Computer Networks, vol. 133, pages 141–156. Elsevier, 2018. (Cité en page 38.)

[Bethencourt 2007] John Bethencourt, Amit Sahai et Brent Waters. *Ciphertext-Policy Attribute-Based Encryption.* Dans 2007 IEEE Symposium on Security and Privacy (SP), pages 321–334. IEEE, 2007. (Cité en page 23.)

[Blanchet 2008] Bruno Blanchet et Avik Chaudhuri. *Automated Formal Analysis of a Protocol for Secure File Sharing on Untrusted Storage.* Dans 2008 IEEE Symposium on Security and Privacy (SP), pages 417–431. IEEE, 2008. (Cité en page 49.)

[Blanchet 2017] Bruno Blanchet. *Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols.* Dans 2017 IEEE 30th Computer Security Foundations Symposium (CSF), pages 68–82. IEEE, 2017. (Cité en page 49.)

[Blanchet 2021] Bruno Blanchet, Ben Smyth, Vincent Cheval et Marc Sylvestre. *ProVerif 2.04 : automatic cryptographic protocol verifier, user manual and tutorial.* `https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf`, 2021. (Cité en page 49.)

[Boneh 2001] Dan Boneh et Matt Franklin. *Identity-based encryption from the Weil pairing.* Dans CRYPTO 2001 : 21st Annual International Cryptology Conference, pages 213–229. Springer, 2001. (Cité en page 42.)

[Boneh 2014] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan et Dhinakaran Vinayagamurthy. *Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits.* Dans EUROCRYPT 2014 : 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 533–556. Springer, 2014. (Cité en page 42.)

[Brakerski 2014] Zvika Brakerski, Craig Gentry et Vinod Vaikuntanathan. *(Leveled) fully homomorphic encryption without bootstrapping.* ACM Transactions on Computation Theory (TOCT), vol. 6, no. 3, pages 1–36. ACM, 2014. (Cité en page 23.)

[Canetti 2001] Ran Canetti et Hugo Krawczyk. *Analysis of key-exchange protocols and their use for building secure channels.* Dans EUROCRYPT 2001 : International Conference on the Theory and Application of Cryptographic Techniques, pages 453–474. Springer, 2001. (Cité en page 27.)

[Chandrasekaran 2020] Balaji Chandrasekaran, Yasuyuki Nogami et Ramadoss Balakrishnan. *An efficient file hierarchy attribute based encryption using optimized tate pairing construction in cloud environment.* Journal of Applied Security Research, vol. 15, no. 2, pages 270–278. Taylor & Francis, 2020. (Cité en page 38.)

[Chaum 1991] David Chaum et Eugène Van Heyst. *Group signatures.* Dans EUROCRYPT 1991 : Workshop on the Theory and Application of Cryptographic Techniques, pages 257–265. Springer, 1991. (Cité en page 43.)

[Checkoway 2011] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner et Tadayoshi Kohno. *Comprehensive Experimental Analyses of Automotive Attack Surfaces.* Dans 20th USENIX Security Symposium. USENIX Association, 2011. (Cité en page 15.)

[Chen 2005] Liqun Chen et Zhaohui Cheng. *Security proof of Sakai-Kasahara's identity-based encryption scheme.* Dans Cryptography and Coding : 10th IMA International Conference, pages 442–459. Springer, 2005. (Cité en page 34.)

[Chen 2010] Liqun Chen et Mark Ryan. *Attack, solution and verification for shared authorisation data in TCG TPM.* Dans Formal Aspects in Security and Trust : 6th International Workshop (FAST), pages 201–216. Springer, 2010. (Cité en page 49.)

[Cheon 2017] Jung Hee Cheon, Andrey Kim, Miran Kim et Yongsoo Song. *Homomorphic encryption for arithmetic of approximate numbers.* Dans ASIACRYPT 2017 : 23rd International Conference on the Theory and Applications of Cryptology and Information Security, pages 409–437. Springer, 2017. (Cité en page 23.)

[Commission Informatique et Libertés 1974] Commission Informatique et Libertés. *Rapport Tricot.* `https://www.cnil.fr/sites/default/files/atoms/files/rapport_tricot_1975_vd.pdf`, 1974. [Online ; accessed January-2022]. (Cité en page 17.)

[Costello 2010] Craig Costello et Douglas Stebila. *Fixed argument pairings.* Dans LATINCRYPT 2010 : First International Conference on Cryptology and Information Security in Latin America, pages 92–108. Springer, 2010. (Cité en page 39.)

[Cramer 2002] Ronald Cramer et Victor Shoup. *Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption.* Dans EUROCRYPT 2002 : International Conference on the Theory and Applications of Cryptographic Techniques, pages 45–64. Springer, 2002. (Cité en page 22.)

[Daemen 1999] Joan Daemen et Vincent Rijmen. *Aes proposal : Rijndael, aes algorithm submission*, 1999. (Cité en page 43.)

[de la Piedra 2022] Antonio de la Piedra, Marloes Venema et Greg Alpár. *ABE Squared : Accurately Benchmarking Efficiency of Attribute-Based Encryption.* IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES), no. 2, page 192–239. TCHES, 2022. (Cité en page 39.)

[Deswarte 1991] Yves Deswarte, Laurent Blain et Jean-Charles Fabre. *Intrusion tolerance in distributed computing systems.* Dans 1991 IEEE Computer Society Symposium on Research in Security and Privacy, pages 110–121. IEEE, 1991. (Cité en page 12.)

[Diffie 1976] Whitfield Diffie et Martin Hellman. *New directions in cryptography.* IEEE Transactions on Information Theory, vol. 22, no. 6, pages 644–654. IEEE, 1976. (Cité en page 22.)

[Dolev 1983] Danny Dolev et Andrew Yao. *On the security of public key protocols.* IEEE Transactions on Information Theory, vol. 29, no. 2, pages 198–208. IEEE, 1983. (Cité en pages 27 et 49.)

[Domingo-Ferrer 2019] Josep Domingo-Ferrer, Oriol Farras, Jordi Ribes-González et David Sánchez. *Privacy-preserving cloud computing on sensitive data : A survey of methods, products and challenges.* Computer Communications, vol. 140, pages 38–60. Elsevier, 2019. (Cité en page 30.)

[ElGamal 1985] Taher ElGamal. *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE Transactions on Information Theory, vol. 31, no. 4, pages 469–472. IEEE, 1985. (Cité en page 22.)

[Ezerman 2015] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen et Huaxiong Wang. *A Provably Secure Group Signature Scheme from Code-Based Assumptions.* Dans ASIACRYPT 2015 : 21st International Conference on the Theory and Application of Cryptology and Information Security, pages 260–285. Springer, 2015. (Cité en page 43.)

[Feng 2020] Chaosheng Feng, Keping Yu, Moayad Aloqaily, Mamoun Alazab, Zhihan Lv et Shahid Mumtaz. *Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV.* IEEE Transactions on Vehicular Technology, vol. 69, no. 11, pages 13784–13795. IEEE, 2020. (Cité en page 32.)

[Gambs 2010] Sébastien Gambs, Marc-Olivier Killijian et Miguel Núñez del Prado Cortez. *Show me how you move and I will tell you who you are.* Dans 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS, pages 34–41. ACM, 2010. (Cité en page 18.)

[Gentry 2008] Craig Gentry, Chris Peikert et Vinod Vaikuntanathan. *Trapdoors for hard lattices and new cryptographic constructions.* Dans 40th annual ACM symposium on Theory of Computing, pages 197–206. ACM, 2008. (Cité en page 42.)

[Gentry 2009] Craig Gentry. *Fully homomorphic encryption using ideal lattices.* Dans 41st annual ACM symposium on Theory of Computing, pages 169–178. ACM, 2009. (Cité en page 23.)

[Gentry 2012] Craig Gentry, Shai Halevi et Nigel P. Smart. *Fully Homomorphic Encryption with Polylog Overhead.* Dans EUROCRYPT 2012 : 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 465–482. Springer, 2012. (Cité en page 23.)

[Gordon 2010] S Dov Gordon, Jonathan Katz et Vinod Vaikuntanathan. *A group signature scheme from lattice assumptions.* Dans ASIACRYPT 2010 : 16th International Conference on the Theory and Application of Cryptology and Information Security, pages 395–412. Springer, 2010. (Cité en page 43.)

[Goyal 2006] Vipul Goyal, Omkant Pandey, Amit Sahai et Brent Waters. *Attribute-based encryption for fine-grained access control of encrypted data.* Dans 13th ACM conference on Computer and Communications Security (CCS), pages 89–98. ACM, 2006. (Cité en pages ix, 23, 94, 120, 123, 127 et 128.)

[Green 2011] Matthew Green, Susan Hohenberger et Brent Waters. *Outsourcing the Decryption of ABE Ciphertexts.* Dans 20th USENIX Security Symposium. USENIX Association, 2011. (Cité en page 38.)

[Guo 2018] Fuchun Guo, Willy Susilo, Yi Mu, Fuchun Guo, Willy Susilo et Yi Mu. *Public-Key Encryption Without Random Oracles*. Introduction to Security Reduction, pages 209–214. Springer, 2018. (Cité en page 34.)

[Hoffstein 1998] Jeffrey Hoffstein, Jill Pipher et Joseph H Silverman. *NTRU : A ring-based public key cryptosystem*. Dans Algorithmic Number Theory Symposium : 3rd International Symposium (ANTS), pages 267–288. Springer, 1998. (Cité en page 22.)

[Hohenberger 2013] Susan Hohenberger et Brent Waters. *Attribute-based encryption with fast decryption*. Dans PKC 2013 : 16th International Conference on Practice and Theory in Public-Key Cryptography, pages 162–179. Springer, 2013. (Cité en pages 38 et 141.)

[Horng 2020] Shi-Jinn Horng, Cheng-Chung Lu et Wanlei Zhou. *An identity-based and revocable data-sharing scheme in VANETs*. IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pages 15933–15946. IEEE, 2020. (Cité en pages 32, 33 et 34.)

[Huang 2009] Dijiang Huang et Mayank Verma. *ASPE : Attribute-based secure policy enforcement in vehicular ad hoc networks*. Ad Hoc Networks, vol. 7, no. 8, pages 1526–1535. Elsevier, 2009. (Cité en page 32.)

[Huang 2018] Qinlong Huang, Yixian Yang et Yuxiang Shi. *SmartVeh : Secure and efficient message access control and authentication for vehicular cloud computing*. Sensors, vol. 18, no. 2, page 666. MDPI, 2018. (Cité en page 32.)

[Huang 2019] Qinlong Huang, Nan Li, Zhicheng Zhang et Yixian Yang. *Secure and Privacy-Preserving Warning Message Dissemination in Cloud-Assisted Internet of Vehicles*. Dans 2019 IEEE Conference on Communications and Network Security (CNS), pages 1–8. IEEE, 2019. (Cité en page 32.)

[ISO 2003] ISO. *Road vehicles — Controller area network (CAN) — Part 2 : High-speed medium access unit*. `https://www.iso.org/standard/33423.html`, 2003. (Cité en page 112.)

[ITU 2011] ITU. *About mobile technology and IMT-2000*. `http://magrawal.myweb.usf.edu/dcom/Ch12_About%20mobile%20technology%20and%20IMT-2000.pdf`, 2011. (Cité en page 112.)

[Kaâniche 2017] Nesrine Kaâniche et Maryline Laurent. *Attribute based encryption for multi-level access control policies*. Dans SECRYPT 2017 : 14th International Conference on Security and Cryptography, volume 6, pages 67–78. Scitepress, 2017. (Cité en page 38.)

[Kalam 2003] Anas Abou El Kalam, R El Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Miege, Claire Saurel et Gilles Trouessin. *Organization based access control*. Dans POLICY 2003 : IEEE 4th International Workshop on Policies for Distributed Systems and Networks, pages 120–131. IEEE, 2003. (Cité en page 74.)

[Ke 2021] Gang Ke, Shi Wang et Huan-huan Wu. *Parallel incremental attribute-based encryption for mobile cloud data storage and sharing*.

Journal of Ambient Intelligence and Humanized Computing, pages 1–11. Springer, 2021. (Cité en page 38.)

[Kiayias 2007] Aggelos Kiayias, Yiannis Tsiounis et Moti Yung. *Group encryption*. Dans ASIACRYPT 2007 : 13th International Conference on the Theory and Application of Cryptology and Information Security, pages 181–199. Springer, 2007. (Cité en page 22.)

[Koscher 2010] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham *et al. Experimental security analysis of a modern automobile*. Dans 2010 IEEE Symposium on Security and Privacy (SP), pages 447–462. IEEE, 2010. (Cité en pages ix et 15.)

[Kremer 2005] Steve Kremer et Mark D. Ryan. *Analysing the Vulnerability of Protocols to Produce Known-pair and Chosen-text Attacks*. Electronic Notes in Theoretical Computer Science, vol. 128, no. 5, pages 87–104. Elsevier, 2005. (Cité en page 49.)

[Krumm 2007] John Krumm. *Inference attacks on location tracks*. Dans Pervasive Computing : 5th International Conference (PERVASIVE), pages 127–143. Springer, 2007. (Cité en page 18.)

[Laguillaumie 2013] Fabien Laguillaumie, Adeline Langlois, Benoît Libert et Damien Stehlé. *Lattice-based group signatures with logarithmic signature size*. Dans ASIACRYPT 2013 : 19th International Conference on the Theory and Application of Cryptology and Information Security, pages 41–61. Springer, 2013. (Cité en page 43.)

[Langlois 2014] Adeline Langlois, San Ling, Khoa Nguyen et Huaxiong Wang. *Lattice-based group signature scheme with verifier-local revocation*. Dans PKC 2014 : 17th International Conference on Practice and Theory in Public-Key Cryptography, pages 345–361. Springer, 2014. (Cité en page 43.)

[Laprie 1996] Jean-Claude Laprie, Jean Arlat, , Yves Deswarte, David Powell, Karama Kanoun, Mohamed Kaâniche et Yves Crouzet. Guide de la sûreté de fonctionnement. Cépaduès, 1996. (Cité en pages ix, 9 et 10.)

[LeMonde 1974] LeMonde. *« Safari » ou la chasse aux français*. `https://www.cnil.fr/sites/default/files/atoms/files/le_monde_0.pdf`, 1974. [Online ; accessed January-2022]. (Cité en page 17.)

[LeMonde 2018] LeMonde. *Cambridge Analytica : 87 millions de comptes Facebook concernés*. `https://www.lemonde.fr/pixels/article/2018/04/04/cambridge-analytica-87-millions-de-comptes-facebook-concernes_5280752_4408996.html`, 2018. [Online ; accessed January-2022]. (Cité en page 17.)

[Li 2013] Jin Li, Xiaofeng Chen, Jingwei Li, Chunfu Jia, Jianfeng Ma et Wenjing Lou. *Fine-grained access control system based on outsourced attribute-based encryption*. Dans ESORICS 2013 : 18th European Symposium on Research in Computer Security, pages 592–609. Springer, 2013. (Cité en page 38.)

[Li 2019] Jiguo Li, Ningyu Chen et Yichen Zhang. *Extended file hierarchy access control scheme with attribute-based encryption in cloud computing.* IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 2, pages 983–993. IEEE, 2019. (Cité en page 38.)

[Libert 2014] Benoît Libert, Moti Yung, Marc Joye et Thomas Peters. *Traceable group encryption.* Dans PKC 2014 : 17th International Conference on Practice and Theory in Public-Key Cryptography, pages 592–610. Springer, 2014. (Cité en page 22.)

[Ling 2015] San Ling, Khoa Nguyen et Huaxiong Wang. *Group signatures from lattices : simpler, tighter, shorter, ring-based.* Dans PKC 2015 : 18th International Conference on Practice and Theory in Public-Key Cryptography, pages 427–449. Springer, 2015. (Cité en page 43.)

[Liu 2016] Xuejiao Liu, Yingjie Xia, Wenzhi Chen, Yang Xiang, Mohammad Mehedi Hassan et Abdulhameed Alelaiwi. *SEMD : Secure and efficient message dissemination with policy enforcement in VANET.* Journal of Computer and System Sciences, vol. 82, no. 8, pages 1316–1328. Elsevier, 2016. (Cité en page 32.)

[LOM 2019] LOM. *LOI n° 2019-1428 du 24 décembre 2019 d'orientation des mobilités.* `https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000039666574`, 2019. [Online ; accessed January-2022]. (Cité en pages 20, 28 et 75.)

[Luo 2018] Wei Luo et Wenping Ma. *Efficient and secure access control scheme in the standard model for vehicular cloud computing.* IEEE Access, vol. 6, pages 40420–40428. IEEE, 2018. (Cité en pages 32, 33 et 35.)

[MAFTIA 2003] MAFTIA. *MAFTIA, Malicious and Accidental Fault Tolerance in Internet Applications : conceptual model and architecture.* Maftia project deliverable d21, Project IST-1999-11583, 2003. (Cité en page 11.)

[Malluhi 2017] Qutaibah M Malluhi, Abdullatif Shikfa et Viet Cuong Trinh. *A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption.* Dans 2017 ACM on Asia conference on Computer and Communications Security (AsiaCCS), pages 230–240. ACM, 2017. (Cité en page 38.)

[Malluhi 2019] Qutaibah M Malluhi, Abdullatif Shikfa, Vinh Duc Tran et Viet Cuong Trinh. *Decentralized ciphertext-policy attribute-based encryption schemes for lightweight devices.* Computer Communications, vol. 145, pages 113–125. Elsevier, 2019. (Cité en page 38.)

[Meier 2013] Simon Meier, Benedikt Schmidt, Cas Cremers et David Basin. *The TAMARIN Prover for the Symbolic Analysis of Security Protocols.* Dans Computer Aided Verification : 25th International Conference (CAV), pages 696–701. Springer, 2013. (Cité en page 49.)

[Micciancio 2007] Daniele Micciancio et Oded Regev. *Worst-case to average-case reductions based on Gaussian measures.* SIAM Journal on Computing, vol. 37, no. 1, pages 267–302. SIAM, 2007. (Cité en page 42.)

[Micciancio 2013] Daniele Micciancio et Chris Peikert. *Hardness of SIS and LWE with small parameters.* Dans CRYPTO 2013 : 33rd Annual International Cryptology Conference, pages 21–39. Springer, 2013. (Cité en page 42.)

[Ostrovsky 2007] Rafail Ostrovsky, Amit Sahai et Brent Waters. *Attribute-based encryption with non-monotonic access structures.* Dans 14th ACM conference on Computer and Communications Security (CCS), pages 195–203. ACM, 2007. (Cité en page 23.)

[Paillier 1999] Pascal Paillier. *Public-key cryptosystems based on composite degree residuosity classes.* Dans EUROCRYPT 1999 : International Conference on the Theory and Application of Cryptographic Techniques, pages 223–238. Springer, 1999. (Cité en page 22.)

[Pan 2019] Jingwen Pan, Jie Cui, Lu Wei, Yan Xu et Hong Zhong. *Secure data sharing scheme for VANETs based on edge computing.* EURASIP Journal on Wireless Communications and Networking, vol. 2019, no. 1, pages 1–11. SpringerOpen, 2019. (Cité en page 32.)

[Peikert 2009] Chris Peikert. *Public-key cryptosystems from the worst-case shortest vector problem.* Dans 41st annual ACM symposium on Theory of Computing, pages 333–342. ACM, 2009. (Cité en page 42.)

[Pirretti 2006] Matthew Pirretti, Patrick Traynor, Patrick McDaniel et Brent Waters. *Secure attribute-based systems.* Dans 13th ACM conference on Computer and Communications Security (CCS), pages 99–112. ACM, 2006. (Cité en page 123.)

[Poznanski 1997] Renée Poznanski. *Le fichage des juifs de France pendant la seconde guerre mondiale et l'affaire du fichier des juifs.* Gazette des archives, vol. 177, no. 1, pages 250–270. Persée-Portail des revues scientifiques en SHS, 1997. (Cité en page 17.)

[Regev 2009] Oded Regev. *On lattices, learning with errors, random linear codes, and cryptography.* Journal of the ACM (JACM), vol. 56, no. 6, pages 1–40. ACM, 2009. (Cité en page 42.)

[RGPD 2018] RGPD. *Règlement Général sur la Protection des Données.* `https://eur-lex.europa.eu/legal-content/FR/TXT/HTML/?uri=CELEX:32016R0679&from=FR`, 2018. [Online ; accessed January-2022]. (Cité en pages 16 et 18.)

[Ruj 2011] Sushmita Ruj, Amiya Nayak et Ivan Stojmenovic. *Improved access control mechanism in vehicular ad hoc networks.* Dans Ad-hoc, Mobile, and Wireless Networks : 10th International Conference (ADHOC-NOW), pages 191–205. Springer, 2011. (Cité en page 32.)

[Sahai 2005] Amit Sahai et Brent Waters. *Fuzzy identity-based encryption.* Dans EUROCRYPT 2005 : 24th Annual International Conference on the Theory

and Applications of Cryptographic Techniques, pages 457–473. Springer, 2005. (Cité en page 23.)

[Sakai 2003] Ryuichi Sakai et Masao Kasahara. *ID based cryptosystems with pairing on elliptic curve*. Cryptology ePrint Archive. 2003. (Cité en page 34.)

[Sandhu 1998] Ravi S. Sandhu. *Role-based access control*. Dans Advances in computers, volume 46, pages 237–286. Elsevier, 1998. (Cité en page 74.)

[Scott 2005] Michael Scott. *Computing the Tate pairing*. Dans CT-RSA 2005 : The Cryptographers' Track at the RSA Conference, pages 293–304. Springer, 2005. (Cité en page 39.)

[Scott 2006] Michael Scott, Neil Costigan et Wesam Abdulwahab. *Implementing cryptographic pairings on smartcards*. Dans Cryptographic Hardware and Embedded Systems 2006 : 8th International Workshop (CHES), pages 134–147. Springer, 2006. (Cité en page 39.)

[Scott 2007] Michael Scott. *Implementing cryptographic pairings*. Lecture Notes in Computer Science, vol. 4575, page 177. Springer, 2007. (Cité en page 39.)

[Scott 2011] Michael Scott. *On the efficient implementation of pairing-based protocols*. Dans Cryptography and Coding : 13th IMA International Conference (IMACC), pages 296–308. Springer, 2011. (Cité en pages 39 et 118.)

[Shafieinejad 2021] Masoumeh Shafieinejad et Navid Nasr Esfahani. *A scalable post-quantum hash-based group signature*. Designs, Codes and Cryptography, vol. 89, pages 1061–1090. Springer, 2021. (Cité en page 43.)

[Shamir 1979] Adi Shamir. *How to share a secret*. Communications of the ACM, vol. 22, no. 11, pages 612–613. ACM, 1979. (Cité en page 117.)

[Stieghahn 2010] Michael Stieghahn et Thomas Engel. *Law-aware access control : about modeling context and transforming legislation*. Dans New Frontiers in Artificial Intelligence : JSAI-isAI 2009 Workshops, pages 73–86. Springer, 2010. (Cité en page 37.)

[Studnia 2015] Ivan Studnia. *Détection d'intrusion pour des réseaux embarqués automobiles : une approche orientée langage*. Thèse, Institut National des Sciences Appliquées de Toulouse, 2015. (Cité en pages ix, 8, 13 et 14.)

[Tan 2019] Syh-Yuan Tan, Kin-Woon Yeow et Seong Oun Hwang. *Enhancement of a lightweight attribute-based encryption scheme for the Internet of Things*. IEEE Internet of Things Journal, vol. 6, no. 4, pages 6384–6395. IEEE, 2019. (Cité en page 38.)

[TheGuardian 2018] TheGuardian. *Facebook says Cambridge Analytica may have gained 37m more users' data*. `https://www.theguardian.com/technology/2018/apr/04/facebook-cambridge-analytica-user-data-latest-more-than-thought`, 2018. [Online ; accessed January-2022]. (Cité en page 17.)

[TheNewYorkTimes 2018] TheNewYorkTimes. *How Trump Consultants Exploited the Facebook Data of Millions.* `https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html`, 2018. [Online ; accessed January-2022]. (Cité en page 17.)

[Torre 2019] Damiano Torre, Ghanem Soltana, Mehrdad Sabetzadeh, Lionel C Briand, Yuri Auffinger et Peter Goes. *Using models to enable compliance checking against the GDPR : an experience report.* Dans 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS), pages 1–11. IEEE, 2019. (Cité en page 36.)

[Torre 2021] Damiano Torre, Mauricio Alferez, Ghanem Soltana, Mehrdad Sabetzadeh et Lionel Briand. *Modeling data protection and privacy : application and experience with GDPR.* Software and Systems Modeling, vol. 20, pages 2071–2087. Springer, 2021. (Cité en page 36.)

[Tsabary 2019] Rotem Tsabary. *Fully secure attribute-based encryption for t-CNF from LWE.* Dans CRYPTO 2019 : 39th Annual International Cryptology Conference, pages 62–85. Springer, 2019. (Cité en pages 43 et 68.)

[Vaanchig 2020] Nyamsuren Vaanchig, Zhiguang Qin et Batjargal Ragchaasuren. *Constructing secure-channel free identity-based encryption with equality test for vehicle-data sharing in cloud computing.* Transactions on Emerging Telecommunications Technologies, vol. 33, no. 5, page e3896. Wiley Online Library, 2020. (Cité en pages 32 et 33.)

[Wang 2020] Geng Wang, Zhen Liu et Dawu Gu. *Ciphertext policy attribute-based encryption for circuits from lwe assumption.* Dans Information and Communications Security : 21st International Conference (ICICS), pages 378–396. Springer, 2020. (Cité en page 43.)

[Waters 2011] Brent Waters. *Ciphertext-policy attribute-based encryption : An expressive, efficient, and provably secure realization.* Dans PKC 2011 : 14th International Conference on Practice and Theory in Public Key Cryptography, pages 53–70. Springer, 2011. (Cité en page 23.)

[Xiong 2020] Hu Xiong, Yingzhe Hou, Xin Huang et Yanan Zhao. *Secure message classification services through identity-based signcryption with equality test towards the Internet of vehicles.* Vehicular Communications, vol. 26, page 100264. Elsevier, 2020. (Cité en page 32.)

[Yang 2022] Yang Yang, Jianguo Sun, Zechao Liu et YuQing Qiao. *Practical revocable and multi-authority CP-ABE scheme from RLWE for Cloud Computing.* Journal of Information Security and Applications, vol. 65, page 103108. Elsevier, 2022. (Cité en page 31.)

[Zhang 2012a] Jiang Zhang et Zhenfeng Zhang. *A ciphertext policy attribute-based encryption scheme without pairings.* Dans Information Security and Cryptology : 7th International Conference (Inscrypt), pages 324–340. Springer, 2012. (Cité en page 42.)

[Zhang 2012b] Jiang Zhang, Zhenfeng Zhang et Aijun Ge. *Ciphertext policy attribute-based encryption from lattices.* Dans 2012 ACM on Asia conference on Computer and Communications Security (AsiaCCS), pages 16–17, 2012. (Cité en pages 23 et 42.)

[Zhang 2022] Wei Zhang, Zhishuo Zhang, Hu Xiong et Zhiguang Qin. *PHAS-HEKR-CP-ABE : partially policy-hidden CP-ABE with highly efficient key revocation in cloud data sharing system.* Journal of Ambient Intelligence and Humanized Computing, vol. 13, no. 1, pages 613–627. Springer, 2022. (Cité en page 31.)

[Zhao 2019a] Yanan Zhao, Yingzhe Hou, Yanan Chen, Sachin Kumar et Fuhu Deng. *An efficient certificateless public key encryption with equality test toward Internet of Vehicles.* Transactions on Emerging Telecommunications Technologies, vol. 33, no. 5, page e3812. Wiley Online Library, 2019. (Cité en pages 32 et 33.)

[Zhao 2019b] Yang Zhao, Xing Zhang, Xin Xie, Yi Ding et Sachin Kumar. *A verifiable hidden policy CP-ABE with decryption testing scheme and its application in VANET.* Transactions on Emerging Telecommunications Technologies, vol. 33, no. 5, page e3785. Wiley Online Library, 2019. (Cité en page 32.)

Annexe A

Code ProVerif du protocole

A.1 Types

```
type s_key.
type attrs.
type accessp.
type abe_mkey.
type abe_skey.
type abe_pkey.
type gs_mkey.
type gs_skey.
type gs_pkey.
```

A.2 Variables Globales

```
free c: channel.
free storage_attrs: attrs.
free storage_ap: accessp.
```

A.3 Événements

```
event vehicle_send(attrs, attrs, bitstring, bitstring).
event vehicle_read(attrs, bitstring, bitstring).
event vehicle_conj(accessp, attrs).
event vehicle_leak(attrs, abe_skey).
event storage_write(bitstring).
event storage_send(bitstring).
event storage_leak(abe_skey).
event stakeholder_read(accessp, bitstring, bitstring).
event stakeholder_conj(accessp, attrs).
event stakeholder_leak(accessp, abe_skey).
```

A.4 Représentation du chiffrement basé attributs

```
free abe_mk: abe_mkey [private].

fun abe_pkgen(abe_mkey): abe_pkey.
fun abe_skgen(accessp, abe_mkey): abe_skey.
fun abe_enc(bitstring, attrs, abe_pkey): bitstring.
fun abe_lock(accessp, attrs): bitstring [private].
fun ext_attrs(attrs, attrs): attrs.

reduc forall m: bitstring, mk: abe_mkey, at: attrs ;
  abe_attrs(abe_enc(m, at, abe_pkgen(mk))) = at.

reduc forall m: bitstring, mk: abe_mkey, at: attrs, sk: abe_skey,
    ap: accessp ;
  abe_dec(abe_enc(m, at, abe_pkgen(mk)),
          abe_skgen(ap, mk),
          abe_lock(ap, at)) = m.
```

A.5 Représentation du chiffrement symétrique

```
fun s_enc(bitstring, s_key): bitstring.

reduc forall m: bitstring, k: s_key ;
    s_dec(s_enc(m, k), k) = m.
```

A.6 Représentation de la signature de groupe

```
free gs_mk: gs_mkey [private].

fun gs_pkgen(gs_mkey): gs_pkey.
fun gs_skgen(attrs, gs_mkey): gs_skey.
fun gs_sign(bitstring, gs_skey): bitstring.

reduc forall m: bitstring, mk: gs_mkey, i: attrs ;
  gs_msg(gs_sign(m, gs_skgen(i, mk)), gs_pkgen(mk)) = m.
```

A.7 Tables Globales

```
table list_attrs(attrs).
table list_msg(bitstring).
table list_locks(accessp, attrs, bitstring).
table list_stakeholders_sk(accessp, abe_skey).
table list_vehicles_sk(accessp, attrs, abe_skey).
```

A.8 Processus de création des attributs

```
let create_list_attrs() =
  !(
     new a: attrs ;
     insert list_attrs(a)
  ).
```

A.9 Processus du véhicule

```
let handle_vehicle(
      va: attrs, vap: accessp,
      vehicle_abe_sk: abe_skey, vehicle_abe_pk: abe_pkey,
      vehicle_gs_sk: gs_skey,   vehicle_gs_pk: gs_pkey) =
  !(
    (* Send a store_request. *)
    new k: s_key ;
    new vehicle_nonce: bitstring ;
    let ct1 = abe_enc((s_key2bs(k), vehicle_nonce), storage_attrs,
    vehicle_abe_pk) in
    out(c, gs_sign(ct1, vehicle_gs_sk)) ;

    (* Read the nonce to use. *)
    in(c, response: bitstring) ;
    let (storage_nonce: bitstring, =vehicle_nonce) = s_dec(gs_msg(
    response, vehicle_gs_pk), k) in

    (* Choose an attribute set to cipher the message msg. *)
    get list_locks(=vap, at, l) in
    new msg: bitstring ;
    let ct2 = abe_enc(msg, at, vehicle_abe_pk) in
    let ct3 = gs_sign(s_enc((ct2, storage_nonce), k), vehicle_gs_sk
    ) in
    event vehicle_send(va, at, ct2, msg) ;
    out(c, ct3)
  ) | !(
    (* Read a message from the storage. *)
    in(c, ct1: bitstring) ;
```

```
26      let ct2 = gs_msg(ct1, vehicle_gs_pk) in
27
28      let at = abe_attrs(ct2) in
29      get list_locks(=vap, =at, l) in
30      let msg = abe_dec(ct2, vehicle_abe_sk, l) in
31      event vehicle_read(va, ct2, msg)
32    ).
```

A.10 Processus du centre de stockage

```
1  let handle_storage(
2        storage_abe_sk: abe_skey,
3        storage_gs_sk: gs_skey,
4        storage_gs_pk: gs_pkey) =
5    !(
6      (* Read a store_request. *)
7      in(c, store_request: bitstring) ;
8
9      get list_locks(=storage_ap, =storage_attrs, l) in
10     let (s_key2bs(k), vehicle_nonce: bitstring) = abe_dec(gs_msg(
       store_request, storage_gs_pk), storage_abe_sk, l) in
11
12     (* Generate and send the nonce. *)
13     new storage_nonce: bitstring ;
14     out(c, gs_sign(s_enc((storage_nonce, vehicle_nonce), k),
       storage_gs_sk)) ;
15
16     (* Read the data sent, check the nonce and store the message.
       *)
17     in(c, ct3: bitstring) ;
18     let (ct2: bitstring, =storage_nonce) = s_dec(gs_msg(ct3,
       storage_gs_pk), k) in
19     event storage_write(ct2) ;
20     insert list_msg(ct2)
21   ) | !(
22     (* Read and send a value. *)
23     get list_msg(ct1) in
24     let ct2 = gs_sign(ct1, storage_gs_sk) in
25     event storage_send(ct2) ;
26     out(c, ct2)
27   ).
```

A.11 Processus de la partie prenante

```
1  let handle_stakeholder(
2        gs_pk: gs_pkey) =
3    !(
4      in(c, ct1: bitstring) ;
```

```
5
6     (* We could have the sk in parameter. However, we procceed this
       way to make leak easier. *)
7     get list_stakeholders_sk(sap, stakeholder_a_sk) in
8     let ct2 = gs_msg(ct1, gs_pk) in
9
10    let at = abe_attrs(ct2) in
11    get list_locks(=sap, =at, l) in
12    let msg = abe_dec(ct2, stakeholder_a_sk, l) in
13
14    event stakeholder_read(sap, ct2, msg)
15  ).
```

A.12 Processus de déploiement des véhicules, du centre de stockage et des parties prenantes

```
1 let create_storage() =
2   let storage_abe_sk = abe_skgen(storage_ap, abe_mk) in
3   let storage_gs_sk = gs_skgen(storage_attrs, gs_mk) in
4
5   insert list_locks(storage_ap, storage_attrs, abe_lock(storage_ap,
       storage_attrs)) ;
6
7   phase 4 ;
8   handle_storage(storage_abe_sk, storage_gs_sk, gs_pkgen(gs_mk)).
9
10 let create_vehicles() =
11   !(
12     new va: attrs ;
13     new vap: accessp ;
14     let vehicle_abe_sk = abe_skgen(vap, abe_mk) in
15     insert list_vehicles_sk(vap, va, vehicle_abe_sk) ;
16     let vehicle_gs_sk = gs_skgen(va, gs_mk) in
17
18     ( phase 2 ;
19       !(
20         get list_attrs(a) in
21         let at = ext_attrs(a, va) in
22         event vehicle_conj(vap, a) ;
23         insert list_locks(vap, at, abe_lock(vap, at))
24       )
25     ) | (
26       phase 4 ;
27       handle_vehicle(va, vap, vehicle_abe_sk, abe_pkgen(abe_mk),
     vehicle_gs_sk, gs_pkgen(gs_mk))
28     )
29   ).
30
31 let create_stakeholders() =
32   !(
```

```
33      new sap: accessp ;
34      let stakeholder_abe_sk = abe_skgen(sap, abe_mk) in
35      insert list_stakeholders_sk(sap, stakeholder_abe_sk) ;
36
37      (
38        phase 2 ;
39        !(
40          get list_attrs(a) in
41          get list_vehicles_sk(vap_private, va_private, unused) in
42          let at = ext_attrs(a, va_private) in
43          event stakeholder_attrs(sap, a) ;
44          insert list_locks(sap, at, abe_lock(sap, at))
45        )
46      )
47    ) | (
48      phase 4 ;
49      handle_stakeholder(gs_pkgen(gs_mk))
50    ).
```

A.13 Processus de divulgation des clés des véhicules, du centre de stockage et des parties prenantes ainsi que des informations publiques

Les macros `<##ifdef ...>`, `<##else>` et `<##endif>` n'appartiennent pas au langage ProVerif, ces macros sont définis via un pré-processeur externe à ProVerif et sont utilisés pour permettre la divulgation à tour de rôle des clés de déchiffrement de chaque entité à l'attaquant.

```
1  let do_vehicle_leak() =
2  <##ifdef VEHICLE_LEAK>
3    get list_vehicles_sk(leak_vehicle_ap, leak_vehicle_attrs,
       leak_vehicle_abe_sk) in
4    event vehicle_leak(leak_vehicle_attrs, leak_vehicle_abe_sk) ;
5    out(c, leak_vehicle_abe_sk).
6  <##else>
7    0.
8  <##endif>
9
10 let do_storage_leak() =
11 <##ifdef STORAGE_LEAK>
12   let leak_storage_abe_sk = abe_skgen(storage_ap, abe_mk) in
13   event storage_leak(leak_storage_abe_sk) ;
14   out(c, leak_storage_abe_sk).
15 <##else>
16   0.
17 <##endif>
18
19 let do_stakeholder_leak() =
20 <##ifdef STAKEHOLDER_LEAK>
```

```
21    get list_stakeholders_sk(leak_stakeholder_ap,
        leak_stakeholder_abe_sk) in
22    event stakeholder_leak(leak_stakeholder_ap,
        leak_stakeholder_abe_sk) ;
23    out(c, leak_stakeholder_abe_sk).
24  <##else>
25    0.
26  <##endif>
27
28  let do_public_leak() =
29    out(c, storage_attrs) ;
30    out(c, storage_ap) ;
31    out(c, abe_pkgen(abe_mk)) ;
32    out(c, gs_pkgen(gs_mk)) ;
33    !(
34    get list_locks(ap, at, l) in
35      out(c, ap) ;
36    out(c, at) ;
37    out(c, l)
38    ) | !(
39      get list_attrs(a) in
40      out(c, a)
41    ) | !(
42      new ap: accessp ;
43      out(c, ap) ;
44      out(c, abe_skgen(ap, abe_mk))
45    ).
46
47  let leaks() =
48    phase 3 ;
49    ( do_vehicle_leak()      | do_storage_leak()
50    | do_stakeholder_leak() | do_public_leak() ).
```

A.14 Processus principal

```
1  process
2      create_list_attrs() | create_storage()
3    | create_vehicles()   | create_stakeholders()
4    | leaks()
```

Titre : Protection des données des véhicules connectés : une approche cryptographique reposant sur le chiffrement basé attributs

Résumé : Les véhicules d'aujourd'hui envoient régulièrement de nombreuses données à propos de leur environnement ou de leur état à un Cloud, chargé du traitement et du partage des données avec de nombreuses parties prenantes. Ce type de communication pose des problèmes de sécurité et de vie privée, notamment face aux nombreuses attaques que subissent les clouds pouvant entraîner une fuite des données en dehors du cloud, ou sur le besoin des utilisateurs d'obtenir plus de contrôle sur leurs données notamment depuis la mise en application du RGPD. Pour ce faire, le chiffrement semble être une bonne solution et notamment le chiffrement basé attributs (KP-ABE) qui permet d'instancier des règles de contrôle d'accès sous la forme d'attributs et d'arbres d'accès.

Pour faire face à ces défis, cette thèse propose 1) un protocole cryptographique, basé sur le chiffrement KP-ABE, permettant de garantir des propriétés de sécurité sur les messages émis par le véhicule contre un attaquant externe et des participants honnêtes mais curieux. Les propriétés du protocole sont vérifiées via l'outil de vérification automatique ProVerif. Une fuite des clés de déchiffrement est également considérée pour évaluer l'impact d'une telle fuite. La vérification des propriétés montre que l'ensemble des propriétés sont vérifiées, et qu'en cas de fuite l'attaquant n'obtient pas plus d'informations que les entités du système ; 2) une méthode permettant de générer le contrôle d'accès en se basant sur la législation, les contrats et le consentement du conducteur ; 3) un prototype permettant d'évaluer les performances du protocole, les résultats de l'évaluation montrent que le protocole est réalisable dans un scénario de véhicule connecté ; et 4) des optimisations du déchiffrement KP-ABE, afin de réduire le temps de calcul des parties prenantes. Ces optimisations reposent sur des pré-calculs au déchiffrement des couplages basés sur le résultat des exponentiations des éléments de la clé de déchiffrement avec les coefficients de reconstruction. Le gain apporté pour le schéma KP-ABE Large Universe va de 6.91% pour 1 attribut à 52.05% pour 64 attributs et pour le schéma KP-ABE Small Universe de 12.71% pour 1 attribut à 80.23% pour 64 attributs.

Mots clefs : Protection des données des véhicules connectés ; Preuve automatique des propriétés ; Chiffrement basé attributs ; Conforme à la législation ; Optimisation déchiffrement KP-ABE

Title: Data protection for connected vehicles: a cryptographic approach relying on attribute-based encryption

Abstract: Today's vehicles regularly send a lot of data about their state or environment to a cloud, which is responsible for processing and sharing the data with many stakeholders. This type of communication raises security and privacy issues, especially in the face of numerous attacks on clouds that can lead to data leakage outside the cloud, or on the need of users to get more control over their data especially since the GDPR enforcement. To that end, encryption seems to be a good solution and in particular attribute-based encryption (KP-ABE) which allows to instantiate access control rules in the form of attributes and access trees.

To address these challenges, this thesis proposes 1) a cryptographic protocol, based on KP-ABE encryption, to guarantee security properties on messages sent by the vehicle against an external attacker and honest but curious participants. The protocol properties are verified via the automatic verification tool ProVerif. A decryption key leak is also considered to assess the impact of such a leak. The properties verification shows that all the properties are verified, and that in case of a leak, the attacker does not obtain more information than the system entities ; 2) a method to generate the access control based on the legislation, driver contracts and driver consent ; 3) a prototype allowing to evaluate the performance of the protocol. The evaluation results show that the protocol is compliant with the ressources available in a vehicle and that it can actually be used in a connected vehicle scenario ; and 4) optimizations of the KP-ABE decryption, in order to reduce the computation time of the stakeholders. These optimizations rely on pre-computations at the decryption of the pairings based on the result of the exponentiations of the decryption key elements with the reconstruction coefficients. The gain for the KP-ABE Large Universe scheme ranges from 6.91% for 1 attribute to 52.05% for 64 attributes and for the KP-ABE Small Universe scheme from 12.71% for 1 attribute to 80.23% for 64 attributes.

Keywords: Data protection for connected vehicles ; Automatic proof of properties ; Attribute-Based Encryption ; Legislation compliant ; KP-ABE decryption optimization