Sécurisation adaptative des objets de l'IoT par méthodes logicielles (White box) et combinées (hardware et software)

Tanguy Godquin

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Sécurisation adaptative des objets de l'IoT par méthodes logicielles (White box) et combinées (hardware et software)

Présentée et soutenue par
Tanguy GODQUIN

Thèse soutenue publiquement le 20/11/2020 devant le jury composé de		
M. BENJAMIN NGUYEN	Professeur des universités, Institut National Sciences Appliquées	Rapporteur du jury
M. MARIO SUDHOLT	Professeur des universités, MINES NANTES	Rapporteur du jury
M. MORGAN BARBIER	Maître de conférences, Université Caen Normandie	Membre du jury
Mme SAMIA BOUZEFRANE	Professeur des universités, CNAM Paris	Membre du jury
M. JEAN-LUC GRIMAULT	Ingénieur de recherche, Orange Labs Caen	Membre du jury
Mme MARYLINE LAURENT	Professeur des universités, TELECOM SUDPARIS	Membre du jury
M. JEAN-MARIE LE BARS	Maître de conférences HDR, Université Caen Normandie	Directeur de thèse

Thèse dirigée par JEAN-MARIE LE BARS, Groupe de recherche en informatique, image, automatique et instrumentation

UNIVERSITÉ CAEN NORMANDIE

MIIS

GREYC

Remerciements

Cette thèse est un rêve d'enfance qui m'a longtemps semblé inatteignable, mais qui pourtant se réalise. Grâce au financement sous convention CIFRE, j'ai eu l'opportunité d'effectuer ma thèse entre le laboratoire GREYC et Orange Labs à Caen, ce qui m'a permis de me former à la recherche académique tout en restant dans le milieu industriel. Ces trois années ont été pour moi une expérience très enrichissante aussi bien sur le plan professionnel que personnel et j'aimerais profiter de ces quelques lignes pour remercier toutes les personnes qui y ont contribué.

Je souhaite dans un premier temps remercier mes directeurs de thèse Jean-Marie Le Bars et Morgan Barbier de m'avoir guidé, accompagné, encouragé, mais également supporté tout au long de cette thèse. Merci d'avoir été présent depuis le début de mon cursus universitaire et jusqu'à ce jour. Merci pour nos nombreux échanges, aussi bien autour d'une table de réunion que d'un café. J'ai beaucoup appris à vos côtés et vous en remercie sincèrement.

Je voudrais remercier l'ensemble des membres du jury. Merci aux rapporteurs, Benjamin Nguyen et Mario Südolt, pour le sérieux et le détail dont ils ont fait preuve dans leurs rapports. Merci également aux examinatrices, Maryline Laurent et Samia Bouzefrane, de m'avoir fait l'honneur de faire partie de mon jury de thèse.

Du côté d'Orange, j'aimerais remercier mes encadrants Jean-Luc Grimault et Chrystel Gaber de m'avoir permis d'effectuer cette thèse dans les meilleures conditions possible. Merci pour votre bienveillance, votre accompagnement dans les multiples démarches administratives, mais surtout pour nos nombreux échanges qui, pour certains, se sont même conclus par un brevet. Comme quoi, nous travaillons aussi pendant les pauses café ! Merci également à Adam Ouarou ainsi qu'à Jean-Philippe Wary, respectivement responsables de domaine et de projet de recherche, pour avoir financé cette thèse et pour m'avoir fait confiance. Je ne peux mentionner Orange sans également remercier les membres de l'équipe Trust Enablers and Secure Elements (TESE) et les collègues du midi pour leur accueil pendant ces trois années. Merci à Cyril Alix, manager de l'équipe, avec qui j'ai partagé le bureau, Sebastien Picant, Philippe Thorel, Mathieu Lecouvey, Stephane Mangon, Stephane Bandin, Olivier Gruson, ainsi que Pascal Peschet et Franck Grupeli, sans oublier bien sûr Jean-Christophe Druesne, responsable du département SESAME et Lyseline Youf qui fut ma manager jusqu'à fin 2019.

Je remercie également les membres de l'équipe SAFE (Security Architecture Forensic and biomEtrie), ancienne équipe monétique et biométrie du GREYC pour m'avoir offert un poste d'ingénieur en attendant d'avoir l'opportunité de faire une thèse, et pour m'avoir accueilli au cours de cette thèse. Merci également aux amis du labo, Mathieu, Denis, Ghali, Peter, Gwen et Amine pour ces très bons moments passés autour de cafés, ainsi que pour nos parties de cartes ou de jeux endiablées.

Un grand merci à mes parents et grands-parents pour m'avoir encouragé dans la poursuite de mes études. Merci également à ma belle-mère, Sylvie, pour tous les gâteaux, glaces et autres gourmandises, mais surtout pour m'avoir accueilli à bras ouverts dans la famille. Cela compte beaucoup pour moi.

Je souhaite aussi remercier mes vieux amis Jean-Alexis, Pia et Jacques. Merci pour tous ces excellents moments tous ensemble. Vous êtes les Sam de mon Frodon, les Turk de mon J.D. ;).

Finalement, je tiens particulièrement à remercier ma femme Alison à qui je dédie cette thèse et sans qui rien de cela n'aurait été possible. Merci de m'avoir suivi dans cette aventure et de m'avoir supporté tout au long de ces trois années. Depuis que nous sommes ensemble, 12 ans déjà, tu m'apportes une immense joie. Je ne sais pas quoi dire de plus que : MERCI !

Résumé

Depuis maintenant plusieurs années, nous assistons à l'essor de l'Internet des Objets (IdO ou IoT en anglais). Suite à de récentes attaques sur ces systèmes, les études ont démontré que la sécurité de ces appareils était majoritairement insuffisante. Afin de remédier à ce problème, nous devrions idéalement mettre en place des mécanismes de sécurité sur l'ensemble des périphériques IoT, cependant cette solution n'est pas toujours envisageable.

Une approche alternative, pour sécuriser ces systèmes, consiste à déployer des services de sécurité en bordure du réseau afin de rapprocher les mécanismes de sécurité au plus près des périphériques non sécurisés.

Le but de cette thèse est de constituer un framework de sécurisation adaptative des objets de l'IoT qui repose sur le positionnement de services de sécurité. Ce travail se décompose en trois contributions qui touchent chacune des aspects différents de notre approche.

La première contribution élabore une stratégie de déploiement de services de sécurité qui minimise leurs coûts de déploiement. Cette approche traduit nos contraintes de positionnement sous la forme d'un problème de graphes que nous proposons de résoudre à l'aide d'outils de théorie des graphes.

La seconde contribution formalise les problèmes de placement de services pour les modéliser sous la forme d'une ontologie. Cette dernière est alors utilisée pour résoudre ces problèmes et permettre de comparer leurs différentes solutions.

La troisième contribution se focalise sur les services de sécurité qui implémentent de la cryptographie *whitebox*. Dans cette contribution, nous proposons un mécanisme d'ancrage de ces implémentations sur un réseau IoT afin de prévenir les attaques par extraction de code ainsi que le vol du périphérique.

Finalement, nous proposons un framework de sécurisation adaptative des objets de l'IoT dans lequel nous positionnons l'ensemble des contributions réalisées pendant cette thèse.

Table des matières

I Introduction et état de l'art **1**

1 Introduction de la thèse et des différents problèmes abordés **3**
1.1 Motivations . . . 3
1.2 Objets connectés . . . 5
1.3 Internet des objets . . . 9
1.4 Contexte sécuritaire de l'IoT . . . 13
1.5 Problématique et problèmes traités dans la thèse . . . 17
1.6 Notre approche . . . 18
1.7 Organisation du document . . . 19

2 Sécurité de l'IoT : un point de vue général **21**
2.1 Architectures . . . 21
2.2 Vie privée dans l'IoT . . . 25
2.3 Challenges de la sécurité IoT . . . 27
2.4 Les attaques dans l'IoT . . . 34

II Contributions **43**

3 Placement de services de sécurité dans le fog **45**
3.1 Contexte du fog computing . . . 45
3.2 Placement de services dans le fog . . . 47
3.3 Notre modèle de sécurité . . . 53
3.4 Placement de services basé sur la centralité de graphes . . . 56
3.5 Expérimentations . . . 63
3.6 Conclusion . . . 77

4 Comparaison des solutions de placement de services pour la sécurité **79**
4.1 Introduction . . . 80
4.2 Formalisation d'un problème de placement . . . 82
4.3 Présentation des ontologies . . . 87
4.4 Méthodologie de comparaison . . . 95

4.5 Illustration sur un cas concret . . . 104
4.6 Conception d'un outil de Comparaison de Solutions de Placement de Services (CSPS) . . . 108
4.7 Conclusion et perspectives de poursuite . . . 112

5 Ancrage réseau de fonctions cryptographiques white-box 113
5.1 Introduction . . . 113
5.2 Présentation de la cryptographie white-box . . . 115
5.3 Méthodologie d'ancrage d'implémentation boîte blanche . . . 123
5.4 Exemple d'un ancrage . . . 132
5.5 Conclusion et améliorations . . . 140

III Framework et Conclusion 143

6 Framework et application 145
6.1 Présentation générale du framework de sécurisation adaptative . . . 145
6.2 Éléments constitutifs du framework . . . 146
6.3 Positionnement des contributions de la thèse vis-à-vis du framework . 149
6.4 Exemple d'utilisation du framework . . . 150
6.5 Gestion de la mobilité et de la dynamicité des objets IoT . . . 151

7 Conclusion & perspectives 153
7.1 Travaux et contributions . . . 153
7.2 Perspectives de travail . . . 155

Publications de l'auteur 157

Bibliographie 159

A Les standards et protocoles de communication dans l'IoT 179
A.1 Internet conventionnel . . . 180
A.2 Sans-contact . . . 182
A.3 LR-WAN . . . 184
A.4 Comparatif et sélection . . . 192

B Représentations logique des classes et propriétés de l'ontologie FSPlacementOntology 195

C Cardinal d'une différence symétrique : une distance - démonstration 199

Table des figures 201

Liste des tableaux 204

Première partie
Introduction et état de l'art

Dans cette partie, nous introduisons le contexte de la thèse, ses problématiques et nos motivations. Cette partie comprend également une étude générale sur les objets de l'IoT, leurs architectures, et se focalise principalement sur les différents aspects de la sécurité dans l'IoT.

Chapitre 1

Introduction de la thèse et des différents problèmes abordés

Dans ce chapitre nous introduisons le sujet de thèse. Pour ce faire, nous commençons par définir les termes "objets connectés" et "Internet of Things (IoT)" afin d'établir un vocabulaire unifié et nécessaire à la lecture du document. Par la suite, nous présentons le contexte général de la sécurité de l'IoT. Finalement nous introduisons la problématique de la thèse ainsi que les différents problèmes que nous essayons de résoudre au travers de cette thèse.

Sommaire

1.1 Motivations 3
1.2 Objets connectés 5
1.3 Internet des objets 9
1.4 Contexte sécuritaire de l'IoT 13
1.5 Problématique et problèmes traités dans la thèse 17
1.6 Notre approche 18
1.7 Organisation du document 19

1.1 Motivations

Depuis plusieurs années, les objets connectés sont omniprésents. Que ce soit auprès des usines avec l'industrie 4.0 [247] ou dans les foyers avec les assistants vocaux (ou autres appareils intelligents), les objets connectés jouent un rôle prépondérant dans nos habitudes de vie. Les services que nous utilisons reposent, sans que nous ne nous en rendions compte, sur une multitude d'objets et d'applications de l'Internet

des objets (ou Internet of Things (IoT)). Le simple fait de consulter la météo le matin implique déjà l'utilisation de 3 entités : un client, souvent représenté par un smartphone, un ordinateur ou encore un assistant vocal, un réseau de capteurs météorologiques qui permet d'acquérir des données sur les conditions météorologiques, ainsi qu'un serveur qui regroupe ces données, les interprète et répond au client.

L'intérêt des entreprises envers l'Internet des Objets ne cesse de croître. Statista [5] prévoit une croissance du marché mondial de l'Internet des Objets avec un taux de croissance annuel d'environ 20%. Ces estimations sont confortées par celles de GrowthEnabler [9] qui estime ce taux à 28,5%. Il est intéressant de voir que les valeurs récoltées et les estimations de ces deux entreprises ne sont pas semblables, mais tendent vers un taux de croissance annuel relativement similaire. En l'absence d'informations sur les données récoltées par ces entreprises, il est envisageable de considérer qu'un choix différent a été effectué sur l'intégration de certaines données dans leurs analyses. La forte croissance du marché de l'Internet des Objets est fortement liée à celle du nombre de périphériques. Les capacités de gestion et d'optimisation des ressources font de l'Internet des Objets une technologie avec de forts enjeux sociétaux. Il est, par exemple, possible d'optimiser la consommation énergétique telle que le présente Mohsenian-Rad et al. [193] en utilisant les *Smart Grids*, d'économiser la consommation d'eau et d'augmenter la productivité des cultures agricoles [132] ou encore de limiter la congestion dans le trafic routier [190].

Les objets connectés, éléments constitutifs de l'Internet des Objets sont très variés de par leurs fonctions, leurs compositions, mais également par leurs protocoles de communication. Ils sont de surcroît déployés dans des environnements défavorables pouvant aboutir à des risques accrus de compromission. L'augmentation des interactions avec leur environnement et leurs emplacements géographiques (dans les foyers ou l'espace public par exemple) en sont les facteurs principaux. En raison de ces risques, il devient primordial de pouvoir assurer le bon fonctionnement de ces appareils tout en préservant leur sécurité. Ce défi est d'autant plus complexe qu'il existe une forte diversité d'objets, de protocoles de communications, mais également d'acteurs, dans l'écosystème IoT. Pour garantir la sécurité de ces objets, tout en respectant les principes d'auto configuration évoqués par Minerva et al. [188], il est essentiel d'adopter une approche adaptative.

La sécurisation, d'un point de vue très général peut être apportée de deux manières. La première consiste à introduire des méthodes de sécurisation matérielles telles que l'ajout d'élément de sécurité physique. Il est communément admis que cette méthode permet d'obtenir une sécurité plus forte, mais requiert l'ajout de composants physiques dans les objets, ce qui peut être une opération coûteuse. La seconde manière consiste à déployer des solutions de sécurisation logicielles. Souvent considérées moins robustes, elles comportent l'avantage d'être déployées sans ajout de matériel, à distance, et offrent une plus grande flexibilité.

1.2 Objets connectés

1.2.1 Définitions

La définition d'un objet connecté, également appelé "smart-device" est sujet à controverse. Dans un premier temps, il est important de replacer le contexte d'utilisation. Communément, le terme objet connecté est utilisé dans un contexte d'application relative à l'Internet des Objets, également appelé Internet of Things en anglais. Il y a alors une association entre les termes "objets" (donc objet connecté) et "things". Oxford définit [10] un objet comme étant une *chose* matérielle qui peut être vue et touchée. D'après un document de l'IEEE [188], "things" fait référence à tout objet physique qui est pertinent pour un utilisateur ou une application.

La communauté scientifique propose différentes définitions, Dorsemaine et al. [98], définissent les objets connectés de la manière suivante :

> "Les objets connectés sont des capteurs et/ou actionneurs exerçant une fonction spécifique avec des capacités de communication vers d'autres équipements. Ils appartiennent à une infrastructure permettant le transport, le stockage, le traitement et l'accès aux données générées par les utilisateurs ou d'autres systèmes" (traduit de [98]).

Zhang et al. [274] proposent une définition similaire en affirmant que les objets connectés sont des

> "objets physiques ou virtuels qui se connectent à l'internet et qui ont la capacité de communiquer avec des utilisateurs humains ou d'autres objets" (traduit de [274]).

Dans leur définition, les auteurs introduisent la possibilité qu'un objet connecté soit virtuel. Cela impliquerait alors que ce qui peut être considéré comme un seul objet physique soit finalement constitué d'une association de plusieurs objets virtuels. Le smartphone serait alors le parfait exemple d'une telle structure avec un unique appareil physique qui regroupe une multitude d'objets virtuels via ses nombreux capteurs.

La définition de Gorin [123] corrobore les définitions précédentes en considérant qu'un objet connecté

> "dispose d'une interface de communication standard (Wi-Fi, Ethernet, Zigbee, Bluetooth, etc.) et de capacités de traitement de données (capteur de température, humidité, etc.)" [123].

Ces trois définitions d'objet connecté définissent toutes le fait qu'un objet connecté soit constitué de capteurs et/ou d'actionneur et soit en mesure de transmettre de l'information à d'autres entités (utilisateurs ou machines).

Dans la suite de ce document, nous définirons donc un objet connecté comme un objet physique ou virtuel doté de capteurs et/ou d'actionneurs, possédant des capacités de traitement de données et capable d'accéder à internet depuis une ou plusieurs interfaces de communication (directement ou par l'intermédiaire d'une passerelle).

1.2.2 Capteur

Un capteur permet de mesurer une quantité physique et de la convertir en un signal lisible et analysable pour des actions ultérieures. Il ne faut pas confondre un capteur avec un transducteur dont le rôle consiste uniquement à convertir un signal. Les capteurs sont composés à minima d'un transducteur.

Il existe une forte diversité de capteurs qui exploitent tous des principes physiques permettant de déterminer diverses quantités. Les capteurs traduisent ainsi des données relatives au monde physique vers le monde digital tel qu'illustré en Figure 1.1.

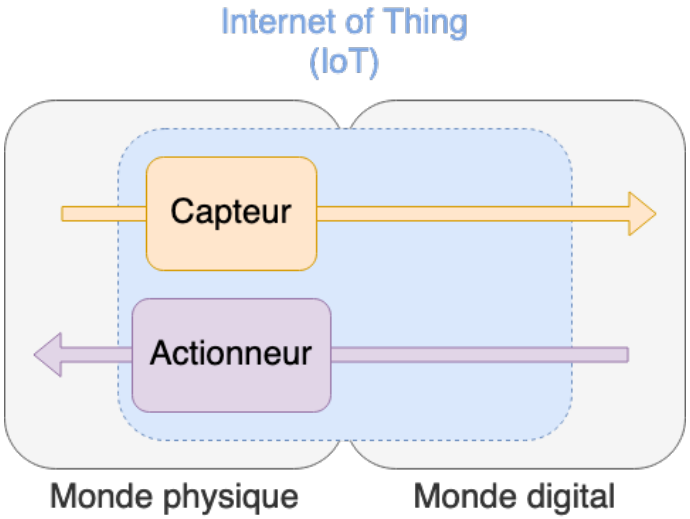


FIGURE 1.1 – Le rôle d'interface de l'IoT

Les capteurs se regroupent au sein de plusieurs catégories en fonction de leurs caractéristiques. Les plus communes correspondent à leurs sources d'énergie ainsi qu'à leur type de sortie.

1.2.2.1 Apport énergétique

Les capteurs ne nécessitent pas systématiquement de sources d'énergie extérieures, ils sont dans ce cas, appelés passifs. La variation de l'impédance, opposition totale à l'écoulement d'un courant électrique, lors de l'application d'une tension, sera alors analysée afin de mesurer la variation de la quantité physique.

À l'inverse, les capteurs actifs nécessitent la présence d'un signal énergétique externe afin d'agir comme des générateurs. Ces capteurs, une fois soumis à l'action du mesurande, convertissent la quantité physique en énergie électrique analysable par d'autres systèmes.

1.2.2.2 Type de sortie

Un capteur est dit analogique lorsque sa sortie est une grandeur électrique dépendante de la quantité physique mesurée et qu'elle peut prendre une infinité de valeurs continues dans un intervalle donné.

À l'inverse, un capteur est dit digital lorsque sa sortie est une séquence d'états logiques (0 et 1 par exemple) pouvant représenter une infinité de valeurs discrètes.

La majorité des microcontrôleurs possèdent des capacités d'interprétation de signaux analogiques. L'utilisation d'un capteur analogique sera préférée dans les cas nécessitant une précision importante ou un délai court pour la fourniture de l'information. Dans des cas moins restrictifs, les capteurs digitaux sont privilégiés puisqu'il est plus facile de travailler avec des signaux digitaux.

1.2.2.3 Exemple de capteur : le capteur thermique

La diversité des capteurs permet de mesurer une multitude de quantités physiques. Parmi les capteurs les plus utilisés dans le domaine de l'Internet des Objets, le capteur de température est un élément récurrent. Un tel capteur peut-être actif, passif, analogique ou encore digital comme abordé précédemment. Le processus de mesure de la quantité physique varie en fonction des phénomènes physiques utilisés. Dans le cas d'un capteur de température par thermistance, une résistance sensible aux variations de température permet de déduire le mesurande en analysant le changement de résistance du composant. Le rayonnement infrarouge d'un objet peut également être analysé par un capteur de température par infrarouge afin de déterminer la température de l'objet analysé.

1.2.3 Actionneur

Un actionneur est un dispositif qui prend en entrée un signal de commande et convertit une source d'énergie en phénomène physique (lumière, chaleur, mouvement, champ magnétique ...) (Figure 1.2). Cette conversion s'effectue par la présence d'un transducteur au sein de l'actionneur.

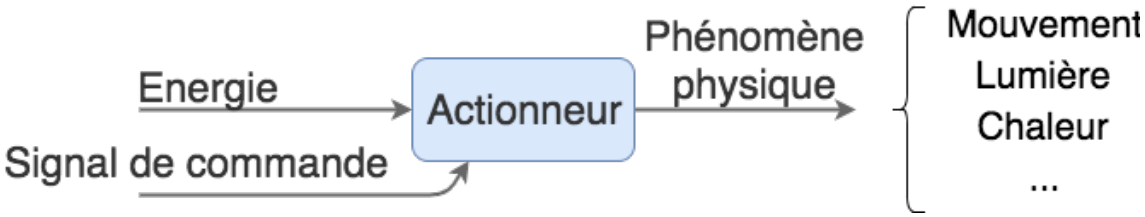


FIGURE 1.2 – Principe de l'actionneur

1.2.3.1 Apport énergétique

Les actionneurs peuvent être alimentés par de multiples sources d'énergie, qu'elles soient hydrauliques, pneumatiques, magnétiques ou encore électriques. Un transducteur convertit alors cette source d'énergie en phénomène physique. Contrairement aux capteurs, les actionneurs permettent au monde digital d'interagir sur le monde physique (Figure 1.1).

1.2.3.2 Exemples d'actionneurs

Il existe une grande diversité d'actionneurs permettant la production de phénomènes physiques variés. L'exemple de la bouilloire regroupe deux de ces phénomènes : la génération de chaleur par une résistance chauffante et une signalisation lumineuse au travers de multiples diodes électroluminescentes. Les phénomènes de déplacement sont produits par d'autres types d'actionneurs comme les vérins (déplacement linéaire uniquement) ou les moteurs.

1.2.4 Exemple d'objets connectés

Parmi les objets connectés les plus populaires dans les foyers, on distingue le système d'éclairage Philips Hue [1] qui offre la possibilité de configurer les paramètres d'éclairage (couleur, intensité ...) et d'interagir avec son environnement en communiquant avec d'autres objets connectés. Ce système communique via Zigbee en utilisant un profil spécifique pour l'éclairage appelé ZLL (*Zigbee Light Link*) lui permettant d'interagir avec d'autres périphériques Zigbee et internet par sa passerelle [204].

Le thermostat Nest [2] est également un objet connecté présent dans certains foyers. Ce thermostat permet d'analyser les habitudes de consommation énergétique d'un foyer et d'apprendre les comportements de ses habitants afin d'optimiser les dépenses énergétiques. Pour ce faire, il comporte deux interfaces de communications, Zigbee et Wi-Fi, lui permettant de communiquer et d'analyser les consommations de l'ensemble des appareils du foyer. Après quelques jours de paramétrage manuel du thermostat, ce dernier sera en mesure de s'auto configurer en fonction de l'activité du foyer (relevé par ses capteurs) et pourra réduire le chauffage lorsque les habitants sont absents.

Les objets connectés ne se limitent pas uniquement aux environnements contraints, les *smartphones* peuvent également être considérés comme des objets connectés grâce à leurs multitudes d'interfaces de communications, leurs capacités de traitements des données et la présence de capteurs (luminosité, caméra...) et actionneurs (enceintes...).

Certains objets ne remplissent pas les conditions pour être considérés comme connectés d'après notre définition. C'est le cas des étiquettes RFID 1 bit (ou transpondeurs 1 bit) voire de la majorité des étiquettes RFID passives. Ces objets n'effectuent pas de traitement de la donnée ; ils se contentent d'agir comme des code-barres radio et ne peuvent ainsi pas être considérés comme des objets connectés malgré la présence d'interface de communication. Il en est de même, pour les télécommandes infrarouges, les télécommandes de garages ou encore les balances numériques, car elle ne comprennent ni de traitement des données ni d'interface de communication vers Internet. Bien que les objets précédents ne soient pas considérés comme des objets connectés, ils peuvent cependant faire partie d'applications IoT qui peuvent interagir avec.

1. `https://www2.meethue.com/fr-fr`
2. `https://nest.com/fr/thermostats/nest-learning-thermostat/overview/`

1.3 Internet des objets

1.3.1 Définition

L'Internet des Objets (IdO) est une traduction française du terme Internet of Things (IoT) introduit en 1999 par Ashton [51]. Pour plus de cohérence, nous utiliserons, dans cette thèse, le terme Internet of Things et son acronyme associé IoT en référence au terme Internet des Objets.

En fonction des domaines dans lesquels le terme IoT est utilisé, sa définition varie. Haller et al. [134] précisent que l'essentiel, dans un contexte d'entreprise, est de bénéficier d'une intégration transparente aux processus industriels. Les auteurs font abstraction de la technologie utilisée pour introduire les concepts clés de l'IoT avec la définition suivante :

> Un monde où les objets physiques sont entièrement intégrés dans le réseau d'information et où ils peuvent participer activement aux processus opérationnels. Des services permettent d'interagir avec les "smart-objects" via internet, d'interroger leur état ainsi que toute information leur étant associée, et ce, tout en considérant les questions de sécurité et de respect de la vie privée. (traduit de [134])

Gorin et Dorsemaine et al. proposent des définitions moins centrées sur le côté industriel. Gorin [123] définit qu'un objet connecté fait partie de l'IoT s'il est directement adressable et s'il peut accéder à un réseau public directement ou par une passerelle réseau. Dorsemaine et al. [98], quant à eux, définissent l'IoT comme un réseau d'objets connectés permettant leur gestion, l'accès à leurs données et avec des capacités de traitement des données. On remarque que ces deux définitions reposent sur la définition de ce qu'est un objet connecté.

Pour répondre au manque de formalisme dans la définition de ce qu'est l'IoT, l'IEEE a publié un document recensant l'ensemble des éléments clés de l'IoT ainsi que deux définitions [188]. D'après les auteurs, l'IoT est constitué d'objets ("things") bénéficiant d'une identité unique, d'un accès internet, d'une intelligence embarquée et de capteurs et/ou d'actionneurs. Un système IoT bénéficie de capacité d'interopérabilité, d'une configuration automatique et offre la possibilité d'être programmé. Les auteurs proposent ainsi deux définitions de l'IoT divisées selon la complexité de l'environnement. Ces dernières peuvent être synthétisées par :

> L'IoT est un réseau complexe, adaptatif et autoconfigurable qui interconnecte des objets connectés à internet via l'utilisation de protocoles de communication standardisés" (traduit de [188]).

Nous adopterons cette définition dans le reste du document.

1.3.2 Historique

Le terme *Internet of Things* (IoT) a fait son apparition en 1999 dans le titre d'une présentation de Kevin Ashton à Practer & Gamble. Jusqu'à cette date, les ordinateurs étaient dépendants de l'Homme pour collecter des informations sur le

monde physique. Ashton [51] met en évidence le besoin de bénéficier de systèmes automatiques de collecte d'informations. Les ordinateurs pourraient ainsi posséder une connaissance accrue des éléments du monde réel et être en mesure de prendre des décisions adaptées aux différentes situations avec comme objectifs par exemple : la réduction du gaspillage et des dépenses, la réparation d'éléments ou encore le suivi de fraîcheur des aliments. Le principe de connaissance, par un appareil informatique, de l'environnement qui l'entoure, s'appelle le "context awareness". Ce désir de connecter les objets du quotidien est illustré par la modification d'un distributeur de boissons Cola en 1982 par Mike Kazar, David Nichols, John Zsarnay et Ivor Durham ou encore la première application de l'Internet des Objets en 1991, le *Trojan Coffee Pot* permettant de consulter le volume de café restant dans le pichet d'une cafetière depuis internet via une caméra.

Suite à l'identification de ce besoin de connectivité, le nombre d'objets connectés s'est accru pour finalement dépasser celui des humains présents sur Terre entre les années 2008 et 2009. Cette date est considérée comme marquant la naissance de l'Internet des Objets d'après Cisco Internet Business Solutions Group (Cisco IBSG). En 2011, l'Internet des Objets est cité et introduit par l'entreprise Gartner dans leur courbe annuelle des technologies émergentes illustrée en Figure 1.3. La technologie est par la suite régulièrement mentionnée et finit par subir un effet de mode impliquant une utilisation massive, souvent maladroite, de l'acronyme "IoT". Après coup, de multiples acteurs du monde informatique et économique prédisent une explosion du nombre d'objets connectés pour atteindre, d'ici 2020, 25 milliards d'objets pour Gartner [117], 50 milliards pour Cisco [102], ou encore 200 milliards pour IBM [41]. L'entreprise Statista [6] prédit 30 milliards d'objets connectés pour 2020 et jusqu'à plus de 75 milliards d'ici 2025. La grande variation du nombre d'objets estimés ou relevés est principalement due aux différentes définitions d'objets connectés.

1.3.3 Impact de l'IoT

La forte croissance du nombre d'objets connectés permet à la technologie d'étendre ses domaines d'application jusqu'à s'étendre aux trois grands secteurs économiques (primaire, secondaire et tertiaire). Beecham Research illustre ces propos en proposant une infographie (Figure 1.4) montrant la forte diversité des domaines d'application de l'Internet des Objets et détaillant les secteurs d'activités concernés par l'IoT 2017.

Secteur primaire

Le secteur primaire est fortement impacté par l'introduction de l'Internet des Objets. MarketsandMarkets prévoit une croissance de 100% du marché de l'agriculture intelligente (*smart agriculture* ou *smart-farming*) entre 2016 et 2022 (11,23 milliards USD pour 2022 contre 5,16 milliards USD en 2016) [20]. Ces prédictions sont appuyées par l'augmentation des projets d'agriculture intelligente tels que [216, 224]. L'agriculture, représentative du secteur primaire, n'est pas la seule activité à être impactée par l'Internet des Objets. Le marché de l'exploitation minière intelligente

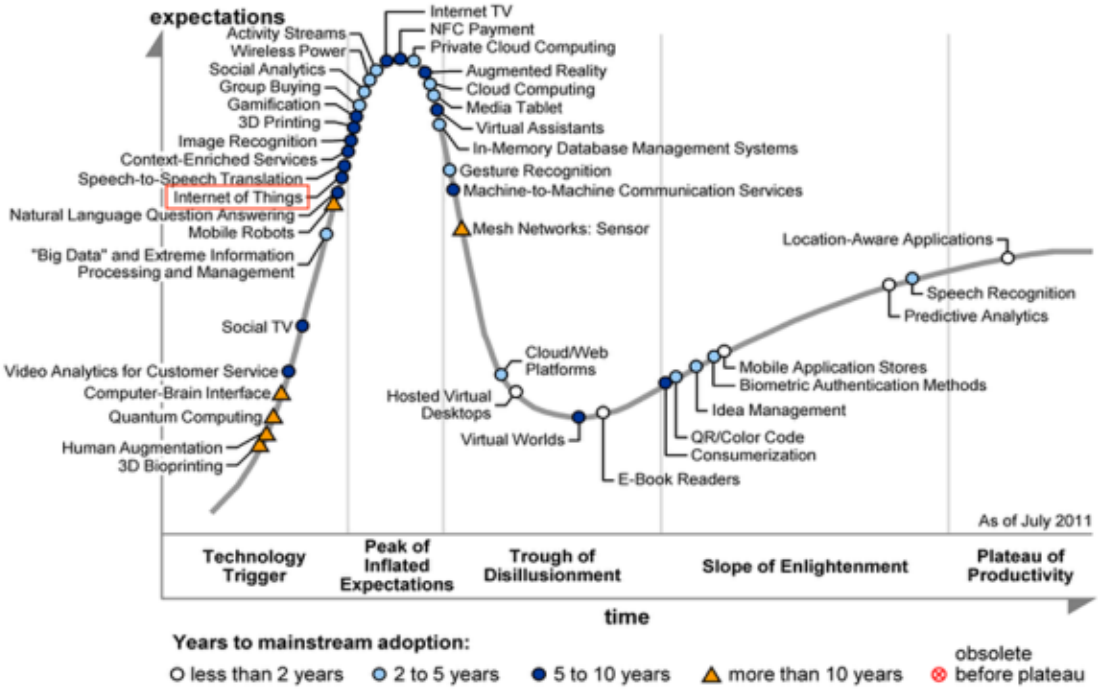


FIGURE 1.3 – Courbe des technologies émergentes, 2011 (source : Gartner 2011)

(*smart-mining*) subit également une forte croissance avec un Taux de Croissance Annuel Composé (TCAC) estimé à 14,9% durant la période 2016-2022 [21].

Secteur secondaire

Depuis quelques années (2011-2013), le secteur secondaire subit une révolution avec l'arrivée de l'*Industrie 4.0* [247] en partie causée par l'introduction de l'Internet des Objets dans l'industrie. Les chaînes de production sont de plus en plus automatisées. Les Industrial Control Systems (ICS) permettent aux différentes machines industrielles de communiquer entre elles afin d'accroitre leur productivité. En plus de la productivité, l'IoT contribue à la réduction du gaspillage des ressources en optimisant leur consommation, offre un environnement de travail plus sûr pour le personnel et fournit de nombreuses statistiques essentielles au fonctionnement des usines.

Secteur tertiaire

L'Internet des Objets se propage également au secteur tertiaire qui voit en 2002 apparaître les premières technologies de *Cloud Computing* lancées par Amazon *Amazon Web Services* permettant aux entreprises de déporter leurs calculs et d'accéder à des capacités accrues. Le *Cloud computing* est un des accélérateurs de l'IoT. Quelques années après son apparition, se développent ainsi de nouveaux secteurs d'activités dont notamment :

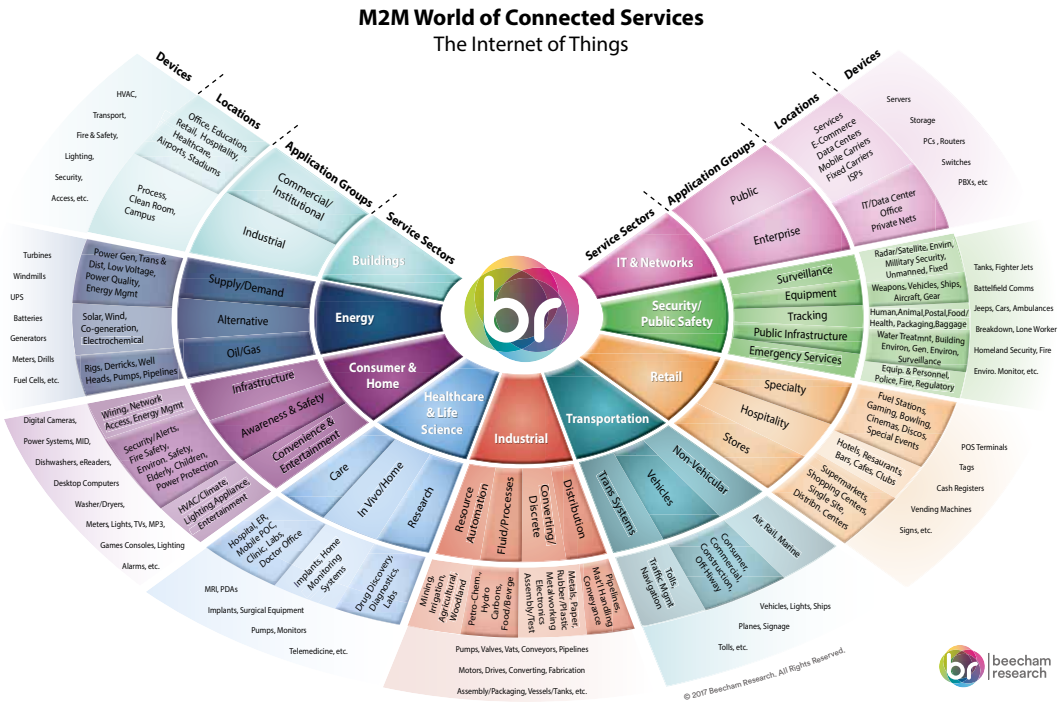


FIGURE 1.4 – Diversité des domaines d'application de l'Internet des Objets en 2017 (source : Beecham Research)

La santé En 2009, Carol Kasyjanski reçoit le premier pacemaker connecté (via Wi-Fi) des États-Unis. Cet appareil permet aux médecins de St Francis Hospital à Roslyn, New York, de suivre son état de santé à distance sans multiplier les déplacements à l'établissement médical.

Le domaine des transports Test de premières voitures autonomes par Google en 2009 sur les autoroutes californiennes.

La domotique Amazon commercialise, en 2014, les premiers assistants domotiques contrôlables par la voix (Amazon Echo).

1.3.4 IoT vs M2M vs IoE

Les acronymes IoT (*Internet of Things*), IoE (*Internet of Everything*) et M2M (*Machine to Machine*) sont utilisés dans de nombreux documents sans forcément bénéficier d'une définition ni d'une distinction établie.

Nous avons précédemment défini l'IoT (Internet des Objets), Section 1.3.1, comme un réseau complexe, adaptatif et autoconfigurable qui interconnecte des objets connectés à internet via l'utilisation de protocoles de communication standardisés.

Ce partage d'information est effectué via la communication entre les différentes entités du réseau. Le M2M correspond à la technologie qui permet à deux périphériques (*Machines*) de communiquer. Plus précisément, l'IEEE définit le M2M comme une

technologie qui permet aux dispositifs en réseau d'échanger de l'information et d'effectuer des actions sans intervention manuelle humaine [188].

Initialement cette technologie utilisait les réseaux satellites et cellulaires. Par la suite, la technologie a intégré les protocoles internet (*IP*) et sans-fil afin de communiquer avec un plus grand nombre d'appareils. Si l'on résume le M2M à la technologie qui permet à deux appareils de communiquer entre eux, on peut considérer le M2M comme sous-ensemble de l'IoT [36, 191]. Le M2M peut exister sans l'IoT alors que ce dernier en est dépendant.

L'IoE (*Internet of Everything*) est une notion introduite par Cisco [101] qui élargit le principe de connectivité présent dans l'IoT à l'ensemble des éléments existants. L'IoE comprend les *Things* de l'IoT constituant un sous-ensemble de l'IoE, mais également les données (*Data*), la population (*People*) et les processus de traitement (*Process*). La population est considérée comme des "nœuds du réseau" qui produisent des données statiques et dynamiques via leurs smartphones ou divers appareils connectés présents sur les personnes [36, 191]. En suivant la définition de l'IoE telle que faite par Cisco, on identifie que l'IoT correspond à un sous-ensemble de l'IoE.

La figure 1.5 reprend les différences entre ces termes sous la forme d'un schéma d'inclusion des éléments.

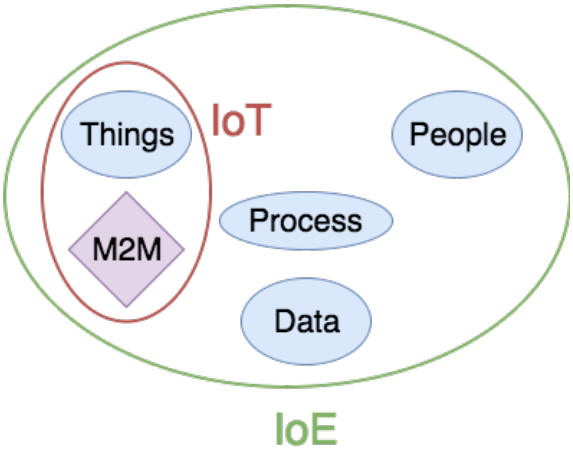


FIGURE 1.5 – Relations simplifiées entre M2M, IoT et IoE

1.4 Contexte sécuritaire de l'IoT

1.4.1 Un manque de sécurité flagrant

Les objets de l'IoT, en raison du nombre important d'appareils et de la variété des domaines impactée, nécessitent une sécurisation appropriée. Le cryptologue Bruce Schneier décrit en 2016 dans un article de blog que "le marché récompense encore largement le sacrifice de la sécurité au profit du prix et du délai de mise sur le marché" [235].

Cette constatation est confirmée par Barcena et Wueest [59] qui, dans leur étude sur 50 des objets les plus commercialisés, ont constaté qu'aucun des ces objets n'imposait de mots de passe suffisamment robustes, n'utilisait d'authentification mutuelle entre clients/serveurs ou ne bénéficiait de protection contre les attaques par force brute.

Les auteurs ont identifié que plusieurs interfaces cloud d'applications IoT n'appliquaient pas d'authentification double facteur. Les retours de missions de SERMA confirment ces propos en identifiant que près de 65% de ces interfaces n'étaient pas suffisamment sécurisés [103]. Une étude menée par HP sur les 10 objets les plus commercialisés appuie ces chiffres (60%).

Ces études ont également mis en évidence que la plupart des services IoT ne chiffrent ni ne signent les firmwares de mise à jour [59], 60% d'après HP [143] ou encore 57% d'après SERMA [103]. D'un point de vue plus général, il a été observé qu'une majorité des objets de l'IoT ne chiffrent pas leurs communications (82% d'après SERMA [103], 70% d'après HP [143]).

Cela n'est pas uniquement dû à l'aspect contraint des objets de l'IoT puisqu'il existe de plus en plus de solutions pour effectuer des opérations cryptographiques sur périphériques limités telles que la cryptographie à bas coût (dites également lightweight) ou encore l'ajout de crypto processeurs. Le problème vient principalement des concepteurs d'objets IoT. En effet, lors de leurs retours de missions, SERMA [103] a constaté que 90% des industriels n'avaient pas connaissance des menaces réelles de l'IoT, que 84% n'évaluaient jamais la sécurité de leurs produits et que près de 70% ne savent pas en évaluer leur robustesse. Quand bien même une évaluation de la sécurité serait demandée, SERMA [103] affirme que près de 90% des consultants en sécurité informatique ne bénéficient pas d'une formation suffisante à la sécurisation des objets de l'IoT.

L'ensemble de ces éléments font des objets de l'IoT des cibles de choix pour les attaquants.

1.4.2 Une nouvelle surface d'attaque

Manadhata et Wing définissent la surface d'attaque d'un système comme un sous-ensemble de ressources qu'un attaquant peut utiliser pour l'attaquer [180]. Les entrées et sorties d'un système, par exemple, constituent des surfaces d'attaque communément utilisées.

La prolifération des objets connectés au sein de multiples domaines d'application (illustrées en figure 1.4) a augmenté le nombre de ressources dont disposaient initialement les systèmes d'information. Ces ressources représentent de nouveaux vecteurs d'attaques sur les objets connectés, mais également sur des services plus critiques y étant reliés comme le montrent Stellios et al. dans leur étude [246].

Les nouvelles surfaces d'attaque introduites par l'IoT peuvent être exploitées afin de pénétrer dans une infrastructure informatique sécurisée et extraire de l'information tel qu'effectué lors de l'attaque d'un casino via un capteur présent dans un aquarium en 2017 [234]. La figure 1.6 illustre ces propos en représentant l'apparition d'un nouveau vecteur d'attaque causé par l'ajout d'une cafetière connectée, accessible

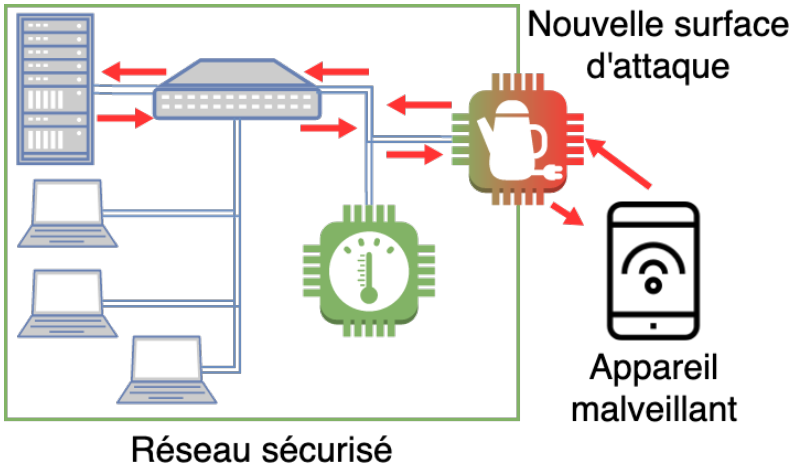


FIGURE 1.6 – Nouvelle surface d'attaque suite à l'ajout d'un objet connecté au sein d'une infrastructure sécurisée.

depuis une application mobile, sur une structure (à priori) sécurisée. Hossain et al. proposent, dans leur papier, un scénario dans lequel un seul objet compromis au sein d'un réseau interconnecté peut compromettre l'ensemble du réseau [142].

Les fonctionnalités des objets connectés peuvent également être exploitées afin de fournir un nouveau canal de communication. Ronen et Shamir introduisent en 2016 une attaque consistant à utiliser des fluctuations de luminosité d'une ampoule connectée afin d'exfiltrer des informations depuis une infrastructure fortement sécurisée [223].

Ces attaques sont appelées *IoT-enabled* car permises par l'ajout d'un élément IoT dans un système d'information. L'étude de Stellios et al. décrivent en détail cette catégorie d'attaques [246]. Les auteurs se focalisent sur les attaques réalisables, via preuves de concept ou exemples réels, tout en cherchant des méthodes pour s'en prémunir.

La difficulté réside dans l'identification de cette multitude de nouveaux vecteurs d'attaque. Miessler s'appuie sur les recommandations de l'OWASP IoT ([7]) afin de proposer un premier travail d'identification [186]. Ocvington et Carskadden proposent d'adapter la métrique de surface d'attaque définie par Manadhata et Wing dans la conception d'un *framework* d'analyse des implications de l'Internet des Objets [87, 179].

1.4.3 Quelques attaques dans l'IoT

À l'instar des Services d'Information (SI) classiques, l'Internet of Things n'est pas exempt d'attaques. L'OWASP propose un top 10 des pratiques à éviter lors de la conception d'applications IoT [12]. Ces pratiques vont des mots de passe faibles ou encodés en dur dans l'appareil au manque de mesures de sécurisations matérielles en passant par l'absence de mécanisme de mise à jour sécurisée. Quand une application IoT n'est pas conçue afin d'éviter ces pratiques, elle se retrouve susceptible de subir

une attaque.

Une approche plus systématique des attaques relatives à l'IoT est proposée en Section 2.4. Les paragraphes suivants introduisent quelques exemples d'attaques sur le monde IoT qui ont bénéficié d'une large couverture médiatique.

Barcena et Wueest décrivent dans leur papier deux attaques utilisant des méthodes d'attaques sur SI classiques [59]. Une première consiste à effectuer une injection SQL auprès de l'interface cloud d'une application connectée offrant ainsi la possibilité à l'utilisateur de déconnecter l'ensemble des objets du service ou de s'emparer des données d'identification. La seconde attaque décrite consiste à effectuer une attaque Man-In-The-Middle lors de la vérification des mises à jour d'un hub intelligent. Pour ce faire, les auteurs effectuent du ARP poisoning pour se faire passer pour le serveur de mise à jour de l'objet. Le paquet de mise à jour n'étant pas chiffré ni signé, les auteurs peuvent ainsi l'altérer et le fournir à l'appareil lors de sa requête de mise à jour.

Similairement, l'attaque de Target en 2014 repose sur une négligence dans l'élaboration du réseau interne de l'entreprise [159]. Dans cette attaque, le système permettant de contrôler le chauffage, la ventilation et la climatisation (HVAC) était directement relié avec le terminal bénéficiant d'un accès aux données bancaires des clients. L'attaquant a pénétré le réseau de Target après avoir subtilisé des identifiants d'accès à l'entreprise de maintenance du système HVAC.

Malgré le fort impact médiatique de l'attaque de Target, ce n'est rien en comparaison de celui du botnet Mirai [139, 50]. Ce botnet exploite une multitude de périphériques IoT dont la plus grande proportion correspond à des webcams avec des identifiants de connexions faibles ou par défaut. Les périphériques alors infectés agissent alors comme des zombies et suivent les ordres de l'attaquant en envoyant une multitude de requêtes auprès des serveurs d'OVH et du site web de KrebsOnSecurity [3].

D'autres attaques, plus représentatives de l'environnement IoT, exploitent la présence de capteurs et d'actionneurs au sein des objets. Ronen et Shamir réalisent une exfiltration de données dans une infrastructure hautement sécurisée par l'intermédiaire d'une lampe connectée et d'un capteur de luminosité [223]. Dans leur attaque, les auteurs font varier légèrement l'intensité lumineuse de manière à ce que cela ne soit pas perceptible à l'œil humain, mais que le capteur de luminosité puisse percevoir la variation. Ils créent ainsi un canal de communication par lequel ils peuvent exfiltrer de l'information.

L'ensemble des attaques décrites précédemment ont des effets principalement restreints au monde digital. À l'inverse, les attaques effectuées sur les Industrial Control Systems (ICS) ont des répercussions sur le monde physique. L'une des attaques les plus connues est celle du vers Stuxnet [163, 160] infectant les systèmes SCADA (Supervisory Control And Data Acquisition). Les systèmes SCADA sont principalement utilisés dans les industries et notamment dans les raffineries et grandes chaînes de production [74].

La plateforme Shodan [4] indexe l'ensemble des objets IoT accessibles publiquement depuis internet. Elle recense près de 55 164 objets de type ICS publiquement accessible

3. KrebsOnSecurity `https://krebsonsecurity.com`
4. Shodan `https://www.shodan.io`

depuis internet illustré en Figure 1.7. Pour ce type de machines, rien n'est plus important que la notion de disponibilité. Il arrive fréquemment que de nombreuses industries refusent d'effectuer des mises à jour de leurs appareils pour éviter une indisponibilité ponctuelle ou tout simplement pour des questions de compatibilité avec certaines suites logicielles. La plateforme Shodan, en plus de l'adresse de l'objet, fourni également des informations relatives aux services sur l'appareil tels que sa version de firmware, les ports ouverts, etc. Il est d'autant plus important de sécuriser ces appareils.

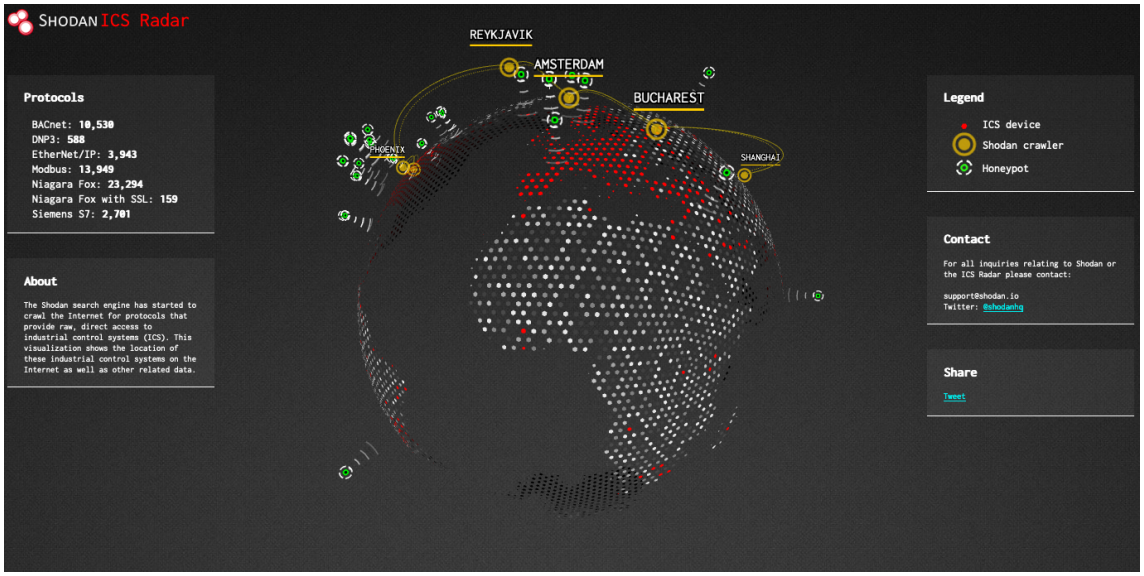


FIGURE 1.7 – Nombre d'objets ICS indexés par SHODAN (à la date du 1er avril 2020) : 55 164

1.5 Problématique et problèmes traités dans la thèse

Idéalement, afin de sécuriser correctement l'ensemble des objets d'un réseau IoT, il faudrait pouvoir déployer des mesures de sécurisation directement sur ces objets. Malheureusement, cette opération n'est pas toujours envisageable pour plusieurs raisons. La première est qu'une fois déployés, de nombreux objets ne sont plus accessibles ou ne peuvent que recevoir des mises à jour spécifiques de leur firmware. Dans d'autres cas, les acteurs des méthodes de sécurisation et les constructeurs ne sont pas identiques, ce qui contraint fortement la compatibilité entre les services de sécurité et les appareils à sécuriser restreignant alors fortement la capacité de déploiement.

En l'absence de déploiements possibles sur la totalité des objets, une alternative consiste à l'effectuer sur les appareils compatibles et essayer d'apporter la sécurisation le plus près possibles des objets qui ne sont pas sécurisés. Ce principe est communément appelé sécurisation en bordure de réseau. Si l'on représente un réseau sous la forme d'un arbre avec pour racine le cloud et les feuilles les objets connectés,

l'objectif est de déployer des mesures de sécurisation au plus près des feuilles. Ce principe de délocalisation des services vers les feuilles est l'essence même du "fog computing" [71].

Afin de proposer une méthode de sécurisation dans le contexte du "fog computing", nous devons nous intéresser à plusieurs problèmes.

Le premier réside dans l'identification des emplacements les plus appropriés pour le déploiement de services de sécurité. Le problème qui se pose est le suivant : soit un réseau IoT et un ensemble de services de sécurité, sur quels appareils doivent être déployés ces services pour offrir une sécurisation maximale du réseau avec un coût de déploiement minimal ?

Ce problème est défini dans la littérature sous plusieurs appellations, le Service Placement Problem (SPP) par Salaht et al. [229] ou encore le Fog Service Placement Problem (FSPP) par Skarlat et al. [244] et suscite fortement la communauté scientifique depuis quelques années. Malgré la forte augmentation de contributions dans ce domaine, les problèmes de placement qui y sont définis ne sont pas identiques, il est alors difficile de les comparer. Notre second problème consiste à trouver une modélisation des problèmes de placement de services IoT d'une manière générale afin de répondre à l'interrogation suivante : comment comparer les différents travaux sur le placement de services IoT de la littérature ?

Finalement, notre troisième préoccupation se focalise sur les services en eux même. Afin de nous assurer que ces derniers ne sont pas déplacés ou extraits par un attaquant nous devons répondre au problème suivant : comment ancrer un service à son emplacement dans un réseau ?

La réalisation de l'approche proposée dans cette thèse réside sur la résolution de ces trois problèmes.

1.6 Notre approche

Les travaux de cette thèse consistent à concevoir un framework capable d'analyser les besoins en sécurité d'objets de l'IoT afin d'adapter le niveau de sécurité à ces besoins et aux usages des objets. Ce framework est composé de deux étapes : la première se focalise sur l'analyse des besoins des objets alors que la seconde vise à les satisfaire en déployant des mesures de sécurité appropriées.

La première étape de conception de ce framework, l'analyse des besoins des objets, a abouti à la publication d'un premier article [120]. Dans ce document, nous apportons une première sensibilisation aux menaces liées à l'IoT en mettant en relation les fonctionnalités des objets avec les menaces qu'elles peuvent introduire. Cet article ne propose pas de solution technique, il apporte une prise de conscience sur la sécurité des objets et incite les constructeurs à réfléchir lors de l'ajout de fonctionnalité facultative à leurs produits aux risques d'avoir une sécurité moindre.

La seconde étape de conception du framework aborde le besoin de sécurité des objets. Dans notre approche, nous associons ce besoin au problème de placement de service dans une infrastructure fog (FSPP). Une première contribution repose sur l'élaboration d'une stratégie de déploiement de services de sécurité qui minimise leurs

coûts de déploiement. Notre réseau IoT est alors représenté sous la forme d'un graphe et nos contraintes de positionnement des services de sécurité sont traduites sous la forme de problèmes de graphe. Dans notre cas, ce problème de graphe correspond à l'identification d'un ensemble dominant favorisant la sélection des sommets avec un poids élevé.

En complément de cette étude, une seconde contribution généralise les problèmes de placement de services de sécurité dans l'IoT et propose une formalisation de ces problèmes à l'aide de logique Monadique Existentielle du Second Ordre (MESO). Cette formalisation est alors retranscrite dans une structure sémantique afin de constituer une ontologie. À l'aide des divers outils mis à disposition des structures ontologies, plus particulièrement les raisonneurs ainsi que leurs règles de déduction, nous construisons une base de connaissance qui va constituer le cœur de notre framework de sécurisation.

D'autre part, nous étudions les méthodes de sécurisation logicielles et introduisons les principes de la cryptographie *whitebox* comme moyen particulier de sécurisation. Au travers de cette contribution, nous étudions la cryptographie *whitebox* et ses limites. Nous découvrons ainsi qu'il existe deux catégories d'attaques auxquelles les implémentations de cryptographie *whitebox* sont vulnérables : les attaques qui extraient le code de l'implémentation (*code lifting*) et celles qui récupèrent la clé de chiffrement (*key recovering*) [109]. Dans cette troisième contribution nous proposons un mécanisme d'ancrage des implémentations de cryptographie *whitebox* qui permet non seulement de prévenir les attaques par extraction de code mais également de détecter lorsque l'objet qui héberge cette implémentation est déplacé de son environnement habituel (lorsque ce dernier est subtilisé par exemple).

Au final, nous décrivons un framework de sécurisation IoT qui permet d'analyser la sécurité des objets d'un réseau IoT et d'y déployer des services de sécurité pour adapter leurs niveaux de sécurité. Nous présentons les différents éléments constitutifs de ce framework et positionnons nos travaux vis-à-vis de ce dernier.

1.7 Organisation du document

Cette thèse est organisée en trois parties constituées de la manière suivante. La première partie comporte deux chapitres. Le premier, Chapitre 1, introduit le sujet de thèse, les motivations, et l'approche effectuée dans le cadre de la thèse. Le second, Chapitre 2, comprend une analyse générale sur la sécurité de l'IoT ainsi que quelques descriptifs techniques de l'IoT.

La seconde partie regroupe les contributions principales de la thèse. Le Chapitre 3 aborde le placement de services de sécurité dans une architecture fog en se basant sur des techniques de théorie des graphes. Un bref état de l'art des travaux en relation avec le problème de placement est fourni en introduction de ce chapitre. Le Chapitre 4 complète l'approche initiale du problème de placement par l'introduction de structures sémantiques reposant sur les ontologies. Ce chapitre est accompagné d'une introduction au domaine de la sémantique du web et des ontologies ainsi qu'une étude des travaux relatifs aux ontologies dans le domaine de l'IoT et du placement de services.

Le Chapitre 5 conclut les contributions principales en présentant la cryptographie *whitebox* et en introduisant un mécanisme d'ancrage local des implémentations *whitebox* dans le réseau.

La dernière partie comprend les deux derniers chapitres de cette thèse. Le Chapitre 6 présente un framework de sécurisation adaptative des objets de l'IoT, positionne les contributions de cette thèse par rapport à ce framework et illustre brièvement son utilisation. Le Chapitre 7 conclut cette thèse par la reprise de l'ensemble des contributions effectuées au cours de cette thèse et ouvre vers de nouvelles thématiques de recherche connexes au sujet de thèse pouvant donner suite à nos travaux.

Chapitre 2

Sécurité de l'IoT : un point de vue général

Afin d'étudier les moyens de sécurisation de l'IoT, il est nécessaire de connaitre les différentes structures associées au domaine de l'IoT, les éléments constitutifs de l'IoT et les manières dont ils interagissent entre eux. Ce chapitre décrit, dans un premier temps, l'architecture générale des applications IoT. Par la suite, la sécurité de l'IoT est étudiée au travers des menaces liées à l'IoT, aussi bien sur l'aspect données personnelles que sur la sécurité en général. Finalement, les challenges de sécurité de l'IoT sont décrits et complétés par une analyse des attaques sur l'IoT.

Sommaire

2.1 Architectures 21
2.2 Vie privée dans l'IoT 25
2.3 Challenges de la sécurité IoT 27
2.4 Les attaques dans l'IoT 34

2.1 Architectures

La démocratisation de l'Internet des Objets et plus particulièrement de ses domaines d'application ont donné naissance à de multiples structures. Afin de pouvoir sécuriser les applications IoT, il est nécessaire d'avoir connaissance de ces structures, de leurs compositions et de leurs interactions. Les sections suivantes décrivent les architectures les plus communes de l'IoT. Une mise en correspondance des couches de ces architectures est illustrée dans la Figure 2.1.

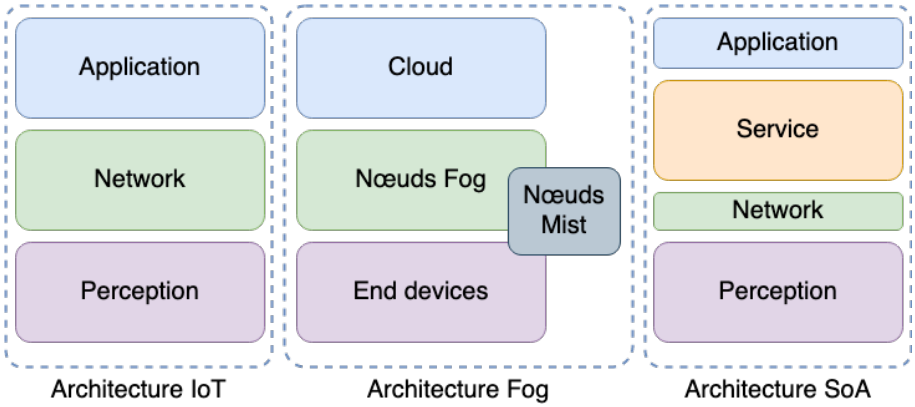


FIGURE 2.1 – Mise en correspondance des architectures IoT, Fog et SoA.

2.1.1 IoT architecture

L'Internet des Objets possède une architecture représentable sous la forme de couches. Ces couches sont communément définies d'après leurs fonctions et leurs éléments constitutifs. L'architecture la plus commune, illustrée en Figure 2.1, se compose de 3 couches nommées respectivement : Perception, Network et Application [176, 66, 275, 167, 169]

2.1.1.1 Perception

La couche *Perception*, également surnommée *sensor* [176] ou *hardware* [66] représente la base de l'architecture IoT. Elle regroupe l'ensemble des capteurs et actionneurs ainsi que la majorité des objets connectés. Son rôle est de mesurer et agir sur l'environnement physique afin de faire le lien entre les Things [188] et le réseau IoT [169].

2.1.1.2 Network

La couche *Network* constitue le cœur de l'Internet des Objets. Elle permet, par l'utilisation de divers protocoles de communications (Wi-Fi, Bluetooth, Zigbee, etc.), la transmission d'informations entre les éléments constitutifs de l'Internet des Objets [169]. Ces protocoles ne sont volontairement pas détaillés dans ce chapitre car leur description n'est pas utile à la compréhension des travaux de cette thèse. Néanmoins le lecteur trouvera en Annexe A une présentation de ces protocoles. Afin de permettre la transmission des informations, la couche *Network* comprend l'ensemble des appareils associés à cette opération tels que les routeurs, switchs, hubs, etc.

2.1.1.3 Application

La couche *Application* est la couche la plus visible de l'architecture IoT. Elle comprend l'ensemble des applications associées à l'Internet des Objets. Son rôle principal est d'agréger les données issues de la couche *Perception* et de les analyser afin de pouvoir fournir divers services.

2.1.2 Service oriented Architecture (SoA)

Avec le développement des applications IoT, un aspect modulaire a fait son apparition. Les composants des applications ont été décomposés en plusieurs sous-composants appelés services afin d'offrir une plus grande liberté dans le déploiement des applications. Cette pratique a donné naissance à l'architecture Service oriented Architecture (SoA) [89, 189].

Cette architecture reprend la structure IoT en 3 couches présentées à la sous-section 2.1.1 et introduit une quatrième couche intitulée *Service* entre *Network* et *Application* comme illustrée en Figure 2.1 [248, 169, 44].

Le principe de l'architecture SoA est de scinder les applications en une multitude de services qui peuvent être par la suite agencés de diverses manières afin de constituer de nouvelles applications ou pour réutiliser des services en commun [52].

De nombreuses autres architectures s'inspirent de SoA en définissant des sous-couches de *Services* ou en introduisant la couche *Business* au niveau de *Application*. Dans leur survey, Al-Fuqaha et al. [44] présentent quelques-unes de ces architectures. Il est notable que, quelque soit le nombre de couches ou l'architecture analysée, la structure générale reste une variation de l'architecture SoA et reste représentable sous la forme de 4 couches : *Perception*, *Network*, *Service* et *Application*.

2.1.3 Fog et Mist architectures

Les architectures Fog et Mist sont issues respectivement des paradigmes de Fog computing et de Mist computing. Ces paradigmes sont issus d'un concept plus large intitulé l'Edge computing, qui consiste à déplacer les services de traitement et d'analyse de données pour les rapprocher de leurs lieux de production et de consommation. En d'autres termes, l'Edge computing descend les fonctionnalités initialement présentes dans le cloud vers la bordure du réseau (end-devices) en rapprochant les couches *Service* et *Perception*.

Le Fog computing est une plateforme hautement virtualisée qui fournit des services traditionnellement présents dans le cloud computing (traitement de données, stockage, connectivité, etc.), généralement, mais pas exclusivement, en périphérie du réseau [71, 70].

Le principe du Mist computing accentue les principes du fog computing en visant à se rapprocher au plus près de la bordure de réseau. L'analogie météorologique de ces trois termes permet de se représenter leurs emplacements. Le cloud (nuage) est situé haut dans le ciel alors que le fog (brouillard) pourrait être représenté par un nuage de basse altitude. Le terme Mist (brume) correspond à un brouillard de faible intensité permettant une visibilité supérieure à un kilomètre. L'analogie considère ce phénomène comme du brouillard de très basse altitude.

En rapprochant les services de la bordure de réseau, ces principes offrent plusieurs avantages. Les avantages principaux consistent à réduire les temps de latence dans le traitement des données [233] tout en offrant une connaissance de l'environnement local. En effet, en réduisant le nombre d'intermédiaires dans la transmission des informations, les temps de réponse sont grandement améliorés. Avec la forte

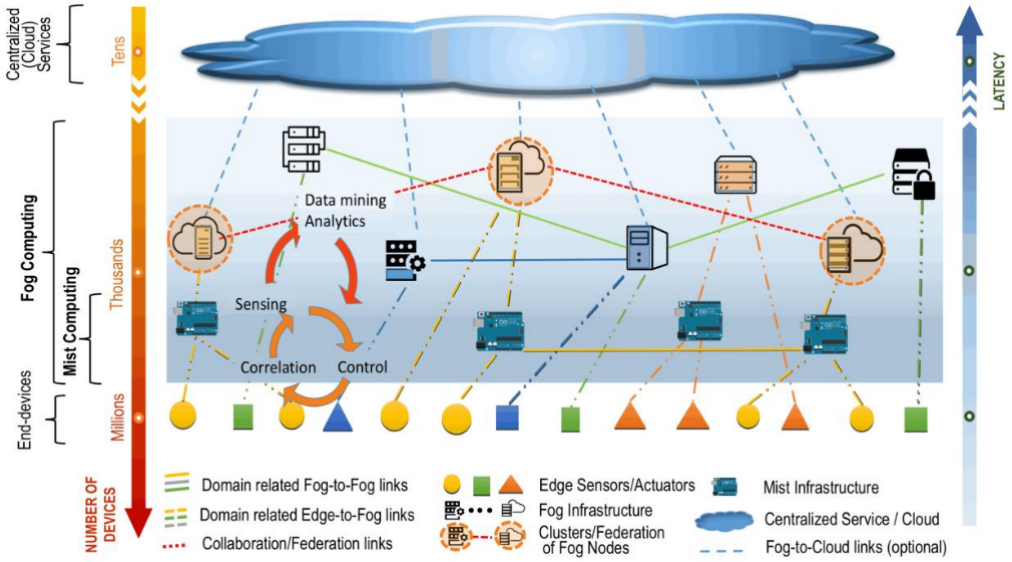


FIGURE 2.2 – Architecture de fog computing (source : [145])

augmentation de données générées par l'IoT, ces principes participent à réduire la quantité d'information transmise et offrent une meilleure montée en charge des services. Les applications peuvent également proposer des fonctionnalités plus adaptées aux besoins des utilisateurs en s'appuyant sur l'environnement [229].

La Figure 2.2 illustre ces paradigmes en présentant l'architecture fog constitué des couches : end-devices, fog-nodes et cloud.

2.1.3.1 End-devices

La couche des End-devices comprend l'ensemble des objets fortement contraints en termes de capacités de calculs. Elle est principalement constituée des capteurs, actionneurs, et divers objets tels que les caméras, thermostats, etc. Cette catégorie d'objets représente la majorité des éléments de l'IoT et offre une couverture géographique très vaste.

2.1.3.2 Fog-nodes

La couche des Fog-nodes est constituée d'éléments moyennement contraints en performances. Ces objets sont capables, grâce à leurs capacités de traitement des données, de stockage et de connaissance de leurs environnements, de fournir divers services en bordure du réseau. Cette couche peut regrouper, entre autres, les objets de type routeurs, points d'accès, serveurs, smartphones ou encore véhicules connectés. La séparation entre les couches End-devices et Fog-nodes n'est pas très distincte.

2.1.3.3 Cloud

La couche Cloud est composée des appareils qui bénéficient des meilleures capacités de traitement de données ou de stockage. Elle comprend la majorité des serveurs,

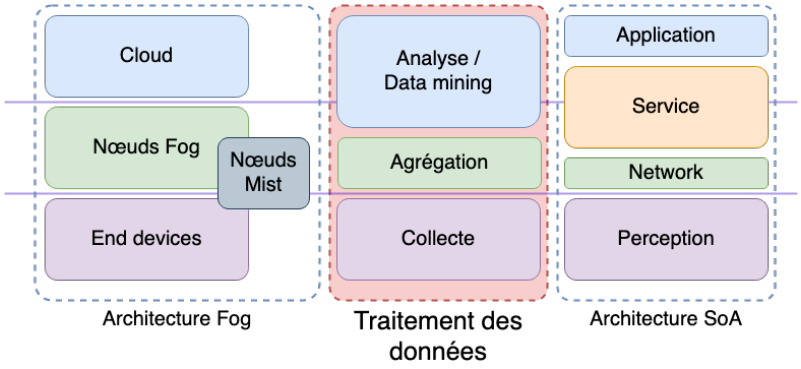


FIGURE 2.3 – Mise en relation des étapes de traitement des données avec les architectures fog et SoA.

calculateurs, et data centers. Son rôle est d'effectuer les opérations les plus complexes ou celles qui ne nécessitent pas de traitement en temps réel.

2.2 Vie privée dans l'IoT

La protection des données personnelles est une opération difficile dans le domaine de l'IoT. En effet, la quantité colossale d'informations couplée avec leurs natures variées et les besoins, parfois de traitement en temps réel, ne facilitent pas cette procédure.

En général, les données sont soumises aux 3 étapes de traitement suivantes : collecte, agrégation et analyse (ou data mining) [269]. La Figure 2.3 illustre ces étapes et les met en relation avec les architectures décrites précédemment en section 2.1.

L'étape de collecte, comme son nom l'indique, consiste à collecter les données sur l'état des objets de l'IoT. Elles peuvent avoir des natures variées et, par exemple, correspondre à des mesures de capteurs, des réponses de communications ou encore tout simplement à des requêtes en provenance des objets. Au cours de cette étape, de multiples protocoles de communication peuvent rentrer en compte. Nous décrivons, en Annexe A, quelques protocoles et standards de communications pour l'IoT parmi les plus communément utilisés.

Lors de l'étape d'agrégation, un ensemble de données similaires sont regroupées afin de construire une source d'information plus importante. Cette étape permet notamment de consolider les informations contextuelles que peuvent fournir différents objets IoT. Par exemple, dans le cas d'une application météorologique, plusieurs données de température issues d'objets proches géographiquement peuvent être regroupées pour accroitre la précision des informations. Dans notre exemple, ces informations peuvent être couplées avec des relevés hygrométriques toujours dans un objectif de consolider l'information résultante de cette opération.

La dernière étape de traitement des données, l'analyse ou data mining, consiste à extraire de nouvelles informations depuis les données agrégées à l'étape précédente. Ces informations ainsi extraites peuvent ainsi alimenter les diverses applications

IoT pour offrir des services variés. Si l'on reprend l'exemple d'une application météorologique, l'étape d'analyse permet, par exemple, de déconseiller aux utilisateurs de sortir de chez eux en cas de fortes températures si le taux d'humidité est également élevé.

2.2.1 Risques

La protection des données personnelles est plus importante dans le domaine de l'IoT que pour les services d'information traditionnels, car ces données peuvent être de nature très variées. La moindre information auparavant considérée anodine peut dorénavant avoir des conséquences dramatiques si elle n'est pas correctement protégée. Les capacités de collecte, d'agrégation et d'interprétation de l'IoT font que certaines informations, anodines individuellement, peuvent être critiques lorsqu'elles sont mises en relation. À titre d'exemple, la consommation électrique d'un foyer peut fournir de l'information sur les types d'appareils électroménagers présents dans le domicile. D'autre part, l'analyse de la consommation électrique en fonction des plages horaires peut fournir à un attaquant des informations sur la présence d'habitants dans un domicile. Hernandez et al. [138] illustrent cet exemple en décrivant une attaque sur le thermostat connecté Nest. Dans leur document, les auteurs accèdent aux informations de consommation électrique du foyer et peuvent utiliser l'objet comme point d'accès afin de compromettre d'autres périphériques du domicile.

2.2.2 Protections

Les sortes de protection de la fuite d'information lors du traitement des données peuvent être réparties en trois catégories : l'anonymisation, le chiffrement et la perturbation [269, 271]. La majorité de ces mesures de protection s'effectue lors des étapes d'agrégation et d'analyse des données sachant que ces dernières regroupent un grand nombre d'informations.

La première catégorie de protections repose sur l'anonymisation des données. L'idée derrière cette pratique est de ne pas être en mesure de déterminer l'identité associée à une donnée tout préservant son apport applicatif. Le principe de k-anonymisation (ou k-*anonymity*) par Sweeney [251] est une des procédures les plus communément utilisées. Cette procédure consiste à, depuis un ensemble de données non anonymes, constituer un nouvel ensemble de données de manière à ce qu'il soit impossible de déterminer une correspondance entre ces deux bases avec plus d'une chance sur k. Pour ce faire, les quasi identifiants (attributs ne permettant pas une identification à eux seuls, mais qui, une fois mise en relation permettent de constituer des identifiants uniques) sont identifiés et généralisés. Généraliser ainsi ces données permet ainsi d'atteindre une k-anonymisation c'est-à-dire que pour chaque n-uplet des données anonymes, il y a toujours au moins k n-uplets avec les mêmes attributs.

D'autres méthodes consistent à dissocier l'origine de la donnée de son information. Anonygator [212] s'inspire de ce principe, lors de l'étape d'agrégation, en séparant le message de son origine.

La seconde catégorie regroupe les procédures de chiffrement de la donnée. La plus grande partie de ces travaux repose sur l'utilisation de cryptographie homomorphe [192]. Le principe de la cryptographie homomorphe, introduite par Gentry en 2009 [118], est d'être en mesure d'effectuer des opérations directement sur des informations chiffrées. Cette pratique est particulièrement appréciée au niveau de l'étape de l'analyse des données ou du data mining en raison de la délégation de cette étape à des entités plus performantes. L'exemple couramment fourni pour illustrer le chiffrement homomorphique est celui du cloud. Dans cet exemple, un utilisateur protège ses données en les chiffrant et les dépose sur un serveur dans le cloud. Par la suite, il souhaite effectuer une requête sur ses données. Pour ce faire, l'utilisateur va chiffrer sa requête et la transmettre au serveur. Ce dernier applique alors la requête chiffrée sur les données de l'utilisateur, elles aussi chiffrées, et retourne le résultat à l'utilisateur. À réception du résultat de sa requête chiffrée, l'utilisateur est en mesure de la déchiffrer. L'intérêt de cette approche est qu'elle ne fournit aucune information sur les données stockées, sur la requête effectuée ni même sur le résultat retourné puisque ce dernier est également chiffré. Le chiffrement homomorphe n'est pas uniquement présent dans l'étape d'analyse. Certains travaux comme Girao et al. [119] introduisent cette notion dans l'étape d'agrégation. À noter cependant que l'utilisation de cryptographie homomorphe est très coûteuse [118].

La dernière catégorie comprend les mesures de perturbation de la donnée. Ces mesures sont particulièrement appréciées en raison du fait qu'elles soient effectuées directement sur la donnée brute [169]. Les perturbations peuvent être de diverses natures. Elles peuvent altérer légèrement la localisation de la donnée afin de masquer une position géographique [276], perturber les relations entre les données [78] ou encore ajouter de l'information [78]. Les méthodes de perturbations sont cependant fortement liées aux applications qui utilisent ces données. En effet, l'altération d'une donnée, aussi infime soit-elle, peut compromettre son intérêt auprès de certaines applications.

2.3 Challenges de la sécurité IoT

La sécurisation de l'Internet des Objets ne se résume pas à se prémunir contre la fuite de données sensibles. L'IoT, comme tout service d'information, requiert une protection considérant les trois propriétés fondamentales de la sécurité : confidentialité, intégrité et disponibilité. Dans leur étude, Yang et al. [272] relèvent les principaux problèmes de sécurité de l'IoT en les mettant en relation avec les 3 couches de l'architecture IoT. Le Tableau 2.1 reprend la synthèse de ces travaux effectuée par Mahamad Noor et Hassan [66]. Dans leur synthèse, les auteurs constatent que le nombre de challenges est plus important sur la couche *Perception* que sur les deux autres. Cela est en partie causé par un accès physique plus facile à ces appareils ou au fait que leur sécurisation est en moyenne moins importante que pour les objets des autres couches. U.Farooq et al. [254] proposent une approche plus détaillée concernant les mécanismes de sécurisations effectifs à chaque niveau de l'architecture IoT.

TABLE 2.1 – Mécanismes de sécurisation de l'architecture IoT d'après [272, 66, 254] .

Couches	**Challenges de sécurité (d'après [272, 66])**	**Mécanismes de sécurité (d'après [254])**
Perception	Détection anormale des capteurs d'un nœud Algorithmes cryptographiques et mécanismes de gestion des clés Anonymisation des données et de l'expéditeur Vulnérabilités des appareils	Authentification Confidentialité des données Confidentialité des informations sensibles Évaluation des risques
Network	Assurer les communications IPSec avec les nœuds IPv6	Authentification Confidentialité des données Sécurité du routage
Application	Systèmes informatiques embarqués configurables	Authentification Sécurité des données Détection d'intrusion Évaluation des risques

2.3.1 Méthodes de sécurisation

Le principe de l'Internet des Objets repose sur la capacité des objets à communiquer ensemble. Typiquement, les objets vont transmettre des informations aux serveurs et recevoir des informations de contrôle. Afin de garantir la sécurité de ces communications, une authentification mutuelle entre les entités est essentielle [66].

L'importance de l'authentification dans la sécurisation de l'IoT est confirmée notamment dans le Tableau 2.1 et plus précisément avec les travaux de U. Farooq et al. [254] où l'on observe que les mécanismes d'authentifications sont présents dans l'ensemble des couches constituant l'architecture IoT. L'étude de [66] appuie ces propos en démontrant l'intérêt de la communauté scientifique pour les mécanismes d'authentification. Dans leur étude, les auteurs ont analysé les publications relatives à la sécurité IoT entre 2016 et 2018 puis ont relevé que près de la moitié de ces travaux portaient sur des mécanismes d'authentification.

Ces mécanismes reposent sur trois types de protocoles d'authentification : les protocoles basés sur des systèmes cryptographiques symétriques, ceux basés sur des systèmes asymétriques et les protocoles hybrides [104, 66]. La Figure 2.4 détaille ces trois types en fournissant quelques exemples pour chaque catégorie.

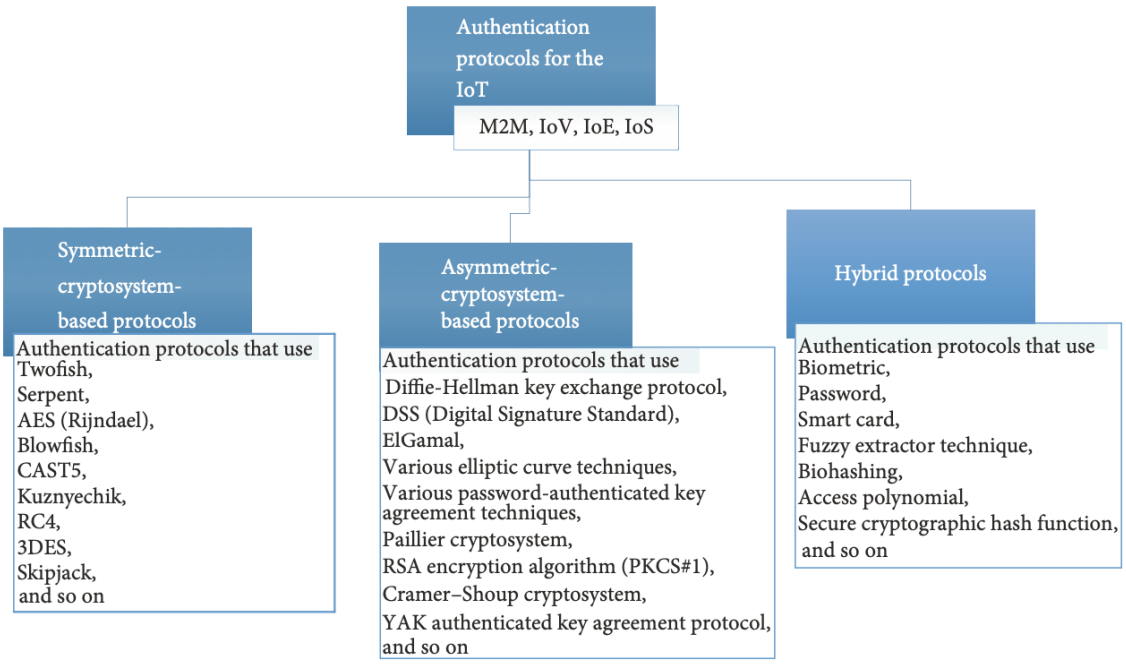


FIGURE 2.4 – Classification des protocoles d'authentification pour l'IoT d'après leurs cryptosystèmes (source : [104])

2.3.1.1 Cryptographie symétrique

La cryptographie symétrique est la forme de cryptographie la plus utilisée dans le domaine de l'Internet des Objets. Les principaux protocoles de l'IoT tels que Z-wave [108, 105], Zigbee [42], BLE [34], Sigfox [112, 217] ou encore LoRaWAN [217] utilisent tous de la cryptographie symétrique.

L'atout majeur de la cryptographie symétrique réside principalement dans sa rapidité d'exécution, mais également dans sa sécurité calculatoire et la capacité de ses algorithmes à être utilisés dans un environnement d'objets connectés. Étant donné que la sécurité calculatoire d'un algorithme de cryptographie symétrique est déterminée par la complexité des meilleures attaques sur ce dernier, il est nécessaire de laisser suffisamment de temps à la communauté pour son évaluation avant de l'utiliser. Le plus utilisé de ces algorithmes est AES (*Advanced Encryption Standard* [3]).

Lorsque les appareils possèdent des contraintes plus fortes sur les capacités, la cryptographie bas-coût ou *Lightweight* est à privilégier puisqu'elle utilise des primitives adaptées aux designs cryptographiques souhaités [196].

La cryptographie symétrique n'est malheureusement pas exempte de défauts ; sa sécurité repose en effet sur le secret de sa clé de chiffrement. Dans le domaine de

l'Internet des Objets, où les appareils sont potentiellement accessibles physiquement par des attaquants, il est difficile de ne pas divulguer cette clé. Plusieurs attaques sur l'implémentation des algorithmes ou sur les objets eux-mêmes existent et peuvent révéler des informations sur la clé de chiffrement utilisée. L'analyse temporelle [158], l'analyse différentielle de consommation (DPA [157]) sont des exemples d'attaques pouvant récupérer la clé de chiffrement stockée sur l'objet. Il est encore plus facile de s'en emparer lorsque cette dernière ne fait pas l'objet de mesures de protection et qu'elle se trouve enregistrée sous forme de fichier par exemple.

2.3.1.2 Cryptographie asymétrique

La cryptographie asymétrique (également appelée "cryptographie à clé publique") comporte deux clés, une publique et une privée. Pour les besoins de chiffrement, la clé publique peut être partagée et permet de chiffrer des données alors que la clé privée permet de déchiffrer les données chiffrées avec la clé publique. Ainsi, si Alice souhaite transmettre un message à Bob, elle va chiffrer son message avec la clé publique de Bob et ce dernier déchiffrera le message reçu avec sa clé privée. Pour des besoins de signature, clé publique et clé privée sont utilisées de la façon suivante : Alice signe son message avec sa clé privée et Bob vérifie la signature à l'aide de la clé publique de Alice. Dans le cas de RSA par exemple, la signature consiste en un chiffrement avec la clé privée d'un condensé du message.

La sécurité des systèmes de cryptographie asymétrique repose sur l'utilisation de fonction à sens unique de manière à ce qu'il soit facile de calculer la clé publique depuis la clé privée, mais que l'inverse soit calculatoirement impossible.

Les notions de confidentialité, intégrité, authentification et non-répudiation procurées par la cryptographie asymétrique résident dans l'authenticité de la clé publique. Afin de garantir ces notions, des infrastructures appelées PKI (*Public Key Infrastructure*) sont utilisées et permettent de valider l'authenticité des clés publiques ainsi que leur appartenance à la bonne entité.

Parmi les algorithmes de cryptographie asymétrique, RSA (Rivest Shamir et Adleman [220]) est probablement le plus populaire. Cependant, depuis maintenant plusieurs années, les algorithmes de la famille des ECC (*Elliptic Curve Cryptography* [156, 187]) sont de plus en plus recommandés. Malgré la sécurité apportée par ces algorithmes, ils ne peuvent être utilisés systématiquement dans l'Internet des Objets en raison de l'aspect contraint de ces appareils.

La cryptographie sur courbes elliptiques offre un énorme avantage dans le domaine de l'IoT en utilisant moins d'espace mémoire pour atteindre un même niveau de sécurité que RSA, plusieurs tentative pour adapter la cryptographie sur courbes elliptiques aux objets connectés ont été effectuées. Cependant, même si l'empreinte mémoire des algorithmes semble être plus adaptée, l'exécution ne se fait pas en temps raisonnable [153]. Une implémentation de cryptographie sur courbes elliptiques adaptées aux environnements contraints (TinyECC) a pu descendre le temps d'exécution à plusieurs dizaines de secondes, temps variant en fonction des plateformes de capteurs [171], mais cela reste incompatible avec de nombreux usages de l'IoT.

2.3.1.3 Protocoles d'authentification hybride

Depuis quelques années maintenant, de plus en plus de protocoles d'authentifications proposent une approche hybride avec notamment l'authentification double facteur. Dans leur étude, Mohamad Noor et Hassan [66] relèvent que 18% des publications relatives à l'authentification dans l'IoT entre 2016 et 2018 reposent sur l'utilisation de protocoles d'authentification multi facteurs. Ces protocoles peuvent induire l'utilisation de mots de passe [240], de données biométriques [93] ou d'éléments physiques comme des *smartcards* [48]. On peut alors supposer que cette tendance va se poursuivre dans les prochaines années avec, entre autres, l'émergence de produits commerciaux basés sur ces protocoles d'authentification hybride.

2.3.1.4 Utilisation d'éléments de sécurité hardware

Comme présenté au-dessus, certains protocoles de sécurisation s'appuient sur l'utilisation de composants matériels. Un énorme atout de l'utilisation de composants matériels (ou hardware) réside dans leur capacité à consolider les mesures de sécurité. Les vulnérabilités relatives au stockage et clés cryptographiques peuvent, par exemple, être réduites en effectuant les opérations cryptographiques directement à l'intérieur de *secure elements*. L'ajout de composants matériels (ou simulé au niveau logiciel) de sécurité permet de protéger un objet dans l'éventualité où celui-ci serait attaqué physiquement.

HCE

La technologie HCE (*Host Card Emulation*), introduite dans la version 4.4 (KitKat) d'Android, permet d'émuler une *smartcard* via l'interface NFC d'un smartphone. Les deux fonctionnalités principales d'une telle émulation consistent à recevoir un ensemble d'instructions ISO7816 et de répondre comme une réelle carte NFC. La technologie HCE est principalement utilisée dans les transactions sans-contact des smartphones. Le classement de HCE dans la catégorie des éléments de sécurité est discutable, car le stockage et l'utilisation des clés ne sont pas à proprement parler pris en charge par un élément de sécurité isolé du reste du terminal, mais par un ensemble de fonctionnalités hardware et software spécifique de la technique HCE. L'aspect principalement logiciel des technologies HCE dans le payement mobile entraîne des compromis dans la sécurisation [33]. Afin d'éviter de tels compromis, certains acteurs de la sécurité comme Thalès proposent de coupler cette technologie avec de réels éléments de sécurité matériels appelés HSM (Hardware security modules) [26] dans leurs solutions eSecurity [1]. Une autre modalité d'utilisation du HCE consiste en son couplage avec des éléments de sécurité localisés sur une plateforme Cloud plutôt que localisés sur l'appareil lui-même. De la même sorte, le HCE peut permettre d'accéder à une plateforme d'éléments de sécurité présents sur le *cloud* plutôt que localement sur l'appareil [255].

1. `https://www.thalesesecurity.com/solutions/use-case/payments/host-card-emulation`

Le HCE permet également, couplé avec SmartCardAPI [23] (ou OpenMobileAPI) d'interagir avec la carte SIM du smartphone. Pascal Urien illustre [256] une telle utilisation dans un système de stockage CoAP / DTLS.

eSIM

L'eSIM, version embarquée de la carte SIM, est un élément de sécurité facilitant l'installation à distance et sécurisée de profils téléphoniques ; c'est-à-dire de toutes les données spécifiques à un abonnement (y compris la clé secrète d'accès au réseau). Par les mêmes moyens, le changement de profil opérateur est possible à distance : passage d'un opérateur de télécommunication A à un opérateur télécommunication B sur le terminal. Elle est généralement soudée dans le circuit de l'appareil l'utilisant, mais peut se trouver également sous le format amovible. L'eSIM conserve le composant de sécurité, utilisé dans l'identification et l'accès aux communications mobiles, ainsi que pour sécuriser l'installation d'un nouveau profil opérateur sur le terminal. Un système de *remote provisioning* a été défini [40] par la GSMA (organisme en charge des standards de la communication mobile qui représente près de 800 opérateurs et constructeurs de téléphonie mobile) afin de charger les profils de télécommunication adaptés à l'opérateur choisi au sein de l'élément de sécurité.

Avec l'eSIM, les fabricants d'objets peuvent intégrer l'élément de sécurité vide de tout profil opérateur. C'est ultérieurement, au moment du déploiement de l'objet et suivant le choix du client quant à l'opérateur préféré, que le profil correspondant sera téléchargé "sur le terrain" et à distance. Cette souplesse facilite le processus de fabrication. Makino et al. proposent [178] une solution de *remote provisioning* des profils respectant la norme GSMA3.1 offrant la souplesse décrite pour les objets de l'IoT.

Security controllers

Les *security controllers* sont des circuits dédiés aux opérations cryptographiques composés d'un crypto processeur et d'un emplacement de stockage sécurisé (*tamper-proof*). Les opérations effectuées par le crypto processeur utilisent des clés cryptographiques stockées de manière sécurisée au sein de cet emplacement. Lorsqu'une opération de chiffrement ou de déchiffrement est requise par le composant, ce dernier l'effectue sans sortir la clé de chiffrement de son stockage sécurisé. Ainsi, les *security controllers* permettent de garantir les notions de confidentialité et d'authentification tout en se protégeant contre les attaques physiques. Le niveau de robustesse de ces composants, aussi bien au niveau matériel que logiciel, est évalué par des normes telles que le celles établies par le FIPS (*Federal Information Processing Standards*) du NIST, ou encore celles du *Common Criteria* (CC). Les *security controllers* constituent des méthodes de sécurisation matérielle suscitant l'intérêt de la communauté scientifique dans la sécurisation de l'Internet des Objets [168].

PPUFs

Les PPUFs (*Public Physically Unclonable Functions*), introduits par Beckmann et Potkonjak [60], consistent en des fonctions physiques inclonables (PUFs) dont la simulation est faisable, mais requiert un temps important avec de grandes capacités de calculs. L'utilisation de PPUFs permet de proposer de nouveaux protocols à clés publiques tout en se prémunissant contre des attaques physiques ou par canaux auxiliaires. Le schéma de conception du PUF devient ainsi la clé publique. Les PPUFs et plus généralement les PUFs constituent des pistes d'exploration particulièrement intéressantes pour la sécurisation de l'Internet des Objets en raison de leurs capacités à fournir des capacités d'authentification à coûts réduits [270].

TPM

Les puces TPM (Trusted Platform Module) sont des solutions de sécurisation complètes. Elles sont composées d'un crypto processeur pour effectuer les opérations cryptographiques, d'un générateur aléatoire et d'un stockage sécurisé pour la gestion de clés cryptographiques. Les ordinateurs utilisent les puces TPM dans leurs processus de chiffrement pour empêcher le déchiffrement des données lorsque la puce n'est pas présente (ordinateur différent). Les capacités de sécurisation offertes par les puces TPM suscitent l'intérêt de la communauté IoT. Après avoir éprouvé la technologie sur son système d'exploitation bureautique, Microsoft propose de l'intégrer dans sa version destinée aux objets connectés [232].

L'utilisation de TPM dans le milieu de l'IoT permet de garantir la gestion d'identités et d'offrir un stockage assurant à la fois l'intégrité et la confidentialité [125]. Ces notions de sécurité sont particulièrement intéressantes dans un environnement de capteurs connectés où l'on souhaite s'assurer de la provenance et de l'intégrité des données ne pas divulguer d'information sur la donnée transmise sans pour autant mettre en place des notions de non-répudiation.

Furtak et al. [113] proposent d'utiliser des composants TPM pour atteindre un niveau de sécurité "militaire" au sein d'un réseau de capteurs sans-fil connectés (WSN). Le réseau est ainsi composé d'une multitude de clusters eux-mêmes composés de plusieurs capteurs et d'un CSN (*Collecting Sensor Node*) chargés d'effectuer les opérations cryptographiques. La sécurité ne pouvant s'appliquer sur les capteurs, elle est apportée à la bordure du réseau (*edge*) afin d'offrir une sécurisation acceptable.

2.3.2 Menaces sur l'IoT

Menaces vs vulnérabilités

Le domaine de l'Internet des Objets est, comme tout système informatique, soumis à de multiples menaces. Une menace est un ensemble de circonstances pouvant causer un préjudice [206]. Ce terme n'est pas à confondre avec une vulnérabilité qui correspond à la potentielle exploitation d'une faiblesse du système pour causer un préjudice [206]. Il est possible de mettre en place des mesures de sécurité pour empêcher une menace de se transformer en vulnérabilité. Ces deux notions se

retrouvent incluses, avec le profil de l'attaquant, dans la définition de risque faite par Bromander et al. [77].

Facteurs descriptifs d'une menace

Humayed et al. [144] proposent de décrire une menace à l'aide de 5 facteurs qui sont : la source, la cible, le motif, le vecteur d'attaque et les conséquences. La source et le motif correspondent au profil de l'attaquant, la cible et le vecteur d'attaque correspondent davantage à l'attaque en elle-même. Le dernier facteur, les conséquences sont particulièrement intéressantes, car elles permettent de déterminer les potentielles vulnérabilités d'un système si aucune mesure n'est prise pour prévenir ces menaces. Les conséquences sont également les principaux éléments analysés lors de la catégorisation des menaces d'un système.

Taxonomie des menaces

Les premiers travaux d'identification des menaces sur le domaine de l'Internet des Objets remontent à 2010 avec Babar et al. [54]. Dans leur document, les auteurs proposent une taxonomie des menaces pour l'Internet des Objets regroupant les menaces relatives à la gestion d'identité, le stockage, les communications, la sécurité embarquée et les menaces physiques. Cette taxonomie, bien que complète au niveau de l'objet, n'aborde pas la forte connectivité présente entre les objets. Cette dernière est abordée par Roman et al. [222] dans leur description des menaces sur l'IoT. Les auteurs abordent les problématiques de sécurité des protocoles et du réseau, la gestion des données et la confidentialité, la gestion d'identité, la confiance et la gouvernance ainsi que la tolérance aux fautes du système.

Hodo et al. [140] reprennent les analyses de Symantec [241] pour proposer une catégorisation plus "haut niveau" abordant les menaces relatives au déni de service (DoS), les virus et malwares, les atteintes à la protection des données, et les faiblesses dans la sécurité des réseaux. Contrairement aux travaux précédents, les menaces proposées par Hodo et al. s'accentuent sur l'aspect logiciel des menaces. Roman et al. [222], quant à eux, abordent les notions de sécurité générales (identité, confidentialité, confiance...). Il est donc intéressant de prendre en compte l'union de ces deux travaux pour avoir une vision périphérique et exhaustive des menaces pouvant impacter les objets connectés. La figure 2.5 illustre l'union de ces deux classifications.

Ces travaux sont partiellement rejoints par ceux de Hazane [136], Hossain et al. [142], et les recommandations de l'OWASP IoT [7] avec l'identification de menaces et problèmes de sécurité relatifs à la disponibilité, la confidentialité et l'intégrité des éléments de l'IoT.

2.4 Les attaques dans l'IoT

La multitude de menaces qui pèsent sur l'Internet des Objets et les vulnérabilités qui s'en suivent rendent possibles de nombreuses attaques sur ces systèmes. Les éléments de l'IoT sont sensibles aux mêmes attaques que les systèmes informatiques

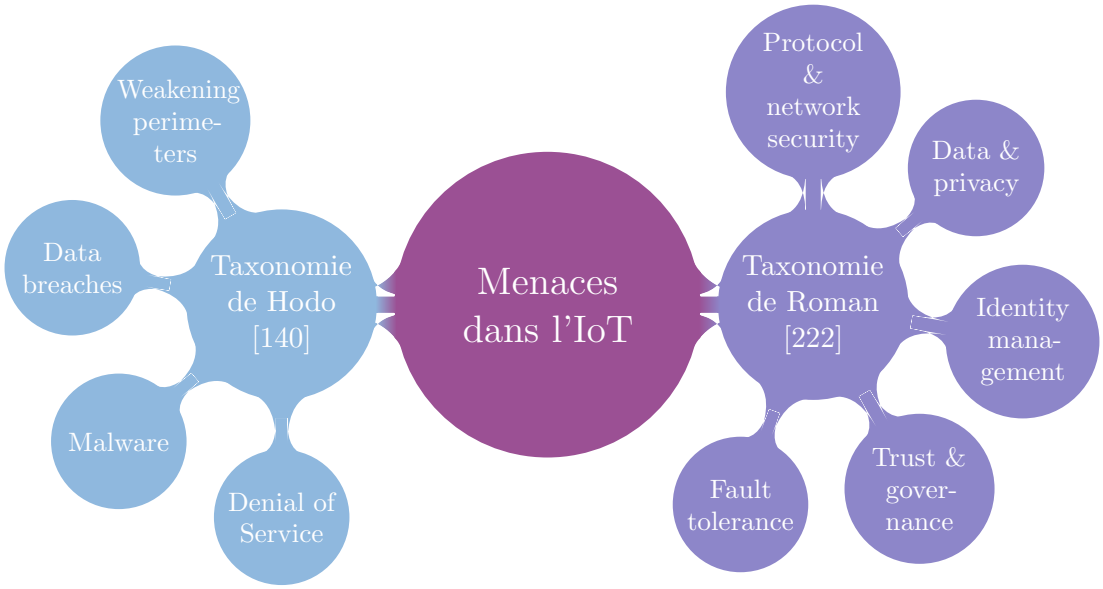


FIGURE 2.5 – Union des menaces de Roman et al. [222] et Hodo et al. [140]

TABLE 2.2 – Aperçu des attaques relatives à chaque couche de l'architecture IoT (d'après [169]).

	Perception	**Network**	**Application**
Attaques	Capture de nœuds	Déni de service	*Phishing*
	Injection de code malveillant	*Spoofing*	Virus / Vers malveillants
	Injection de fausses données	*Sinkhole*	Scripts malveillants
	Rejeu	*Wormhole*	
	Cryptanalyse	*Man in the middle*	
	Canaux auxiliaires	Routage d'information	
	Écoute illicite	*Sybil*	
	Interférence	Accès non autorisé	
	Privation de mise en veille		

non contraints (ordinateurs, serveurs, smartphones...). La majorité des attaques dans le domaine de l'IoT exploitent des vulnérabilités inhérentes aux systèmes contraints qui ne peuvent appliquer de mesures de sécurité suffisantes pour s'en protéger. Dans leur étude, Lin et al. [169] mettent en exergue les besoins de l'IoT en décrivant diverses attaques ainsi que les couches de l'architecture impactées. Le Tableau 2.2 associe ces attaques avec les couches de l'architecture IoT impactées.

<table>
<tr><th>Physical Attacks</th><th>Network Attacks</th><th>Software Attacks</th><th>Encryption Attacks</th></tr>
<tr><td>Node Tampering</td><td>Traffic Analysis Attacks</td><td rowspan="2">Virus and Worms</td><td>Side Chanel Attacks</td></tr>
<tr><td>RF Interference</td><td>RFID Spoofing</td><td rowspan="6">Cryptanalysis Attacks:
a) Ciphertext Only Attack
b) Known Plaintext Attack
c) Chosen Plaintext or Ciphertext Attack</td></tr>
<tr><td>Node Jamming</td><td>RFID Cloning</td><td rowspan="2">Spyware and Adware</td></tr>
<tr><td>Malicious Node Injection</td><td>RFID Unauthorised Access</td></tr>
<tr><td>Physical Damage</td><td>Sinkhole Attack</td><td rowspan="3">Trojan Horse</td></tr>
<tr><td>Social Engineering</td><td>Man In the Middle Attack</td></tr>
<tr><td rowspan="2">Sleep Deprivation Attack</td><td rowspan="2">Denial of Service</td></tr>
<tr><td rowspan="2">Malicious scripts</td><td rowspan="4">Man In the Middle Attack</td></tr>
<tr><td rowspan="2">Malicious Code Injection on the Node</td><td rowspan="2">Routing Information Attacks</td></tr>
<tr><td rowspan="2">Denial of Service</td></tr>
<tr><td>Sybil Attack</td></tr>
</table>

FIGURE 2.6 – Classification d'attaques sur l'IoT proposée par Andrea et al. (source : [49])

2.4.1 Types d'attaques

La sécurisation des éléments de l'IoT passe en premier lieu par l'identification des types d'attaques affectant ces systèmes. Plusieurs travaux portent sur cette identification par le biais de la définition de catégories d'attaques et la proposition de taxonomies. Hossain et al. [142] ont proposé une première taxonomie très complète en 2015. Elle caractérise les attaques selon plusieurs critères, comme les capacités des objets, la localisation de l'attaquant, le type de l'attaque (passive ou active), les objectifs de l'attaque (dégâts et stratégie), le niveau de compromission du système et les couches protocolaires empruntées pour la réalisation de l'attaque. Malgré la grande diversité de caractéristique présentée, cette taxonomie rencontre ses limites à catégoriser l'ensemble des attaques.

La taxonomie d'attaques proposée par Andrea et al. [49] (illustrée en Figure 2.6), plus facile à appréhender, catégorise les attaques suivant le niveau sur lequel elles agissent : attaques physiques, logicielles, sur le réseau et mécanismes de chiffrement. Les auteurs ont délibérément retiré les attaques environnementales de leur taxonomie jugeant qu'un attaquant n'avait pas de contrôle sur ces phénomènes. Cette taxonomie n'empêche pas qu'une attaque puisse appartenir à plusieurs catégories comme c'est le cas pour les attaques par déni de service (*Denial of Service* DoS).

Cette taxonomie est reprise par Deogirikar et Vidhate [92] en complément de certaines caractéristiques de la taxonomie de Hossain et al. [142] : à savoir, le type de l'attaque, la localisation de l'attaquant, le préjudice de l'attaque. Ces nouveaux travaux introduisent les notions de détection et prévention des attaques ainsi que les vulnérabilités exploitées afin de proposer une méthodologie de comparaison d'attaques (illustrée dans le Tableau 2.3).

TABLE 2.3 – Caractéristiques de comparaison d'attaques sur l'IoT d'après [92]

Paramètres	**Descriptions**
OSI Layer	La couche du modèle OSI visée par l'attaque
Attack Type	Attaque passive ou active
Attacker Location	Positionnement de l'attaquant dans le système (Interne ou Externe)
Attack Threat	Propriété de sécurité menacées
Damage Level	Impact de l'attaque
Detection Chance	Les chances de détection l'attaque
Possibility of Prevention	Possibilité de prévention de l'attaque
Attacks based on	En quoi consiste l'attaque (ex : dénis de service)
Vulnerability	Vulnérabilité exploitée par l'attaque
Existing solutions and their limitations	Solutions existantes pour se prémunir de l'attaque et leurs limitations

En parallèle de ces travaux, Ronen et Shamir [223] proposent une nouvelle taxonomie d'attaques spécifiques à l'Internet des Objets basée sur les fonctionnalités des objets. Dans la première catégorie, les attaques n'exploitent pas la fonctionnalité de l'objet et ne s'intéressent qu'aux capacités de calcul et de connectique de l'objet. Les auteurs abordent ensuite les attaques qui cherchent à réduire ou détruire les fonctionnalités d'un objet. En troisième catégorie sont présentées les attaques reposant sur un mauvais usage des fonctionnalités de l'objet. Enfin, une quatrième catégorie d'attaques regroupe les attaques relatives à l'extension des fonctionnalités offertes par l'objet dont l'attaque [223] fait partie.

2.4.2 Exemple d'attaques

La majeure partie des attaques touchant l'Internet des Objets sont classables suivant la taxonomie de Andrea et al. [49] et la variante proposée par Deogirikar et

Vidhate [92] tout en respectant la classification proposée par Ronen et Shamir [223].

2.4.2.1 Denial of Service (DoS)

Les attaques de déni de service peuvent affecter l'ensemble d'un système et relever de plusieurs des catégories décrites par Andrea et al. [49] : le côté logiciel en bloquant l'accès à certaines fonctionnalités, le composant lui-même en le privant de sommeil ou encore le réseau tel qu'effectué dans l'attaque sur le dispositif *Amazon Key* [79]. L'attaque menée par le *botnet* Mirai [139] est également représentative de cette catégorie d'attaque.

2.4.2.2 Node Tampering

Alors que les attaques par DoS se résument à restreindre les fonctionnalités, l'altération des éléments propose souvent de les étendre. L'attaque proposée par Ronen et Shamir [223] altère le comportement d'une ampoule connectée en faisant varier son intensité de manière imperceptible à l'œil nu, afin de créer un canal d'exfiltration d'informations.

2.4.2.3 Malicious Code Injection on the Node

L'injection de code malicieux dans un nœud afin de le corrompre représente une attaque importante. En fonction du code introduit dans l'objet, il est possible de changer l'intégralité de son comportement. Barcena et Wueest [59] proposent d'exploiter une faille dans le processus de mise à jour du firmware afin de faire télécharger une version compromise de celui-ci.

2.4.2.4 RF Interference

Les attaques par interférences radio recouvrent une grande variété d'attaques au niveau de la réception/émission des fréquences radio. Francillon et al. [110] présentent une attaque relayant les communications entre une voiture et une *Passive Keyless Entry and Start* (PKES). L'attaque permet de relayer les communications entre la clé et la voiture en possédant une antenne à proximité de chacune de ces entités. D'autres attaques plus représentatives de cette catégorie consistent à saturer les fréquences radio afin de perturber les communications radio (*radio jamming*).

2.4.2.5 Virus, Worms, Spyware, Trojan ...

Les attaques à l'aide de virus, ou vers préoccupent la communauté. On s'attend [85] à ce que l'engouement autour des *ransomwares* et autres virus pour les systèmes informatiques se propage à l'Internet des Objets à la manière du *botnet* Mirai [139].

2.4.2.6 Side Channel, Cryptanalysis

Les algorithmes de chiffrement subissent des attaques par canaux auxiliaires (Side Channel Attacks) souvent suivies d'analyses cryptographiques afin d'exploiter

les informations recueillies. L'une des attaques les plus populaires est l'attaque par analyse temporelle (*Timing Attack*) [158] consistant à analyser le temps de réponse des algorithmes pour en déduire des informations sur la clé de chiffrement. D'autres attaques consistent à analyser la consommation électrique des composants durant l'exécution de l'algorithme tel que les attaques par analyse différentielle de consommation (*Differential Power Analysis*, DPA) [157], ou encore à injecter des fautes lors de l'exécution de l'algorithme de chiffrement (*Differential fault analysis*, DFA) [64] afin de faire ressortir des informations sur la clé de chiffrement utilisée.

Il existe une multitude d'attaques possibles sur les objets de l'Internet of Things allant de la plus basique comme la destruction du matériel, n'ayant pas de prérequis en compétences, à la cryptanalyse demandant une forte compréhension du fonctionnement des algorithmes cryptographiques et de leur exécution.

2.4.3 Modèle de l'attaquant

La sécurisation d'un système requiert de savoir contre quelle entité il doit être protégé. En règle générale et encore plus particulièrement dans le cadre de l'Internet des Objets, il est important de définir un modèle qui reflète les capacités et objectifs d'un attaquant afin de protéger un objet contre ce dernier.

La définition d'un modèle d'attaquant appliqué à l'Internet des Objets requiert de s'intéresser aux capacités de l'attaquant (ses moyens) et à ses objectifs. Stellios et al. [246] présentent un modèle d'attaquant composé de 3 notions, la première correspond à l'accessibilité à l'objet (physique ou logique), la seconde aux capacités de l'attaquant (Compétences, Ressources ...) et la dernière aborde les motivations de l'attaquant. Ce modèle, bien que très basique, peut encore être simplifié en intégrant l'accessibilité à l'objet dans les capacités de l'attaquant. En effet, accéder à un objet sous-entend un coût : technique (comme pénétrer un bâtiment, démonter l'objet, se brancher sur l'objet), temporel ou encore monétaire (exemple : déplacement jusqu'à l'objet ou encore achat de matériel).

Il serait alors possible de définir un attaquant en fonction de ses moyens et de ses objectifs. La figure 2.7 présente ces deux notions et illustre leurs sous-catégories, à savoir : les compétences et ressources de l'attaquant, ainsi que les différents types d'objectifs d'un attaquant. Cette représentation n'est pas exhaustive et peut inclure une multitude de nouvelles sous-notions.

Autres modèles d'attaquants

Lors de la définition d'un modèle d'attaquant, il est intéressant de reprendre les travaux effectués en cryptographie sur les modèles formels de sécurisation tels que le modèle de Dolev-Yao [95]. Ce modèle s'intéresse aux informations qui transitent au travers d'un canal de communication contrôlé par un attaquant et décrit les capacités de cet attaquant. Il n'est pas adapté au domaine de l'Internet des Objets puisqu'il ne prend pas en compte les aspects physiques des objets connectés. Rochetto et

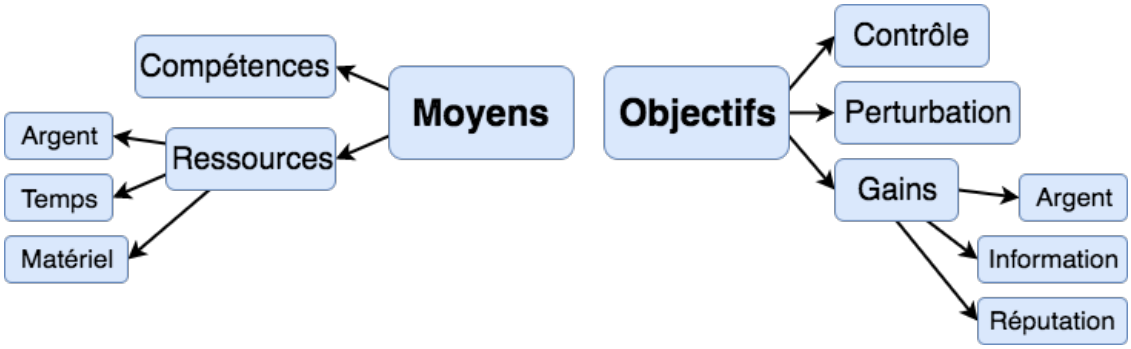


FIGURE 2.7 – Moyens et objectifs d'un attaquant

Tippenhauer [221] répondent à ce manque en proposant, dans leur modèle *Cyber-Physical Dolev-Yao* (CPDY), l'ajout de nouvelles règles. Malgré la pertinence de ce modèle et sa capacité à prendre en compte des attaques par canaux auxiliaires, il souffre des mêmes manques que le modèle de Dolev-Yao [95], à savoir le fait de ne pas prendre en compte le côté probabiliste d'une attaque (l'attaquant devine la clé de chiffrement) [135]. L'utilisation de tels modèles dans l'Internet des Objets semble compliquée étant donné la grande diversité des objets obligeant les modèles à redéfinir de nouvelles règles à chaque nouvelle configuration d'objet et d'usage. De plus, la notion de "gain" ne semble pas avoir une grande place dans ces modèles, or, cette notion correspond à l'objectif même de l'attaque même si elle peut revêtir des formes diverses suivant la nature de l'attaque.

Il existe également des modèles qui se focalisent sur une fonction de sécurité précise comme le modèle Freiburg Privacy Diamond (FPD) [279] qui se concentre sur l'évaluation des mécanismes d'anonymat en mobilité. Modèle particulièrement intéressant pour des objets mouvants tels que les smartphones ou encore voitures, mais qui ne concerne qu'une sous-partie des objets connectés et n'aborde pas l'ensemble des notions de sécurité.

Profils d'attaquants

Les travaux de Mosenia et Jha [195] abordent brièvement les modèles d'attaquants en présentant des profils typiques. Les auteurs présentent les objets de l'IoT comme sensibles à une multitude d'attaquants ("*Occasional hackers*", "*Cybercriminals*", "*Hacktivists*", "*Governments*"), mais n'abordent pas le profil du chercheur en sécurité (ou souvent appelé "*White-Hat*").

On suppose les attaquants intelligents, c'est-à-dire qu'ils ne cherchent pas à attaquer un système à perte (que ce soit une perte de temps, d'argent, ou autre). Si l'on respecte le modèle "Moyens / Objectifs" que l'on a défini ci-dessus, cela signifie que lors d'une attaque, un attaquant va chercher à maximiser ses gains avec un minimum de moyens (cas idéal : Moyens << Objectifs). Ce raisonnement s'applique également au profil du chercheur en sécurité puisque ce dernier va investir des moyens et du temps, pour percevoir un salaire (Objectifs).

Onik et al. [202] abordent également cette problématique de modèle d'attaquant à l'aide de 5 profils typiques d'attaquants :

Joker : Bricoleur, hacker occasionnel, curieux

White-Hat hackers : Chercheurs en sécurité

Black-Hat hackers : Cyber-criminels

Little sisters : Espions industriels

Big brothers : Gouvernements ou organisations liées

Les profils d'attaquants définis par Onik et al. [202] peuvent être rapprochés avec le modèle "Moyens / Objectifs" abordé plus haut. Le tableau 2.4 illustre le rapprochement entre ces deux notions.

TABLE 2.4 – Qualification des moyens et objectifs à partir des profils d'attaquants définit par Onik et al. [202]

Profil d'attaquant	**Moyens**	**Objectifs**
Joker	Faibles	Moyen
White-Hat hackers	Moyen	Élevé
Black-Hat hackers	Variable	Variable
Little sisters	Élevés	Élevés
Big-Brothers	Très élevés	Très élevés

Les moyens du *Joker* sont considérés comme faibles. Un attaquant de cette catégorie ne va pas investir beaucoup d'argent ni acheter du matériel hors de prix ; il va chercher à assouvir sa curiosité, bricoler pour comprendre le fonctionnement des objets. Les objectifs d'un tel attaquant sont moyens, puisque même avec un niveau de débutant, il est en mesure d'effectuer des attaques significatives. L'attaque de Caudill [79] représente parfaitement un tel attaquant. Dans cette dernière, un attaquant (un livreur par exemple) peut empêcher une serrure connectée de se verrouiller après sa visite, lui permettant de pénétrer de nouveau dans le logement. L'attaque étant basée sur un simple déni de service, un simple bricoleur débutant est en mesure de la perpétrer.

Sur le même principe, le profil *White-Hat* est représenté par l'attaque de Ronen et Shamir [223] sur des Philips Hue qui, avec un minimum de matériel et de bonnes connaissances, ont réussi à faire varier l'intensité de l'ampoule de façon imperceptible à l'œil humain pour exfiltrer des données.

Le profil de l'attaquant *Black-Hat*, ou cyber-criminel est très variable. Certains attaquants possèdent de forts moyens et peuvent concevoir des attaques de grande ampleur et d'autres sont plus limités. Il existe cependant plusieurs cas impliquant des cyber-criminels qui ont effectué des attaques d'envergure mondiale avec des moyens relativement faibles telle que l'attaque du *botnet* Mirai [139].

La distinction entre les profils *Little sisters* et *Big brothers* reste faible. Ces deux profils d'attaquants nécessitent des moyens importants pour atteindre des objectifs élevés. Le premier profil correspond au domaine de l'entreprise alors que le second se

focalise sur les gouvernements, cependant il arrive fréquemment que les deux types d'entités se réunissent dans un intérêt commun. C'est le cas de l'alerte TA18-106A [15] de l'US-CERT (United States Computer Emergency Readinesss Team) qui mentionne que des acteurs importants du monde *cyber* exploitent des routeurs et switchs du monde entier afin de pratiquer l'espionnage et s'emparer de propriétés intellectuelles.

L'attaque du virus *Stuxnet* [163] est représentative des attaquants de type *Big-Brothers*. L'objectif de ce virus était de détruire des cibles militaires. Sa conception était plus complexe que les autres virus connus à cette date. Stuxnet est supposé provenir d'un gouvernement [160] et certains supposent même qu'il provient des États-Unis [231].

Deuxième partie
Contributions

L'enjeu de cette partie est de présenter les différentes contributions qui ont été réalisées pendant cette thèse. Nous commençons par présenter notre première contribution, une stratégie de déploiement de services de sécurité dans l'IoT qui minimise les frais de déploiement. Nous modélisons un réseau IoT sous la forme d'un graphe et traduisons nos contraintes de placement par un problème de graphe. Ce problème correspond à l'identification d'un ensemble dominant qui favorise la sélection des sommets avec un poids élevé. Finalement nous proposons une approche de résolution de ce problème qui repose sur l'utilisation des mesures de centralité de graphes.

Par la suite, nous mentionnons notre seconde contribution, la formalisation des problèmes de placement de services de sécurité et leurs modélisations au travers une structure sémantique, à savoir une ontologie. En complément d'une description détaillée de cette structure, nous présentons comment cette dernière peut être utilisée pour la résolution de problèmes de placement ainsi que la comparaison de solutions à ces problèmes. Nous illustrons nos propos en construisant une base de connaissances, issue des travaux de notre première contribution, ainsi qu'un outil de résolution et comparaison des solutions de placement qui repose sur cette dernière.

Finalement, cette partie se conclut par la présentation de notre troisième contribution relative à l'ancrage d'implémentations *white-box* sur leur environnement local. Nous y abordons les différentes propriétés nécessaires pour constituer ce mécanisme d'ancrage, la compilation des programmes *white-box* ainsi que leur déploiement. Cette contribution se conclut avec une mise en application du mécanisme d'ancrage au travers divers scénarios qui démontrent l'apport d'une telle approche. Notre mécanisme d'ancrage permet notamment de nous prémunir contre les attaques par extraction de code et le vol de l'objet.

Chapitre 3

Placement de services de sécurité dans le fog

Ce chapitre présente la première contribution de la thèse relative au placement de services dans le fog. Nous décrivons brièvement les principes de ces problèmes et montrons l'intérêt d'une approche basée sur les outils de théorie des graphes. Une méthodologie est décrite et mise en application sur un jeu de données issues de la ville de Santander. Il en résulte une stratégie de déploiement de services de sécurité avec un impact minimal sur le flux d'information ainsi que sur les coûts de déploiement.

Sommaire

3.1 Contexte du fog computing 45
3.2 Placement de services dans le fog 47
3.3 Notre modèle de sécurité 53
3.4 Placement de services basé sur la centralité de graphes 56
3.5 Expérimentations 63
3.6 Conclusion 77

3.1 Contexte du fog computing

Les capacités limitées des objets de l'IoT restreignent l'accès d'une grande partie des objets aux services de sécurité tels que la cryptographie, la gestion d'identité ou encore la détection d'intrusion. Ce n'est que très récemment que des systèmes embarqués intègrent des fonctionnalités de sécurité. On aperçoit dorénavant des microcontrôleurs équipés de crypto processeurs afin d'accélérer les calculs cryptographiques, des éléments de sécurité intégrés aux circuits imprimés ou encore des

enclaves sécurisées mises à disposition des développeurs pour la gestion de leurs secrets. Malgré l'augmentation de la proportion d'appareils qui utilisent ces nouveaux composants, les fonctionnalités de sécurité ne sont pas systématiquement exploitées [103]. Ces fonctionnalités peuvent être mal implémentées, avoir des vulnérabilités ou encore être dépassées. Prenons l'exemple des algorithmes cryptographiques, les tailles de clés préconisées sont en constante expansion, le niveau de sécurité d'un système peut alors être impacté en cas de manque de suivi dans le processus de mise à jour. La durée de vie d'un objet connecté, une fois déployé dans la nature, est d'environ une dizaine d'années. Un manque de maintenance de ces appareils durant cette durée peut aboutir à l'exploitation de multiples vulnérabilités. Ce phénomène a d'ailleurs été observé lors de l'attaque du botnet Mirai en 2014 [50].

La solution la plus évidente serait de remplacer la totalité des objets déployés par des versions plus récentes et performantes afin qu'ils soient en mesure d'accueillir les services de sécurité nécessaires aux applications souhaitées. Cette approche n'est évidemment pas envisageable en raison du coût financier que cela impliquerait.

D'une manière générale, les produits actuels de l'IoT se reposent sur l'utilisation du cloud pour remédier à ces problèmes. Les services interrogés par ces objets sont centralisés dans des infrastructures très performantes telles que des serveurs ou datacenters.

Particulièrement intéressante au niveau logistique, cette approche rencontre ses limites depuis quelques années maintenant. En effet, de nombreuses applications IoT requièrent désormais des temps de réponse très courts. La latence induite par l'utilisation de services dans le cloud rentre alors en conflit avec ce désir de se rapprocher d'un traitement des données en temps réel.

Une autre préoccupation majeure de l'utilisation de services centralisés dans le cloud concerne la confidentialité de la donnée en elle-même. Plusieurs méthodes existent pour traiter des données personnelles sans compromettre leur confidentialité dont notamment le chiffrement homomorphe, cependant son utilisation est fortement contraignante [118].

Afin de répondre aux besoins de réduction des temps de latence, un nouveau paradigme, le Edge computing a été inventé. Ce dernier englobe d'autres paradigmes comme le Fog computing ou encore le Multi-access Edge Computing (MEC).

Nous avons défini le fog computing à la Section 2.1.3 comme une plateforme hautement virtualisée qui fournit des services traditionnellement présents dans le cloud computing (traitement de données, stockage, connectivité, etc.), généralement, mais pas exclusivement, en périphérie du réseau [71, 70]. Le Multi-access Edge Computing (MEC) quant à lui représente un paradigme initialement pensé par les opérateurs de télécommunication pour les réseaux cellulaires et qui vise à étendre les capacités du cloud computing en bordure du réseau cellulaire [208].

L'influence des opérateurs de télécommunication est particulièrement visible sur l'emplacement des plateformes MEC (ou noeuds MEC du réseau). Contrairement au fog computing où les noeuds fog sont positionnés à des emplacements stratégiques entre l'utilisateur et le cloud, les plateformes MEC ne peuvent être déployées qu'à l'intérieur des Radio Access Networks (RANs) comme sur les Base Stations (BS) qui fournissent des services mobiles dans une certaine zone géographique [208].

L'emplacement de ces plateformes MEC et fog est d'une importance cruciale pour offrir des services de la meilleure qualité possible au plus grand nombre d'utilisateurs.

La migration des services initialement hébergés dans le cloud vers la bordure du réseau présente de nouvelles opportunités de recherche relatives à l'emplacement des noeuds fog, mais surtout concernant le placement des services. Ce nouveau problème s'intitule le Service Placement Problem (SPP) [229] ou le Fog Service Placement Problem (FSPP) [244]. Un service doit-il être le plus proche de la bordure de réseau ou au contraire, en proximité du cloud ? Les travaux sur le positionnement de service dans le fog consistent à apporter une réponse à ces interrogations. La réponse étant que cela dépend du service à déployer.

3.2 Placement de services dans le fog

Le problème de placement de services dans le fog est un domaine qui suscite beaucoup d'intérêt de la part de la communauté IoT depuis quelques années. D'un point de vue très général, ce problème consiste à trouver les emplacements les plus appropriés pour des services en respectant des contraintes préétablies. Ce problème est communément associé à des problèmes d'optimisation de nature différente.

Dans cette section, nous allons décrire les principes fondamentaux du problème de placement de services, les différentes définitions possibles de ce problème puis nous nous focaliserons sur notre problématique en explorant une forme spécifique de ce problème. Par la suite, nous décrirons quelques travaux connexes à notre approche du problème de placement et terminerons par une définition plus formelle de notre problème qui nous suivra tout au long de ce document.

3.2.1 Modélisation générale du problème de placement

D'après Salaht et al. la définition d'un problème de placement de services dans l'IoT requiert la modélisation de l'application, l'infrastructure et le modèle de déploiement [229]. La Figure 3.1 reprend ces modélisations en fournissant un aperçu de leurs compositions.

Généralement, la modélisation de l'infrastructure consiste à représenter le réseau IoT sur lequel les services doivent être déployés. Cela comprend les ressources, ainsi que les caractéristiques disponibles sur le réseau. Les ressources correspondent aux appareils physiques tels que les serveurs, les capteurs, actionneurs, ordinateurs personnels, smartphones, alors que les caractéristiques font allusion aux unités de performance de ces appareils (fréquence du processeur, mémoire, latence du réseau, bande passante ...).

Alors que le modèle de l'infrastructure est lié au réseau, les modèles d'application et de déploiement sont quant à eux spécifiques aux services à placer dans le réseau. Ces derniers seront abordés plus en détail dans les sections suivantes.

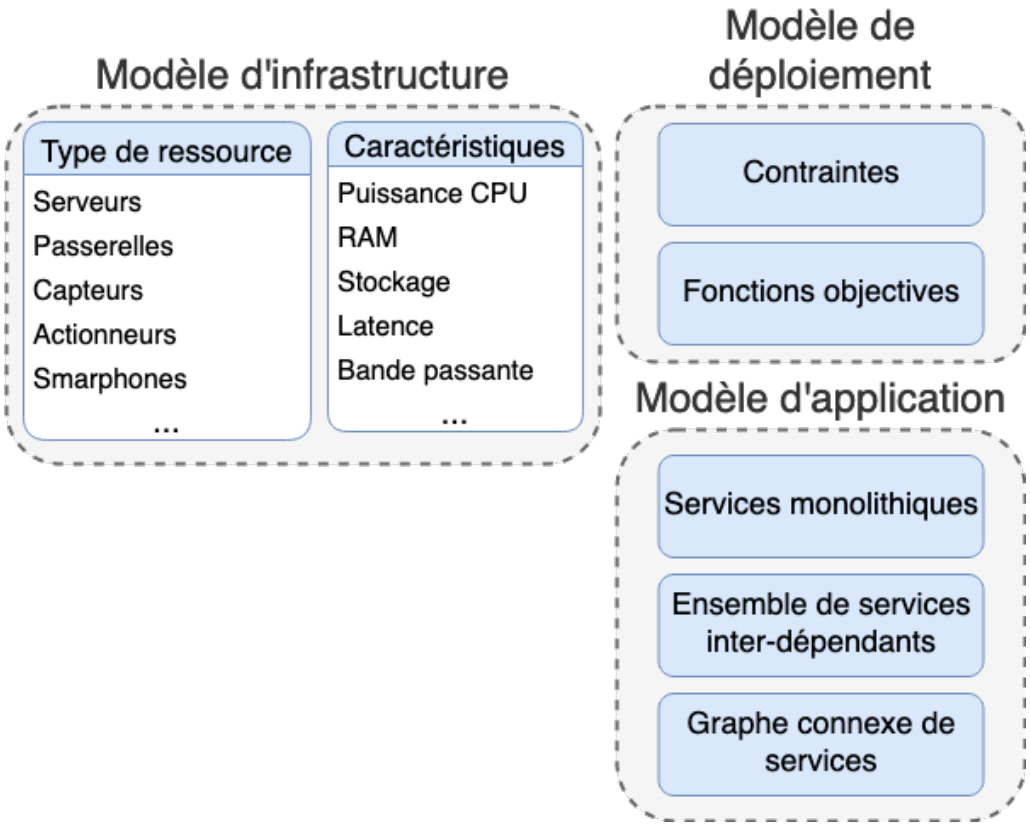


FIGURE 3.1 – Aperçu des modèles pour la description du problème de placement de service (SPP) (inspiré de [229])

3.2.2 Modèle d'application

Le modèle d'application décrit les différentes structures d'applications qu'il est possible de rencontrer sur les travaux sur le placement de services de sécurité. Ces applications sont généralisables sous la forme de trois catégories : les applications constituées de services indépendants, celles qui comportent des services interdépendants et les dernières, dont l'ensemble des services, dépendent les uns des autres.

Représentons notre modèle d'application sous la forme d'un graphe $A = (S, I)$ avec S qui correspond à l'ensemble des services et I représente l'ensemble des interactions entre ces services.

Notre première catégorie d'applications, les services indépendants (également nommés monolithiques) peuvent être représentés par $A = (S, I)$ où $I = \emptyset$ en raison de l'absence d'interaction entre les services. Cette catégorie d'application est la plus communément abordée dans la littérature [244, 245, 147, 229].

La seconde catégorie comprend les applications dont certains services sont interdépendants. Si l'on reporte cela sur notre graphe $A = (S, I)$, cela signifie que $I \neq \emptyset$. Le graphe A est alors constitué de plusieurs composantes connexes. Une représentation sous la forme d'un graphe orienté est également nécessaire pour représenter l'orientation de l'interaction (arcs au lieu d'arêtes). En raison de l'in-

terdépendance des services, la complexité de positionnement est plus importante que pour les services monolithiques. Cette modélisation de dépendance de services intéresse particulièrement la communauté scientifique en raison de son réalisme. De nombreux travaux définissent leurs problèmes de positionnement en fonction de ces applications tels que Xia et al. [267] ou Doriguzzi-Corin et al. [97].

La troisième catégorie représente les applications dont l'ensemble des services interagissent ensemble. Comme pour la catégorie précédente, le graphe représentatif $A = (S, I)$ est orienté cependant, il n'est constitué que d'une unique composante connexe. Les problèmes de placement définis selon ce modèle d'application sont moins courant et reposent principalement sur une représentation de l'application sous la forme d'un DAG (Directed Acyclic Graph), un graphe orienté sans circuit [229, 96].

3.2.3 Modèle de déploiement

Le modèle de déploiement d'une application consiste à modéliser les conditions dans lesquels les services de l'application seront placés dans le réseau. Ce modèle est constitué de deux concepts systématiquement utilisés dans les travaux sur le placement de services dans l'IoT, à savoir les contraintes et fonctions objectives [229].

3.2.3.1 Contraintes

Les contraintes d'un problème de placement de service correspondent aux conditions qui doivent être satisfaites pour déterminer qu'une solution de placement est valide. Les critères utilisés dans l'élaboration de ses contraintes portent généralement sur les informations disponibles dans le modèle d'infrastructure, à savoir les caractéristiques des appareils et du réseau.

Les contraintes portent principalement sur 3 aspects : les composants physiques du réseau, le réseau en lui-même, et l'application. Certaines études prennent également en considération l'aspect énergétique dans l'élaboration des contraintes de placement d'une application [76, 194].

Les travaux sur le placement de services dans le fog se focalisent systématiquement sur ces 3 catégories avec une légère variance en fonction des services adressés. La majorité des travaux abordent le placement de services d'une manière générale et définissent donc des contraintes généralisables. La fréquence du processeur, la mémoire, le stockage, la latence ou encore la bande passante sont des caractéristiques récurrentes de ces études pour définir des contraintes [244, 245, 147, 96, 267]. Certains travaux abordent les notions de localisation géographique [96] ou encore des informations sur l'appareil tels que son type ou la présence d'un composant particulier [244, 245, 96].

Dans leur étude, Brogi et al. [76] présentent une taxonomie des contraintes associées au problème de placement de services dans le fog. Pour cela, les auteurs abordent des approches relatives à la topologie du réseau. Ils précisent cependant qu'elle regroupe les travaux qui nécessitent d'être exécutés sur des appareils comportant des composants spécifiques tels que des capteurs ou actionneurs. La topologie au sens de structure générale du réseau comme nous pourrions l'entendre par une représentation sous la forme de graphe n'est pas abordée.

3.2.3.2 Fonctions objectives

Les travaux sur le positionnement de services s'associent en principe à des problèmes d'optimisation. En complément de contraintes relatives aux applications, il est fréquent qu'une fonction objective soit fournie. Cette fonction agit comme un évaluateur pour une solution de placement donnée. La comparaison des scores entre deux solutions de positionnement permet alors de déterminer quelle solution est la meilleure.

Les critères d'optimisation sont multiples [76, 229], ils peuvent concerner les aspects réseau avec l'optimisation de la bande passante et des temps de réponse [147, 267], la minimisation des coûts de déploiement [96], ou encore l'harmonisation des performances [244, 245].

Les fonctions objectives sont fortement dépendantes des applications à déployer.

3.2.4 Descriptif supplémentaire du problème de placement de services

La description des problèmes de placement de services nécessite davantage d'informations contextuelles concernant les conditions dans lesquelles l'orchestrateur, l'entité responsable du placement, doit effectuer son analyse.

Salaht et al. proposent une taxonomie du placement de services qui aborde ces données contextuelles [229]. Dans leurs travaux, les auteurs proposent de catégoriser les problèmes de placement selon quatre catégories, la coordination du placement, la connaissance lors de la décision de placement, la dynamicité et le support de la mobilité. Cette taxonomie est illustrée dans la Figure 3.2.

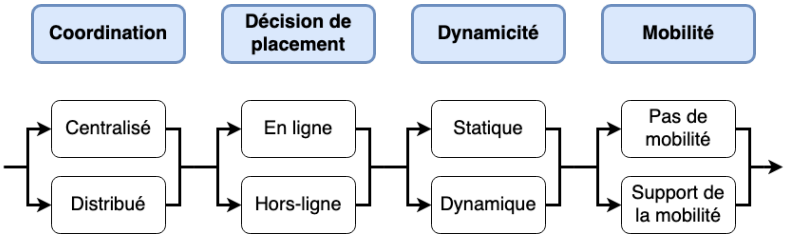


FIGURE 3.2 – Taxonomie du placement de Service (inspiré de [229])

La coordination au niveau de l'orchestrateur est une information primordiale lors de la définition d'un problème de placement. Effectivement, un système décentralisé soulève de nouvelles interrogations concernant le placement des orchestrateurs, mais également sur le découpage du réseau.

De manière similaire, si une décision de placement s'effectue hors-ligne l'orchestrateur doit avoir une connaissance totale du réseau à un instant précis pour identifier une solution de placement le plus approprié possible. À l'inverse, une décision en ligne offre plus de souplesse lors de l'identification du placement. De plus, l'orchestrateur peut mesurer les changements induits par le déploiement d'un service et réagir en conséquence pour adapter le placement des suivants.

Les aspects dynamiques et mobiles se rejoignent en partie concernant l'impact sur le processus décisionnel. Un support de la mobilité implique une capacité de l'orchestrateur à prendre connaissance du changement d'état de ses informations.

3.2.5 Algorithmes pour résoudre les problèmes de placement

Brogi et al. et Salaht et al. proposent [76, 229] de regrouper les algorithmes pour résoudre ces problèmes de placement en trois catégories.

La première catégorie d'algorithmes regroupe les travaux qui proposent une démarche mathématique dans la résolution des problèmes de placement de services. Ces travaux se focalisent principalement sur l'utilisation des principes de Integer Linear Programming (ILP) [244, 245] ou son homologue Mixed Integer Linear Programming (MILP) [97]. Cette dernière est particulièrement intéressante pour le placement de services sur un faible nombre d'appareils, mais devient vite très inefficace lorsque le nombre d'appareils croît fortement. Doriguzzy-Corin et al. présentent dans leurs travaux une formulation sous la forme MILP de leur problème de placement [97]. Les auteurs utilisent dans un premier temps un solveur pour résoudre leur problème de placement, mais observent rapidement les limitations de cette approche à mesure que la taille de leur réseau augmente. Les auteurs proposent finalement une nouvelle approche heuristique qui accélère la résolution du problème tout en offrant des performances raisonnables.

La seconde catégorie d'algorithmes pour la résolution des problèmes de placement de services repose sur l'utilisation d'algorithmes de recherche ainsi que les différentes heuristiques associées. Reposant souvent sur des algorithmes gloutons ou comportements heuristiques, ces méthodes offrent un compromis entre la précision des résultats des méthodes mathématiques et le temps de résolution [97, 267]. Les travaux de Gupta et al. [127] ainsi que ceux de Mahmud et Buyya [177] sont les travaux les plus représentatifs de cette catégorie. Dans ces derniers, les auteurs proposent d'effectuer le placement de service en partant de la bordure du réseau vers la cloud en se focalisant uniquement sur les contraintes physiques des appareils. D'autres travaux proposent des approches reposant sur des techniques de backtracking tel que Xia et al. [267].

Finalement, la troisième catégorie regroupe les différentes méthodes et algorithmes exotiques proposés dans la littérature. On y retrouve des approches basées sur le deep learning [252], ou encore sur la théorie des jeux comme les travaux de Zhang et al. qui reposent sur le jeu de Stackelberg [273].

Malgré une représentation des réseaux IoT sous la forme de graphe, nous n'avons pas relevé d'approches du problème de placement de services qui reposent sur l'utilisation d'algorithmes de théorie des graphes.

3.2.6 Notre problématique

Les problèmes de placement de services peuvent se résumer de manière très abstraite par la phrase suivante : soit un ensemble de services, où puis-je et où dois-je placer ces services pour maximiser leurs efficacité. Nous avons discuté dans les

sections précédentes de la grande diversité des problèmes de placement de services. Si un problème se définit en partie par son modèle d'application, on remarque aisément que deux problèmes qui ont des modèles d'application différents sont par conséquent eux même différents.

Notre problématique initiale consiste à nous interroger sur l'emplacement idéal d'un ensemble de services de sécurité pour offrir le meilleur compromis entre sécurité et coût de sécurisation sur notre réseau. Lors de notre étude des travaux sur le placement de services, nous avons cependant observé qu'en dépit d'une représentation sous la forme de graphe des réseaux IoT, les outils de théorie des graphes n'étaient pas utilisés dans la résolution du problème. Il se pose alors la question de la pertinence de ces outils sur la résolution de ce problème. Sommes nous capables d'utiliser les outils de théorie des graphes pour résoudre des problèmes de placement de services dans l'IoT et en particulier des services de sécurité ?

Pour répondre à ces interrogations, il nous est nécessaire de définir notre problème de placement.

3.2.7 Description de notre problème de placement

Nous avons discuté dans les sections précédentes (3.2.1 et 3.2.4) de différentes notions nécessaires à la description d'un problème de placement.

Nous avons fait le choix de définir des conditions d'orchestration optimales telles qu'elles sont fréquemment établies dans la littérature. L'orchestration du positionnement est effectuée de manière centralisée évitant ainsi de nouvelles interrogations sur le partage des appareils ou la sélection des orchestrateurs. L'identification de la solution est effectuée hors-ligne ce qui signifie que notre orchestrateur bénéficie de toute la connaissance nécessaire afin de déterminer une solution. Nous n'avons pas opté pour le support de la mobilité des appareils ni la dynamicité du réseau. Ces conditions d'orchestrations sont illustrées dans la Figure 3.3 et mises en correspondance avec la taxonomie de placement de Salaht et al. [229] (Figure 3.2)

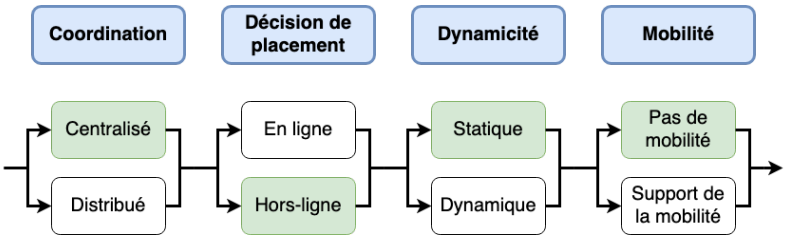


FIGURE 3.3 – Nos conditions de placement selon la taxonomie du placement de Service de Salaht et al. [229])

Représentons notre réseau IoT sous la forme d'un graphe $G = (V, E)$ avec les objets pour sommet et les arêtes qui représentent les communications entre les objets. Définissons maintenant notre modèle d'application comme un graphe $A(S, I)$ où S représente l'ensemble des services et I l'ensemble des interactions entre ces services. Dans le cas de notre problème, nous aurons $I = \emptyset$.

Notre premier objectif est de placer chaque élément $s \in S$ de A dans G pour former une solution de positionnement appelée Ps. Soit deux fonctions, C et FO correspondant respectivement à une fonction de vérification de contraintes (qui retourne Vrai si toutes les contraintes sont satisfaites) et une fonction objective (qui évalue une solution de positionnement et retourne le score de cette solution). Notre second objectif est de déterminer pour quelle solution Ps, sachant $C(Ps) = Vrai$, le score de $FO(Ps)$ est le plus élevé (si maximisation) ou le plus faible (si minimisation).

3.2.8 Travaux connexes

Nous avons abordé, dans les sections précédentes plusieurs travaux sur le positionnement de services dans le fog. La majorité de ces travaux se focalisent sur le placement de services génériques sans aborder la sécurité [147, 244, 245, 96, 267, 58] et sans utiliser des méthodes reposant sur la théorie des graphes [147, 244, 96, 267].

Parmi les travaux étudiés, seul un papier applique des principes de théorie des graphes dans le cadre de placement de sécurité. Les travaux de Doriguzzi-Corin et al. abordent le problème de placement de services chainés avec une approche mathématique (ILP) [97]. Les auteurs présentent également une heuristique de résolution de leur problème qui repose sur un graphe pondéré en fonction des capacités de ses sommets. Pour ce faire, leur heuristique exploite la structure de graphe via l'intermédiaire de calculs de plus courts chemins. En dehors du fait que notre problème de positionnement est différent de celui des auteurs, notre différence principale réside dans l'approche du placement. Doriguzzi-Corin et al. abordent le problème de placement au sein d'un chemin dont l'origine et la destination sont fixées [97]. Notre approche diffère d'autant plus que dans notre cas, nous souhaitons protéger l'ensemble du réseau et non un chemin spécifique. Pour ce faire, nous devons analyser l'ensemble des plus courts chemins du réseau ce qui correspond à avoir un grand nombre de points de départ et d'arrivée. De plus, les auteurs représentent leurs contraintes à l'aide de programmation linéaire là où nous exploitons notre structure de graphes avec la notion d'ensembles dominants et de centralités. L'approche des auteurs est toutefois très intéressante aux vues de résultats obtenus et plus particulièrement concernant la technique de pondération de graphe utilisée.

3.3 Notre modèle de sécurité

La sécurité absolue est une illusion. Lorsque l'on affirme qu'un système d'information est sécurisé, c'est en réalité un abus de langage. Il faudrait également déterminer dans quel cadre le système est sécurisé, ou plus particulièrement contre quel type de menaces le système est protégé. Dans les sous-sections suivantes, nous allons décrire notre modèle de sécurité et plus particulièrement contre quels types de menaces et surtout d'attaquants nous souhaitons nous prémunir.

3.3.1 Modèle de menace

Les travaux de Stellios et al. [246] présentent plusieurs modèles d'attaquant IoT qu'ils décrivent d'après 3 catégories : l'accès à l'appareil, les capacités de l'attaquant et ses motivations. Dans notre étude, nous nous focaliserons sur le modèle de l'*outsider* correspondant à un attaquant qui n'a pas d'accès direct à l'objet. Cet attaquant ne peut attaquer un système qu'au travers un intermédiaire tel qu'illustré dans la Figure 3.4. Ce dernier peut traverser plusieurs appareils avant d'atteindre sa cible comme représentée pour les chemins d'attaque 1 et 2, aussi, l'idée consiste à bloquer l'attaque en s'interposant sur l'un des chemins empruntés pour l'attaque.

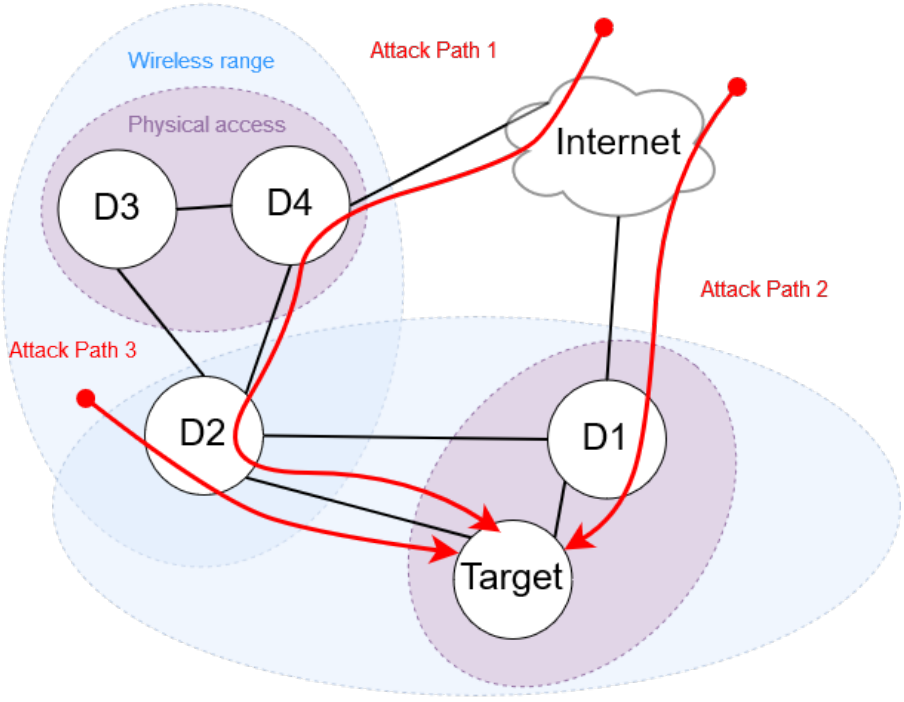


FIGURE 3.4 – Exemples de chemins d'attaque par un attaquant de type *outsider*.

Les ressources de notre attaquant correspondent à celles d'une personne normale avec des capacités techniques modérées (entre néophyte et expert) dont les motivations fluctuent en fonction du réseau ciblé. Plus spécifiquement, notre modèle d'attaquant se situe entre celui du hacker occasionnel et du cybercriminel tel que défini en sous-section 2.4.3 d'après les travaux de Mosenia et Jah [195] ainsi que ceux de Onik et al. [202].

La présence de l'attaquant à portée directe de l'objet ou sa capacité physique d'y accéder ne sont pas considérées en raison des coûts plus importants impliqués dans la réalisation de telles attaques. Nous supposons que la quantité d'attaquants est inversement proportionnelle à leurs moyens, cela signifie qu'il existe un très grand nombre d'attaquants avec de faibles moyens alors qu'une quantité plus réduite possèdent des moyens importants.

Le choix d'un tel modèle est un compromis entre le coût de sécurisation du système en fonction du nombre d'attaquants évincés.

La contribution de ce chapitre permet toutefois de prévenir des attaques provenant d'autres modèles d'attaquants. Dès lors que ces derniers ne possèdent pas la capacité d'accéder physiquement à l'objet et qu'ils ne se situent pas à portée de communication directe avec celui-ci, leurs attaques peuvent être évincées. En fonction des capacités qu'un attaquant va mettre en œuvre pour accéder au périphérique, ce dernier pourra potentiellement ne plus appartenir à la catégorie *outsider*.

3.3.2 Security As A Service (SecAAS) pour le edge computing

Dans cette contribution, nous proposons de nous prémunir contre le modèle de menace décrit précédemment (Section 3.3.1) en déployant des services de sécurité en bordure du réseau. Cette procédure, reposant sur le paradigme de fog computing, permet notamment de réduire fortement la latence, d'accroitre le contrôle sur la génération et le traitement de données tout en maintenant une proximité physique avec les appareils.

Les services sont communément déployés sous la forme de Virtual Network Functions (VNF) au sein de machines virtuelles ou containers dans les objets. Dans certains cas plus rares, ces services peuvent prendre la forme de nouveaux appareils introduits dans le réseau en complément ou remplacement d'appareils existants.

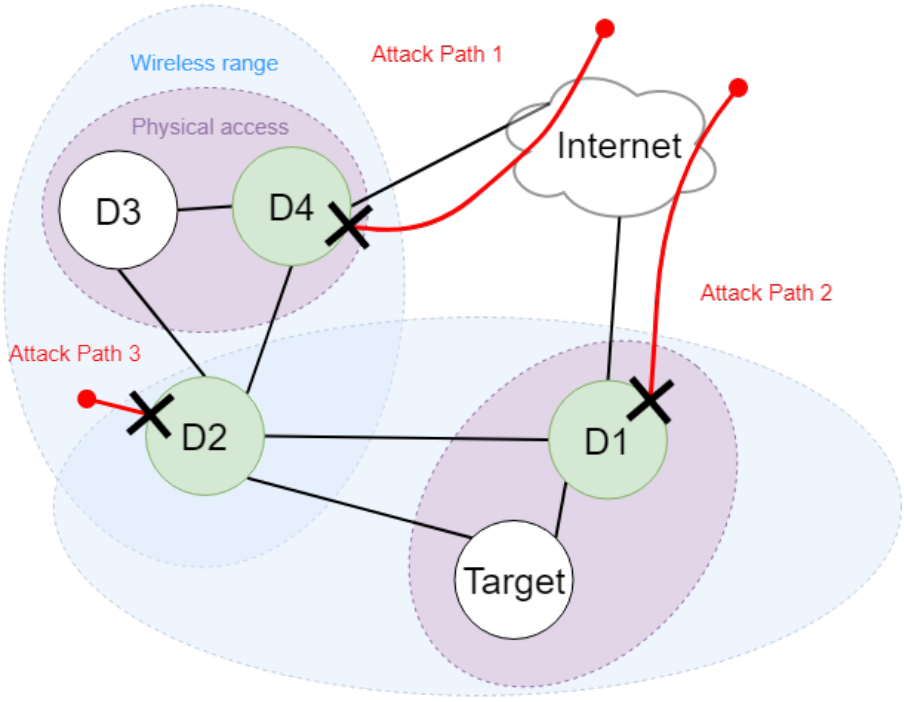


FIGURE 3.5 – Exemple de services de sécurité (nœuds verts) sur le modèle considéré.

La Figure 3.5 illustre un exemple de déploiement de services de sécurité pour se prémunir comme les menaces identifiées à la Figure 3.4. On distingue que les trois chemins d'attaque (rouge) précédemment identifiés sont maintenant interrompus par des nœuds hébergeant des services de sécurité (vert).

Le déploiement de services de sécurité ne doit pas nécessairement s'effectuer sur l'ensemble des appareils du réseau. Cela doit être effectué vis-à-vis des menaces identifiées sur le réseau.

3.4 Placement de services basé sur la centralité de graphes

Les objectifs énoncés dans la section précédente nécessitent qu'une attention particulière soit apportée concernant le placement des services de sécurité. Effectivement, ces derniers doivent être en mesure de satisfaire au maximum leurs contraintes spécifiques tout en assurant une couverture de service maximale. De plus, les coûts de déploiements inhérents à ces services incitent à l'optimisation de leurs placements afin de maximiser l'efficacité tout en minimisant les frais.

Nous proposons, dans cette section, une méthode de placement de services de sécurité pour l'IoT reposant sur ces préoccupations. Cette méthode permet notamment de répondre à notre première problématique introduite à la Section 1.5 consistant à s'interroger sur le placement optimal, dans un réseau IoT, de services de sécurité pour bénéficier d'une sécurisation maximale tout en limitant les coûts de déploiement.

Afin d'étudier le problème de positionnement de services dans l'IoT, nous devons dans un premier temps modéliser notre réseau.

Les travaux de cette contribution sont itératifs, dans un premier temps nous avons déterminé un modèle simplifié supposant que tous les objets du réseau bénéficiaient de capacités identiques et qu'aucun n'avait de capacités suffisantes pour l'accueil de services de sécurité. Dans cette première approche, une fois les localisations de déploiement de services identifiées, il est nécessaire de procéder à l'amélioration de ces appareils. De même, notre modèle initial suppose que l'ensemble des interfaces de communications sont identiques aboutissant ainsi à un modèle faiblement réaliste.

Dans un second temps, nous proposons d'accroitre le réalisme de notre modèle en l'améliorant afin qu'il puisse prendre en considération les capacités inhérentes des objets du réseau IoT.

3.4.1 Formalisation du problème

En raison de la forte diversité des objets de l'IoT, nous proposons de représenter notre réseau IoT sous la forme d'un graphe dont les sommets représentent les objets et les arêtes correspondent aux communications entre ces objets. Une telle représentation offre un fort degré d'abstraction et permet d'appliquer les outils issus de la théorie des graphes au domaine de la sécurité IoT.

Au cours de ce paragraphe, nous noterons $d = d(a, b)$ la distance entre deux sommets a et b. Si a désigne un objet et b un service de sécurité, d correspond

au nombre de sauts nécessaires à l'objet (a) pour atteindre ce service (b). Le cas idéal consisterait à ce que chaque objet puisse systématiquement héberger leurs propres services de sécurité ($d = 0$), cependant cela n'est pas toujours le cas. Un compromis doit donc être effectué entre la distance du service par rapport à un objet et la sécurité qu'il apporte. L'hébergement du service sur un voisin direct ($d = 1$) de l'objet semble acceptable en supposant que l'attaquant n'interfère pas avec la communication. Éloigner ce service à une distance plus élevée ($d \geq 2$) fait apparaitre les premiers problèmes de sécurité relatifs à la présence d'intermédiaire qui relaient l'information. De plus, la notion d'accès direct ($d = 1$) des objets aux services respecte les paradigmes de fog et MEC consistant à ce que les appareils puissent accéder directement aux services les plus proches [208].

Respecter le compromis ainsi déterminé correspond à s'assurer que chaque objet du réseau bénéficie d'un accès direct à l'un des appareils qui offrent des services de sécurité que nous appellerons nœud de sécurité.

Rappelons que $G = (V, E)$ désigne un graphe non orienté où V correspond à l'ensemble des sommets et E l'ensemble des arêtes. $N(x)$ définit par $y \in V | (x, y) \in E$ est le voisinage de x. On définit G à partir d'un réseau IoT en prenant comme ensemble de sommets l'ensemble des objets et comme ensemble d'arêtes l'ensemble des interactions entre ces objets.

La modélisation du besoin d'accéder directement depuis chaque objet à un nœud de sécurité sur le graphe correspond à l'identification d'un ensemble dominant du graphe. Un ensemble dominant est un sous-ensemble S de sommets d'un graphe $G = (V, E)$ tel que pour chaque sommet $x \in V$, soit x appartient à S ($x \in S$), soit x est adjacent à un sommet y appartenant à S ($(x, y) \in E$ et $y \in S$).

En suivant cette définition d'ensemble dominant, on remarque que V forme un ensemble dominant. Dans notre cas, cette solution n'est pas souhaitable car elle est onéreuse, mais également irréalisable puisque nous avons défini que tous les objets ne pouvaient héberger leurs services de sécurité. Afin de réduire les frais de conversion des objets en nœuds de sécurité, il est souhaitable d'identifier un ensemble de taille minimum. La recherche d'un ensemble dominant de taille minimum est un problème d'optimisation prouvé NP-difficile, l'approximation d'un ensemble de taille minimum reste difficile [115].

C'est pour ces raisons que nous n'envisageons pas de trouver un ensemble dominant de taille minimum, trouver un minimum local (ensemble minimal) est suffisant.

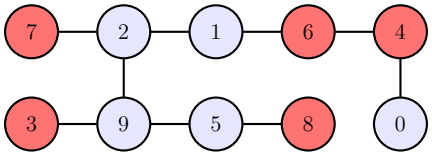


FIGURE 3.6 – Ensemble dominant (nœuds rouges) localisé en dehors du flux d'informations.

3.4.2 Construction d'un ensemble dominant minimal avec des sommets pondérés

La méthode proposée dans cette section suppose que nous possédons une fonction de pondération des sommets d'un graphe. L'algorithme 2, suggéré par Paul Dorbec, retourne alors un ensemble dominant de taille minimal en fonction des poids de ces sommets.

Pour ce faire, les sommets sont visités par ordre croissant en fonction de leurs poids et sont conservés en dehors de l'ensemble dominant jusqu'à ce qu'il n'y ait plus le choix de faire autrement. Ce comportement permet la sélection des sommets dont la valeur de priorité est la plus importante dans l'ensemble dominant. La généricité de l'algorithme ainsi décrit fait qu'il peut être utilisé dans des contextes différents, de plus, son efficacité dépend en grande partie de la fonction de pondération des sommets utilisée.

Notons F l'ensemble des sommets qui n'ont pas encore été parcourus par l'algorithme. Lorsqu'un sommet est analysé, ce dernier est soit déplacé dans l'ensemble dominant (DS), dans celui des sommets couverts (C, sommets adjacents à ceux de DS) ou soit déplacés dans l'ensemble B correspondant aux sommets visités qui seront ultérieurement membres de C mais nécessitent un voisin dans DS.

Une liste d'adjacence notée $N_F(x)$ contient, à chaque instant, les voisins d'un sommet x dans un ensemble F. Pour des raisons pratiques et de lisibilité, l'algorithme nécessite la définition des deux fonctions suivantes : $ForcedDominant$ et $Propagation$ décrites dans l'algorithme 1).

Algorithme 1 Fonctions utilisées dans l'algorithme 2

```
1: function ForcedDominant(node)
2:     si N_F(node) = ∅ alors
3:         Return True
4:     for all n ∈ N[node] ∩ B faire
5:         si |N_F(n)| = 1 alors
6:             Return True
7:     Return False
8: function Propagation(node)
9:     for all n ∈ N[node] ∩ B faire
10:        Move n from B to C
```

La fonction $ForcedDominant$ détermine si un sommet donné doit être ou non ajouté dans l'ensemble dominant (DS). Pour ce faire, il vérifie si le sommet n'a plus d'autres choix pour être couvert ou si l'un de ses voisins dépend de lui pour sa couverture. Dans ces deux cas, l'ajout de ce sommet dans DS est nécessaire.

La fonction $Propagation$ est appelée lorsqu'un sommet est ajouté dans l'ensemble dominant. Elle a pour effet de couvrir l'ensemble des voisins du sommet ajouté à DS qui a déjà été visité et qui se situent dans l'ensemble B.

L'algorithme proposé (2) parcours un ensemble donné de sommets. Si lors de ce parcours, un sommet n'est pas déjà couvert, mais qu'il possède un voisin encore non

parcouru (signifiant que le sommet pourra être couvert par ce voisin ultérieurement), il est mis en attente (ensemble B). Lorsqu'un sommet ne possède plus de voisins non parcourus pour le couvrir, ce dernier est ajouté dans l'ensemble dominant (DS).

Puisque la liste des sommets fournie en entrée de l'algorithme ($vList$) est triée par ordre croissant de poids, l'ensemble dominant ainsi identifié (DS) favorise les sommets les plus lourds. Les sommets les plus légers sont dans un premier temps mis en attente dans C ou B permettant ainsi de privilégier l'ajout des sommets les plus lourds dans DS.

Algorithme 2 Algorithme du Ordered Dominating Set (ODS).

Input :
$vList$: ordered list of V
Output :
DS : dominating set vertices

1: $F \leftarrow V$
2: $DS, B, C, N_F \leftarrow \emptyset$
3: **tant que** $vList \neq \emptyset$ **faire**
4: $a \leftarrow vList[0]$
5: **si** $a \in F$ **alors**
6: **si** ForcedDominant(a) **alors**
7: Move a from F to DS
8: Propagation(a)
9: **sinon si** $|N_F(a)| = 1$ **alors**
10: $b \leftarrow N_F(a)[0]$
11: Move a from F to C
12: Move b from F to DS
13: Propagation(b)
14: **sinon**
15: Move a from F to B
16: **sinon**
17: $vList \leftarrow vList - a$
18: Return DS

3.4.3 Topologie IoT et mesures de centralité

La découverte d'un ensemble dominant minimal n'est pas notre seule préoccupation. Nous veillons également à nous intéresser à la topologie du réseau afin que l'emplacement des sommets de l'ensemble dominant soit consistant avec le transit du flux d'information dans le réseau.

La Figure 3.6 présente un scénario ou les sommets de l'ensemble dominant (rouge) ne sont pas positionnés en travers du flux d'information. Un tel positionnement impacte fortement le coût des communications qui requièrent le passage par un nœud de sécurité.

Dans cette section, nous nous focaliserons sur la fonction de pondération mentionnée à la section précédente. Notre première approche consiste à pondérer les sommets en fonction de leur importance dans le graphe. L'importance d'un sommet dans un graphe est calculable à l'aide de mesures de centralité de graphe. Il existe plusieurs mesures de centralité, il est nécessaire de déterminer laquelle est la plus pertinente en fonction des propriétés de sécurité souhaitées. Les paragraphes suivants décrivent les quatre mesures les plus populaires, à savoir, la degree centrality, l'eigenvector centrality, la closeness centrality et la betweenness centrality. Nous verrons expérimentalement que cette dernière semble la plus appropriée pour notre situation.

Degree centrality

La degree centrality mesure l'importance d'un sommet en fonction de son degré, le nombre de voisins du sommet.

Eigenvector centrality

L'eigenvector centrality part du postulat que l'importance d'un sommet dépend de l'importance de ses voisins [226, 69]. Soit un graphe G, l'importance d'un sommet $i \in G$ est calculée par l'équation suivante dans laquelle n correspond au nombre de sommets de G, A est la matrice d'adjacence de G et λ_{max} représente la plus grande eigenvalue (valeur d'importance, positive d'après le théorème de Perron-Frobenius) :

$$C(i) = \frac{1}{\lambda_{max}} \sum_{j=1}^{n} A_{ij} C(j).$$

Closeness centrality

La closeness centrality mesure la proximité d'un sommet avec le reste des sommets du graphe [227]. Cette mesure est calculée en analysant tous les chemins les plus courts entre le sommet mesuré et l'ensemble des autres sommets du graphe. La valeur ainsi déterminée correspond à la moyenne des distances entre le sommet mesuré et le reste du graphe. Cette mesure se représente sous l'équation normalisée suivante où n correspond au nombre de sommets du graphe ainsi que g_{ij}, la taille de la géodésique (chemin le plus court) entre i et j :

$$C'(i) = \frac{n-1}{\sum_{\substack{j=1 \\ i \neq j}}^{n} g_{ij}}.$$

La closeness centrality est une mesure particulièrement adaptée lors que l'on souhaite déterminer quels sont les sommets à partir desquels il est possible d'atteindre le reste du graphe le plus rapidement possible. Parmi les utilisations possibles de cette mesure, il est possible de mentionner le placement de casernes de pompiers dans les villes [90] ou encore le placement de serveurs de mises à jour dans les réseaux IoT.

Betweenness centrality

La betweenness centrality correspond au nombre de fois où un sommet agit comme composant d'un chemin le plus court entre deux autres sommets du graphe [111]. L'importance d'un sommet i est exprimée à l'aide de l'équation suivante dans laquelle g_{jk} correspond au nombre de géodésiques (chemins les plus courts) reliant les sommets j et k avec $j \neq k$, on note $g_{jk}(i)$ le nombre de ces géodésiques qui traversent le sommet i :

$$C(i) = \sum_{\substack{i \neq j \\ i \neq k}} \frac{g_{jk}(i)}{g_{jk}}.$$

Supposons que, dans un graphe, l'information transite le long des chemins les plus courts, la betweenness centrality évalue la contribution d'un noeud sur le transit d'information du réseau.

En fonction des mesures de centralité utilisées, les valeurs d'importance des sommets peuvent diverger fortement. La Figure 3.7 illustre ces propos en mettant en évidence les sommets avec la plus grande valeur de centralité pour les quatre mesures décrites précédemment.

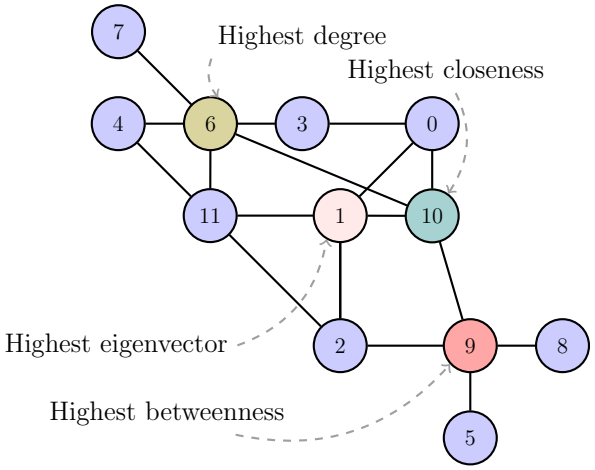


FIGURE 3.7 – Différences entre valeurs de centralité sur un graphe donné.

3.4.4 Considération des contraintes réelles

Nous avons décrit précédemment notre nécessité de réduire les frais associés au déploiement des services de sécurité dans le réseau avec notamment l'identification d'un ensemble dominant de taille minimale. Pour ce faire, une première méthode a été réalisée en se basant uniquement sur la topologie du réseau et les mesures de centralité de graphe. Ces travaux ont abouti à l'écriture et la publication d'un papier scientifique dans une conférence internationale [121].

Utiliser les mesures de centralité uniquement sur la topologie du réseau possède cependant quelques limites. En effet, dans notre première méthode, les capacités

des objets du réseau ne sont pas prises en compte, seule la topologie est considérée. Il serait alors pertinent d'introduire ces capacités dans notre modélisation afin d'adapter les méthodes de déploiement de services en fonction de ces services. Il semble cohérent de supposer que le déploiement d'un service virtualisé sous la forme d'une VNF sera potentiellement moins coûteux si l'objet possède déjà des capacités d'accueil suffisantes. Dans le cas contraire, un coût supplémentaire est à envisager pour accroitre les capacités de l'objet d'accueil. L'objectif est donc de favoriser le déploiement des services de sécurité sur les dispositifs qui bénéficient de capacités suffisantes pour ainsi limiter les frais de déploiement tout en préservant les contraintes d'optimisation préalablement définies.

Pour ce faire, nous définissons une notion de priorité (valeur unique) pour chaque objet liée à ses capacités telles que la cadence de son/ses processeurs, sa mémoire, son stockage, la bande passante ou encore toute autre caractéristique spécifique à cet appareil. Nous supposons ainsi qu'une pondération basée sur cette notion de priorité est appliquée sur les sommets de notre graphe de telle sorte que les objets les plus adaptés à recevoir les services de sécurité soient les plus lourds, qu'ils soient prioritaires. Les sommets qui correspondent à ces objets seront ainsi favorisés dans la sélection de l'ensemble dominant minimal.

Dans le cadre de nos expérimentations et l'évaluation des ensembles dominants identifiés, nous introduisons une mesure de qualité de l'ensemble dominant. La qualité d'un ensemble dominant correspond à la proportion de sommets prioritaires dans cet ensemble.

Afin d'accroitre la qualité de l'ensemble dominant minimal identifié, il est possible d'inclure les valeurs de priorité des sommets dans le calcul des valeurs de centralité. L'inclusion de ces valeurs dans le calcul des centralités est effectuée par l'intermédiaire d'une fonction de pondération d'arêtes afin d'associer la topologie du réseau aux valeurs de priorité.

Nous avons précédemment suggéré en Section 3.4.3 que la betweenness centrality semble la mieux adaptée à notre cas d'utilisation. Cette mesure repose fortement sur l'identification des chemins les plus courts dans un graphe pour déterminer les valeurs de centralité. L'introduction d'une fonction de pondération des arêtes du graphe se reposant sur les valeurs de priorité agit sur cette identification des chemins les plus courts. Cette approche a pour effet d'altérer les valeurs de centralité des sommets du graphe et donc de permettre à l'aide de l'algorithme 2 de favoriser l'ajout des sommets prioritaires dans l'ensemble dominant.

La pondération des arêtes permet, lorsque le graphe est parcouru et qu'il existe plusieurs chemins entre deux sommets, de sélectionner le chemin de poids le plus faible. Ainsi, il sera préférable de sélectionner le chemin le plus léger même si sa taille en nombre de sauts est supérieure aux autres chemins disponibles. Procéder de la sorte permet de favoriser les communications entre les objets à forte priorité et donc d'accroitre la qualité de l'ensemble dominant minimal identifié.

Les utilisations d'une telle méthode de positionnement concernant principalement tous les services qui requièrent d'être placés en travers du flux d'information. À tire d'exemple, deux objets voulant communiquer ensemble de manière sécurisée sont en mesure de le faire en utilisant les nœuds de sécurité comme des passerelles

pour leurs communications. Afin d'attester du passage de la communication au travers un ou plusieurs nœuds de sécurité, des mécanismes de preuve de passage peuvent être utilisés. Un exemple de ces mécanismes est le Proof Of Transit [75] consistant à partager une ou plusieurs parties de secrets entre les nœuds à attester en respectant le principe du partage de secrets de Shamir (Shamir Secret Sharing). Les nœuds de sécurité utiliseraient alors ces secrets ou parties de secrets pour signer les communications qui leur passent au travers. Les destinataires seraient alors en mesure de vérifier que l'ensemble des signatures requises ont été appliquées sur la communication attestant ainsi de l'application des services de sécurité demandés.

3.5 Expérimentations

Nous avons défini dans la section précédente, notre méthodologie d'un point de vue théorique. Dans cette section, nous illustrons l'application de cette méthodologie sur des données réelles issues du monde l'IoT. Notre objectif est d'identifier les emplacements optimaux afin de déployer des services de sécurité pour proposer un service de chiffrement bout-en-bout.

Les résultats obtenus sont ensuite interprétés confirmant ainsi l'apport de la méthodologie proposée sur le placement de services de sécurité dans un réseau IoT.

3.5.1 Jeu de données (Dataset)

Les dispositifs de l'IoT génèrent des quantités importantes de données cependant ces dernières ne sont pas toujours disponibles publiquement. La majorité des données accessibles sont constituées d'informations issues de capteurs. L'acquisition de données relatives aux communications de dispositifs IoT entre eux est cependant une tâche très difficile.

Au cours de notre étude sur l'IoT, nous avons acquis un ensemble de données comportant des informations sur les objets ainsi que sur leurs interactions. Ces données, que nous appellerons SIoT, sont issues des travaux dans le projet Social IoT [25].

Pour notre étude, nous avons généré un graphe issu de la matrice d'adjacence Ownership Object Relationship (OOR) comportant des informations relatives à des objets publics et privés. Nous avons par la suite filtré les objets privés afin d'obtenir un graphe connexe issu des objets publics du jeu de données. La Figure 3.8 représente le graphe généré depuis cette matrice d'adjacence, la composante conservée correspond à celle entourée par le cercle vert.

Le graphe ainsi obtenu correspond à un graphe non orienté composé de 1 458 sommets et de 35 657 arêtes dont les sommets représentent les objets du réseau IoT et ses arêtes, leurs communications. Les communications représentées dans le graphe sont en réalité des communications potentielles entre les objets. Dans les travaux de Social IoT [25], une arête est créé lorsque deux objets sont à portée de communication sans-fil. Cette portée varie en fonction des protocoles de communications disponibles

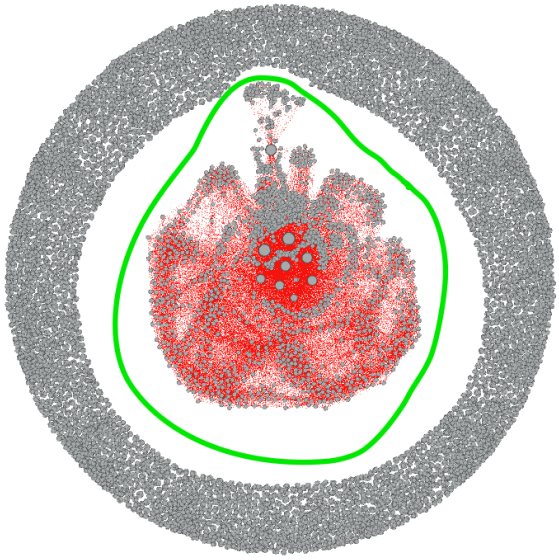


FIGURE 3.8 – Composante préservée (en vert) de la matrice d'adjacence OOR.

entre les objets ; les distances considérées sont les suivantes : Bluetooth : 40 mètres, Wi-Fi : 400 mètres, LoRa : 1 500 mètres.

En complément des multiples matrices d'adjacence fournies par SIoT, le dataset est accompagné d'un descriptif de ses objets. Le descriptif recense les informations suivantes : l'identifiant de l'objet, l'identifiant de son propriétaire, la catégorie de l'objet ainsi que des informations sur sa marque et son modèle. Malheureusement, ces dernières données sont fournies uniquement sous la forme d'identifiants dont la correspondance n'est pas fournie. Les informations relatives aux catégories d'objets sont cependant fournies, la Table 3.1 illustre la distribution de ces catégories en fonction des objets de notre graphe.

TABLE 3.1 – Distribution des catégories d'objets de notre graphe.

Catégorie	**Nombre d'appareils**	**Proportion**
Point of interest	95	6.5%
Environment and weather	140	9.6%
Indicator	10	0.7%
Street Light	506	34.7%
Parking	677	46.43%
Alarms	30	2%

Nous avons discuté du besoin de considérer des contraintes (Section 3.4.4) du monde réel dans l'élaboration de modèle. Pour ce faire, nous avons proposé d'introduire une valeur de priorité agrégeant plusieurs critères comme la fréquence du processeur, la mémoire ou encore le stockage disponible sur un appareil donné.

Dans le cadre de notre étude, la capacité d'un objet à accueillir un service de sécurité est binaire, soit il est capable d'accueillir ce service ou il ne l'est pas. Nous avons scindé nos valeurs de priorité en deux catégories, forte et basse priorités

représentant respectivement les objets capables d'héberger le service et ceux qui ne le sont pas.

L'introduction d'une nouvelle catégorie serait intéressante dans la représentation des capacités d'un objet. Nous discutons des difficultés inhérentes à l'ajout de catégories supplémentaires en section 3.5.6. Les difficultés ainsi rencontrées ne feraient cependant que de croître exponentiellement en fonction du nombre de catégories à traiter.

En l'absence de suffisamment d'informations sur les capacités réelles des objets de notre graphe, nous avons dû faire des choix dans l'affectation des valeurs de priorité. Le projet de Smart Santander duquel le dataset a été constitué présente différents projets en relation avec la ville intelligente [24]. L'un des projets proposés consiste en une application d'irrigation des parcs et jardins. Dans cette proposition, des lampadaires connectés sont utilisés comme répéteurs afin de relayer l'information entre les différentes interfaces de communication présentes dans le réseau. La ville de Santander n'est pas la seule à s'intéresser à rendre intelligents les systèmes d'éclairage publics. En effet, les villes de San Diego, Chicago et Los Angeles ont investi dans des lampadaires connectés regroupant un grand nombre de capteurs [22, 1, 114].

Ces appareils, initialement utilisés pour éclairer les voiries, voient leurs fonctionnalités se développer fortement. De nombreux modèles intègrent désormais des infrastructures de télécommunication (micro cell towers), fournissent de la couverture réseau au travers du Wi-Fi et proposent même d'agir comme borne de rechargement pour les véhicules électriques. L'entreprise NEC propose un modèle de lampadaire connecté qui intègre une Base Station (BS) 5G tout en offrant un accès direct à des applications cloud.

Suite à ces constatations, nous avons décidé de considérer les objets de notre dataset de catégorie "Street Light" comme des objets prioritaires en supposant que le déploiement de services sur ces appareils serait moins onéreux. De plus, l'omniprésente de ses objets dans le contexte de la ville connectée ainsi que leur forte couverture géographique en font des cibles de choix dans le déploiement de services de sécurité. Favoriser le déploiement de services de sécurité sur cette catégorie d'objet semble donc être pertinent dans un contexte de ville intelligente.

3.5.2 Calcul de priorité

Nous avons déterminé précédemment notre volonté d'avoir un clivage de nos valeurs de priorité en deux catégories, forte et basse priorité, respectivement H et L. Notons cependant que plus la valeur est faible, plus la priorité est importante. Un nœud avec une priorité de 1 aura donc une priorité plus importante qu'un nœud avec une priorité de 1.2.

Lors de nos expérimentations, les nœuds correspondants aux lampadaires connectés seront prioritaires avec une forte valeur de priorité H alors que les autres bénéficieront de faibles valeurs de priorité L. Ces valeurs de priorité entre les sommets de notre graphe sont par la suite utilisées dans notre fonction de pondération d'arêtes.

Soit notre graphe $G = (V, E)$, pour chaque sommet $a \in V$, notons $P(a)$ sa valeur de priorité. La fonction de pondération d'arête dérive directement des valeurs de P.

TABLE 3.2 – Distribution des poids des arêtes dans notre graphe

Poids	**Nombre d'arêtes**	**Proportion**
1	961	2.7%
1.2	9,088	25.49%
1.44	25,608	71.82%

Pour tout $e \in E$, avec a et b les extrémités de l'arête e, le poids de e correspond à : $W(e) = P(a) * P(b)$. La distribution des poids des arêtes de notre graphe est illustrée dans le tableau Table 3.2.

Pour rappel, appliquer une pondération sur les arêtes du graphe permet d'inclure les valeurs de priorité des nœuds dans le calcul des chemins les plus courts effectué par le calcul de centralité. L'idée est de trouver un compromis dans la sélection des noeuds entre favoriser les noeuds à forte priorité et pénaliser ceux dont la priorité est faible.

Nous avons déterminé expérimentalement que les valeurs de priorité les plus appropriées pour notre graphe sont $L = 1.2$ et $H = 1$. Ces valeurs dépendent évidemment beaucoup du jeu de données utilisé ; elles pourraient ne pas être pertinentes pour un autre graphe. La Figure 3.11 de la Section 3.5.5 illustre notre approche dans la sélection de ces valeurs de priorité. On y remarque l'évolution de nos critères d'évaluation en fonctions des différences entre nos valeurs de priorité. La Section 3.5.6 discute plus en détail ces choix et leurs répercussions.

3.5.3 Calcul de la centralité de sommets pondérés

Au début de cette section, nous avons déclaré vouloir effectuer le placement de services de sécurité en effectuant un ersatz de chiffrement bout-end-bout. Le principe consiste à ce que deux objets qui souhaitent communiquer ensemble de manière sécurisée, puissent le faire par l'intermédiaire de nœuds de sécurité situés à proximité de ces objets.

Afin d'éviter des frais de communication supplémentaires liés au mauvais positionnement du nœud de sécurité, un positionnement des services en travers du flux d'information est souhaité. L'utilisation de la betweenness centrality semble donc théoriquement plus appropriée. Un second avantage de cette mesure de centralité, partagé par la closeness centrality et exploité par la pondération des arêtes du graphes, consiste à intégrer nos mécanismes de priorité des nœuds sans modification de algorithme ni de notre méthodologie.

Lors du calcul des chemins les plus courts dans les cas de la betweenness et de la closeness centrality, les arêtes dont les poids sont les plus faibles (forte priorité) seront préférées. Au contraire, les arêtes dont les poids sont élevées (faible priorité) seront moins propices à être sélectionnées puisque qu'elles augmentent la distance totale du chemin. L'utilisation de la pondération des arêtes en fonction des priorités des sommets permet ainsi d'influer le calcul des valeurs de centralité pour donner plus d'importance aux sommets dont la priorité est élevée.

Une fois que le calcul des valeurs de centralité des sommets du graphe est effectué, nous pouvons appliquer notre méthodologie dont notamment l'Algorithme 2 afin d'identifier un ensemble dominant central de petite taille. Pour rappel, l'algorithme utilisé permet de favoriser l'ajout dans l'ensemble dominant des sommets dont la priorité est élevée. Pour ce faire, l'algorithme écarte, dans un premier temps, les sommets dont la priorité est faible. Un sommet est placé dans l'ensemble dominant uniquement si lorsque son placement en dehors n'est plus possible.

3.5.4 Métriques pour l'évaluation

Plusieurs métriques peuvent être utilisées pour évaluer les résultats de notre algorithme d'identification d'ensemble dominant central minimal. Pour reprendre les exigences concernant le nombre de sommets de l'ensemble ainsi que son positionnement, nous introduisons les métriques d'évaluation suivantes : le nombre de sommets de l'ensemble dominant, la qualité de l'ensemble dominant, la protection apportée par le positionnement de cet ensemble ainsi qu'une valeur normalisée de la protection indépendante de la taille du graphe.

Nous avons précédemment défini en Section 3.4.4 la qualité d'un ensemble dominant comme la proportion de noeuds prioritaires dans cet ensemble. Plus cette proportion est importante, plus la qualité de l'ensemble dominant sera forte. Dans le cadre de nos expérimentations, la qualité d'un ensemble dominant fait référence au nombre d'objets de type lampadaires connectés ("Street Light") qui font partie de l'ensemble dominant.

La protection procurée par un ensemble dominant est calculée à partir de la proportion de paires de sommets du graphe bénéficiant d'au moins un chemin le plus court qui est considéré sécurisé. Le calcul est effectué à l'aide de l'équation suivante :

$$protection = \frac{\sum_{a,b} S(a,b)}{\binom{n}{2}}$$

où :

$S(a,b) = 1$ si il existe un chemin sécurisé entre les sommets a et b
$S(a,b) = 0$ sinon.

Un chemin le plus court est considéré sécurisé si ses extrémités (origine, destinataire) sont adjacentes ou si chaque sommet de cette paire est soit membre de l'ensemble dominant du graphe ou adjacent à un sommet de l'ensemble dominant. La Figure 3.9 représente l'ensemble des combinaisons possibles qui définissent ce qu'est un chemin sécurisé. Les sommets en rouge sont membres de l'ensemble dominant alors que ceux en bleu peuvent, ou non, faire parti de cet ensemble.

Nous proposons également une version normalisée de la valeur de protection afin de bénéficier de valeurs qui sont indépendantes de la taille du graphe. Cette métrique d'évaluation est calculée à l'aide de l'équation suivante :

$$valeur\ normalisée = protection * \frac{n}{|DS|}.$$

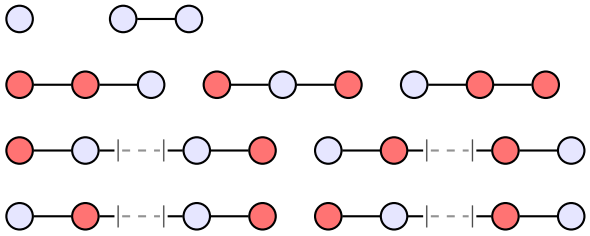


FIGURE 3.9 – Illustration des chemins sécurisés.

L'étape de normalisation de la valeur de protection est nécessaire afin de considérer la taille de l'ensemble dominant comparé à celle du graphe. En effet, en absence de cette considération, un ensemble dominant comportant l'ensemble des sommets du graphe offrirait une protection de 100%. Cela serait contre-intuitif avec nos désirs de réduire la taille de l'ensemble dominant.

Reprenons la Figure 3.6, dans laquelle nous avions déterminé le positionnement de l'ensemble non optimal car en dehors du flux d'information. D'après nos métriques d'évaluation, la protection offerte par cet ensemble dominant sur le graphe de la Figure 3.6 est de 53%. En utilisant notre méthodologie d'identification d'ensemble dominant central avec la betweenness centrality, nous obtenons, sur le même graphe, une protection de 100%. La Figure 3.10 illustre l'ensemble dominant identifié à l'aide de notre méthodologie. On remarque que ce dernier est effectivement positionné en travers du flux d'information contrairement à celui de la Figure 3.6.

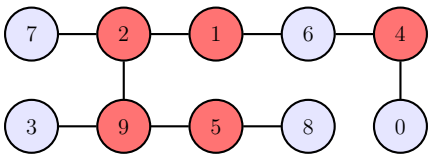


FIGURE 3.10 – Ensemble dominant (rouge) identifié avec notre méthodologie sur le graphe de la Figure 3.6.

3.5.5 Résultats

Supposons que l'information transite de manière optimale dans le graphe, c'est-à-dire selon le chemin le plus court entre deux sommets, l'utilisation de la betweenness centrality semble être la plus appropriée pour le placement de nos services de sécurité et donc également pour notre exemple de ersatz de chiffrement bout-en-bout.

L'Algorithme 2 présenté en Section 3.4.2 a été appliqué plusieurs fois sur des listes de sommets dont l'ordre changeait à chaque itération. Par la suite, l'ensemble dominant identifié a été analysé et évalué. Lors de nos expérimentations, nous nous sommes principalement focalisés sur l'impact de l'ordre des sommets sur l'ensemble dominant résultat de l'algorithme. Pour ce faire, nous avons comparé les quatre mesures de centralité principales à savoir la closeness (CC), eigenvector (EC), degree (DC) et la betweenness (BC). Pour tester l'impact des mesures de centralité sur l'ensemble dominant, nous avons effectué des tests avec plusieurs listes de sommets

ordonnés de manière aléatoire (R). Pour rappel, notre algorithme favorise la sélection des sommets situés en fin de la liste fournie en entrée, cette dernière a donc été ordonnée par ordre croissant de centralité.

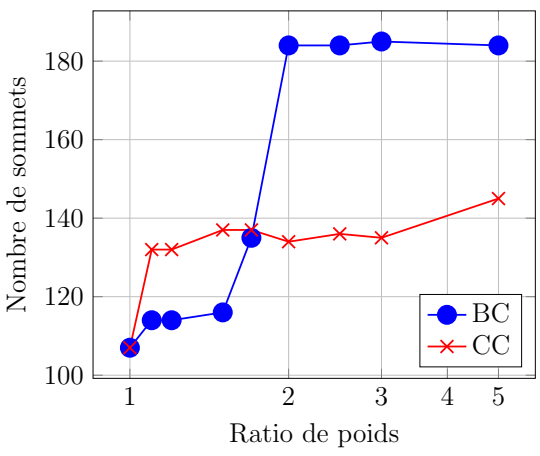
Les ensembles dominants retournés par l'algorithme ont ensuite été analysés selon nos critères d'évaluation définis en Section 3.5.4 (taille, qualité, protection de l'ensemble dominant et protection par sommet). La comparaison avec des listes ordonnées de manière aléatoire met en évidence l'importance de l'utilisation de centralité de graphe dans notre approche.

Les résultats de nos comparaisons sont reportés dans le tableau Table 3.3. La Figure 3.12 complète ces informations en fournissant un aperçu des catégories des objets ainsi que leurs proportions dans l'ensemble dominant.

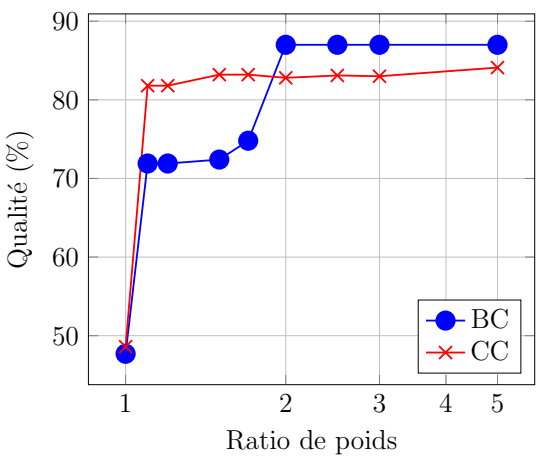
Nous avons analysé les effets du changement de ratio de poids dans la pondération d'arêtes de notre jeu de données entre la betweenness (BC) et la closeness centrality (CC). Les résultats sont représentés sur la Figure 3.11 où chaque courbe correspond à un critère d'évaluation en fonction du ratio entre le poids des sommets à forte priorité et ceux à basse priorité : L/H.

TABLE 3.3 – Comparaison des résultats d'ensemble dominant centraux en fonction de l'ordre de la liste des sommets (ordonnée par centralités et aléatoire).

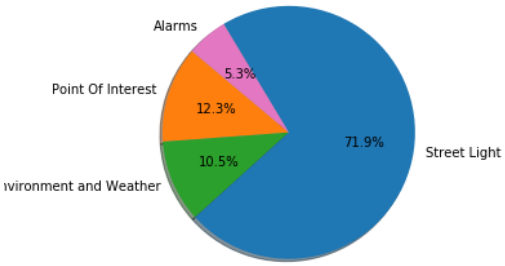
Graphe	Mesure	**Centralitiés**				**Aléatoire**	
		BC	*CC*	*DC*	*EC*	*R* seed=82	*R* seed=45
Dataset IoT non pondéré (1 458 sommets)	Taille DS	107 (7.33%)	107 (7.33%)	155 (10.63%)	176 (12.07%)	191 (13.10%)	212 (14.54%)
	Protection (%)	100	99.92	99.95	99.91	90.87	99.16
	Protection normalisée	13.63	13.62	9.402	8.277	6.937	6.820
	Qualité	47.7%	48.6%	44.5%	46.0%	51.3%	52.4%
Dataset IoT pondéré (1 458 sommets) ratio poids : 1.2	Taille DS	114 (7.819%)	132 (9.053%)	155 (10.63%)	176 (12.07%)	191 (13.10%)	212 (14.54%)
	Protection (%)	99.79	99.79	99.95	99.91	90.87	99.16
	Protection normalisée	12.76	11.02	9.402	8.277	6.937	6.820
	Qualité	71.9%	81.8%	44.5%	46.0%	51.3%	52.4%



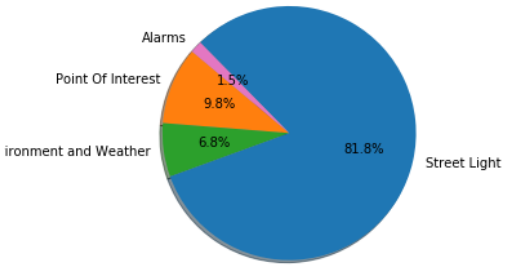
(a) Taille



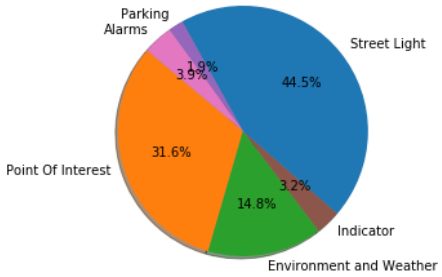
(b) Qualité



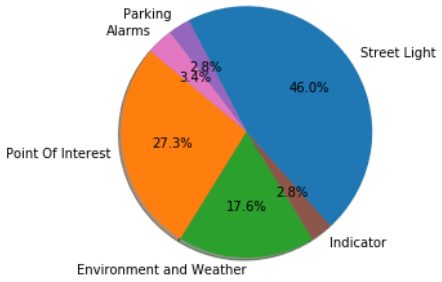
(a) Betweenness (114 sommets)



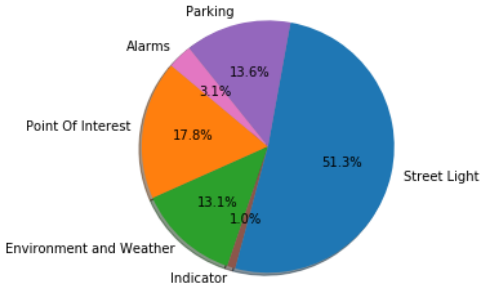
(b) Closeness (132 sommets)



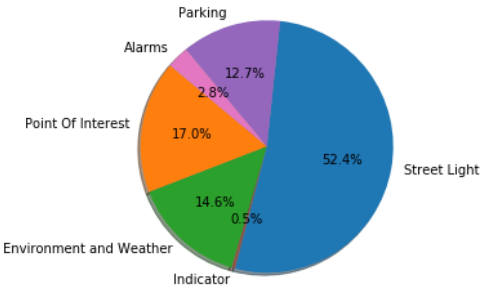
(c) Degree (155 sommets)



(d) Eigenvector (176 sommets)



(e) Aléatoire (seed=82) (191 sommets)



(f) Aléatoire (seed=45) (212 sommets)

FIGURE 3.12 – Qualité des ensembles dominants centraux sur notre dataset pondérée (ratio de poids : 1.2).

3.5.6 Analyse sur notre dataset IoT

Les résultats de notre étude indiquent que l'utilisation des mesures de centralité de graphe, en particulier la betweenness et closeness centrality produit des ensembles

dominants de plus petite taille. En comparant les ensembles dominants ainsi obtenus, on remarque, d'après nos critères d'évaluation, une augmentation générale des performances avec l'utilisation des mesures de centralité.

La Table 3.3 indique uniquement deux agencements aléatoires des sommets pour des raisons de lisibilité des résultats. Lors de nos études, nous avons effectué des tests sur un plus grand nombre d'ordres aléatoires retournant de manière générale des résultats similaires à ceux illustrés dans le tableau. Les performances de ces ensembles dominants varient entre 180 à 210 sommets avec des indices de protection allant de 81% jusqu'à 99,91%.

Les résultats de notre étude mettent en évidence l'intérêt de l'utilisation des centralités dans notre approche. En effet, nous constatons des tailles presque deux fois moins importantes pour les ensembles dominants utilisant la betweenness et la closeness centrality comparée à ceux utilisant des listes de sommets ordonnés par ordre aléatoire.

Le gain ne se limite pas au nombre de sommets des ensembles dominants, il est également distinguable au travers le positionnement de l'ensemble dominant.

Notre première constatation est que la majorité des résultats du tableau Table 3.3 révèle une protection avoisinant les 100% bien que cette valeur ne soit atteinte que pour la betweenness centrality sur le graphe non pondéré. Cette observation nécessite d'être mise en perspective avec le nombre de sommets des ensembles dominants. En effet, un ensemble dominant comportant un grand nombre de sommets aura tendance à offrir une valeur de protection forte. Comme décrit lors de la définition de nos métriques d'évaluation, il est important de normaliser cette valeur en fonction du nombre de sommets des ensembles dominants afin d'avoir un indice de comparaison plus précis.

L'analyse de l'indice de protection par sommet nous offre plus d'information de comparaison sur les ensembles dominants identifiés. On remarque ainsi, sur le graphe non pondéré que l'utilisation de la betweenness centrality retourne un indice presque deux fois supérieur que ceux issus des listes aléatoires. Ce critère d'évaluation met également en évidence la différence entre la betweenness et la closeness centrality avec les autres mesures de centralités avec des gains supérieurs d'environ un tiers comparé à la meilleure des autres centralités, la degree centrality.

Au vu de ces premiers critères d'évaluation, nous pouvons faire l'hypothèse que la betweenness centrality et la closeness centrality offrent des résultats similaires. On remarque cependant que 18 sommets ne sont pas partagés entre ces deux résultats. Leurs similarités reposent principalement sur la méthode de calcul des centralités à savoir, l'utilisation des chemins les plus courts.

Nos premières observations sur les résultats obtenus sur le dataset non pondérées peuvent être reportées sur le dataset pondérés. Comme supposé, on remarque, une légère augmentation de la taille des ensembles dominants causée par l'ajout de contraintes supplémentaires dans le processus de sélection.

On distingue également une diminution des indices de protection ce qui signifie que moins de chemins les plus courts dans le graphe sont considérés sécurisés (illustrés en Figure 3.9). Il est important de préciser que cela ne signifie pas que notre graphe n'est plus totalement sécurisé, cela signifie simplement qu'un coût (sous la forme de sauts

dans le graphe) supplémentaire sera présent pour la sécurisation des communications. Notre dataset est constitué de 1 458 sommets, de 35 657 arêtes formant ainsi 1 062 153 chemins les plus courts, donc communications potentielles. Avec l'utilisation de notre méthodologie seule 2 275 de ces communications vont subir un coût additionnel pour leur sécurisation ce qui représente 0.2% des communications du réseau.

L'augmentation de la qualité des ensembles dominants est particulièrement intéressante dans l'optique de réduire les frais inhérents au déploiement de services de sécurité dans un réseau IoT. Pour rappel, nous avons défini la qualité d'un ensemble dominant par la proportion d'appareils en mesure d'héberger un service de sécurité dans l'ensemble dominant. Le déploiement de services de sécurité sur cette catégorie d'objets pourrait se faire par l'intermédiaire d'une simple mise à jour. En supposant qu'une mise à jour logiciel soit moins onéreuse que la modification ou le remplacement d'un appareil, augmenter la qualité de l'ensemble dominant réduit fortement les frais liés au déploiement de services de sécurité.

À noter que dans le cas de notre jeu de données, il ne serait pas possible d'atteindre une qualité de 100%, car cela signifierait n'avoir que des noeuds correspondants à des objets de type "street light". Or, il se trouve que l'ensemble de ces objets, au nombre de 506, ne constitueraient pas un ensemble dominant du graphe. La pondération appliquée permet ainsi d'accroitre la qualité de l'ensemble dominant tout en limitant fortement le nombre de sommets dans cet ensemble tout en réduisant les coûts de communications potentiels.

L'utilisation de la pondération est intéressant ; cependant il est important de rappeler que cela dépend fortement des poids affectés. La Figure 3.11 démontre l'impact du ratio de poids (L/H) sur le nombre de sommets et la qualité de l'ensemble dominant. Lors de notre étude nous avons déterminé expérimentalement que les valeurs qui retournaient les meilleurs résultats étaient : $H = 1$ et $L = 1.2$. Si ces travaux étaient reproduits sur un autre jeu de données, ces valeurs seraient très probablement différentes. La Figure 3.11 met d'ailleurs en évidence l'importance d'une sélection minutieuse de ces valeurs.

Notre étude et le cas d'usage présenté, un ersatz de chiffrement bout-en-bout, démontrent l'apport de la betwenness centrality dans l'identification de nos ensembles dominants centraux. La betweenness centrality met en évidence les sommets par lesquels l'information transite dans le graphe ce qui correspond à nos exigences. Les autres mesures de centralité ne sont cependant pas à négliger. La closeness centrality définie l'importance d'un sommet à sa capacité à diffuser de l'information le plus rapidement possible dans un graphe. Cela signifie que si la vitesse de diffusion d'une information dans un réseau IoT était un critère de positionnement pour un service de sécurité, cette mesure de centralité serait plus appropriée.

Nous avons fait le choix de séparer l'algorithme déterminant un ensemble dominant, celui mesurant les centralités et celui donnant une pondération des arêtes afin d'offrir plus de souplesse aux utilisateurs dans leurs choix de positionnement.

3.5.7 Expérimentations sur des graphes de réseaux sociaux

Nous avons décrit en Section 3.5.1 les difficultés d'obtenir un dataset adapté à notre étude. N'ayant pas trouvé de second dataset pour valider nos études, nous avons analysé les caractéristiques de notre graphe afin d'identifier des graphes ayant les mêmes caractéristiques et corroborer nos résultats.

3.5.7.1 Analyse de notre graphe IoT

Afin d'identifier des graphes similaires au notre, nous devons dans un premier temps décrire ses caractéristiques. Nous nous sommes focalisés sur les trois caractéristiques suivantes : la densité, le diamètre et sa distribution des degrés. Ces résultats sont rapportés dans le tableau Table 3.4.

Densité

La densité d'un graphe $G = (V, E)$ correspond au ratio du nombre d'arêtes par rapport à son nombre de paires de sommets. Cette valeur varie entre 0, chaque sommet est isolé, à 1, le graphe est complet. La densité d'un graphe non orienté se calcul à l'aide de la formule suivante où n représente le nombre de sommets de G ($|V|$) et $|E|$ les nombres d'arêtes de G.

$$densité(G) = \frac{|E|}{\binom{n}{2}}.$$

Notre graphe IoT possède une densité faible avec une valeur de 0.03357.

Diamètre

Le diamètre d'un graphe mesure la longueur maximum des chemins les plus courts entre les sommets d'un graphe. Dans notre cas, cette valeur est de 5 ce qui est faible étant donné le nombre de sommets du graphe (1 458) ; on observe ce qui est appelé un phénomène de "petit-monde" (ou *small-world*)[209].

Distribution des degrés

Le degré d'un sommet correspond à son nombre d'arêtes incidentes. La Figure 3.13 trace la distribution des degrés de notre graphe. On remarque que cette courbe possède un aspect similaire à une loi de puissance, un grand nombre de sommets possèdent un degré faible alors qu'un très faible nombre de sommets bénéficient d'un degré élevé.

Afin de déterminer si cette distribution suit réellement une loi de puissance, une étude mathématique serait nécessaire [84, 243]. Dans notre cas, il est difficile de déterminer si cette distribution est vraiment représentative d'un réseau IoT, car nous n'avons pas suffisamment de données pour faire une généralisation.

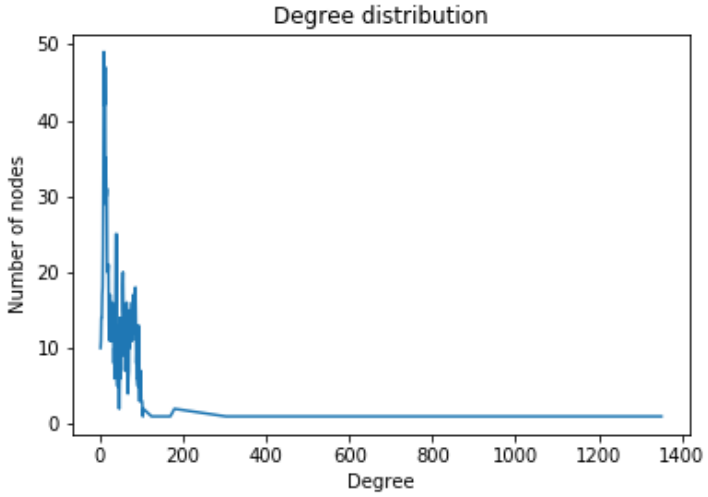


FIGURE 3.13 – Distribution des degrés du graphe IoT

En se focalisant sur les caractéristiques de notre graphe IoT, on remarque qu'il possède des caractéristiques similaires avec ceux des réseaux sociaux [88].

Notre idée est d'utiliser les graphes de réseaux sociaux, en raison de leurs fortes disponibilités, en remplacement de graphes IoT pour valider notre méthodologie.

3.5.7.2 Comparaison avec graphes de réseaux sociaux

Les graphes de réseaux sociaux ne faisant pas référence à des objets IoT, il est difficile d'en déduire des notions de priorité pour représenter leurs caractéristiques. Dans cette étude, nous nous sommes contentés de valider la première partie de notre approche, à savoir, celle reposant sur la topologie d'un graphe non pondéré.

Nous avons comparé les caractéristiques de notre graphe IoT au graphe de réseau social Ego-Facebook [182]. Les résultats ainsi obtenus sont reportés dans le tableau Table 3.4.

TABLE 3.4 – Comparaison des propriétés de graphes entre notre graphe IoT et Ego-Facebook [182].

	Graphe IoT	Ego-Facebook
Densité	0.03357	0.01082
Diamètre	5	8

On observe que ces deux graphes bénéficient effectivement de caractéristiques similaires. Malgré une densité trois fois supérieure du graphe IoT par rapport à Ego-Facebook, ces valeurs restent très faibles et reflètent le côté sparse de ces graphes (faible nombre d'arêtes par rapport au nombre de noeuds).

Les deux graphes révèlent également de petits diamètres par rapport à leurs nombres de sommets validant ainsi la propriété de petit-monde.

Finalement, en comparant les courbes de distribution des degrés des deux graphes (Figure 3.13 pour le graphe IoT et Figure 3.14 pour Ego-Facebook), on remarque qu'elles semblent suivre une distribution similaire avec un aspect de loi de puissance. Comme abordé dans la section précédente, on ne peut affirmer que ces distributions suivent réellement une loi de puissance sans une analyse statistique plus approfondie.

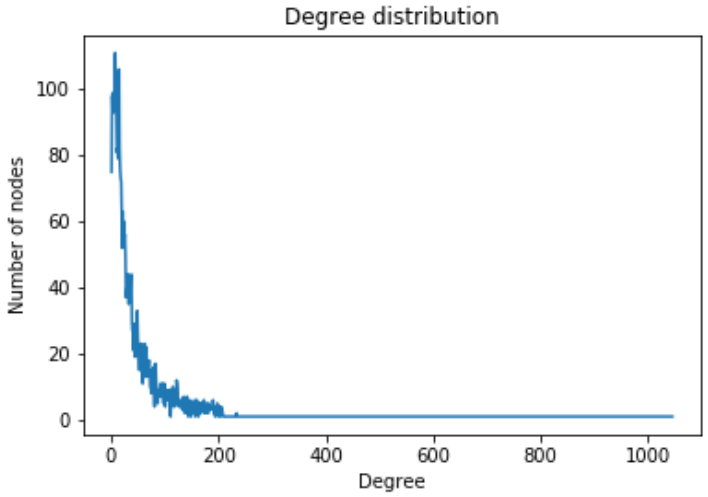


FIGURE 3.14 – Distribution des degrés du graphe Ego-Facebook [182].

En raison des fortes similitudes entre nos deux graphes, nous vérifions si les observations effectuées lors de notre étude sur le graphe IoT se retrouvent également sur le graphe Ego-Facebook.

3.5.7.3 Ensemble dominant central sur graphes de réseaux sociaux

Nous proposons de valider les résultats de notre méthodologie obtenus à la Section 3.5.5 en utilisant un graphe de réseau social (Ego-Facebook) bénéficiant de caractéristiques similaires. Les sommets du graphe Ego-Facebook représentent des individus, il n'est donc pas possible de faire une association avec la méthode de pondération abordée à la Section 3.5.2 consistant à identifier et privilégier les sommets capables d'accueillir les services de sécurité. Dans cette analyse, seule la première partie de nos travaux sera utilisée pour la comparaison concernant l'utilisation de topologie du réseau.

Nous avons donc appliqué notre méthodologie au graphe Ego-Facebook selon les mêmes critères qui ont été appliqués pour la Section 3.5.5. Nous avons évalué les ensembles dominants selon plusieurs ordres de listes de sommets, avec centralités et avec ordre aléatoires. Notons que lors de l'utilisation d'aléa, nous avons conservé les mêmes seeds afin de garantir la reproductibilité des résultats. En raison de l'absence de pondération, l'évaluation de la qualité des ensembles dominants n'était pas pertinente à représenter. Le tableau Table 3.5 illustre les résultats cette comparaison.

Malgré la différence d'origines entre notre graphe IoT et le graphe de réseau social Ego-Facebook, les résultats obtenus Table 3.5 montrent une performance de

Table 3.5 – Comparaison des ensembles dominants centraux entre notre graphe IoT et Ego-Facebook [182].

Graphe	Métrique	Centralités				Aléatoire	
		BC	CC	DC	EC	R seed=82	R seed=45
IoT (1458 sommets)	Taille DS	107 (7.33%)	107 (7.33%)	155 (10.63%)	176 (12.07%)	191 (13.10%)	212 (14.54%)
	Protection (%)	100	99.92	99.95	99.91	90.87	99.16
	Protection normalisée	13.63	13.62	9.402	8.277	6.937	6.820
Ego-Facebook (4039 sommets)	Taille DS	475 (11.76%)	501 (12.40%)	480 (11.88%)	519 (12.85%)	633 (15.67%)	669 (16.56%)
	Protection (%)	99.89	99.43	97.71	99.45	41.96	66.99
	Protection normalisée	8.494	8.016	8.222	7.740	2.677	4.044

notre méthodologie similaire. L'utilisation des mesures de centralité réduit la taille des ensembles dominants. Cette constatation est également présente pour les autres métriques d'évaluation : la protection générale et la protection normalisée sont plus importantes avec l'utilisation de centralité que lorsque les sommets sont ordonnés de manière aléatoire. L'obtention de résultats similaires dans les deux graphes analysés valide notre hypothèse de départ suivant laquelle l'utilisation de la betweenness centrality offre des performances intéressantes pour le placement de services de sécurité.

3.6 Conclusion

Au sein de cette contribution, nous avons proposé une stratégie de déploiement de services de sécurité dans un réseau IoT qui minimise les frais de déploiement. Nous avons représenté notre réseau IoT sous la forme d'un graphe et traduit nos contraintes de placement sous la forme d'un problème de graphe, à savoir : l'identification d'un ensemble dominant favorisant les sommets dont le poids est important. Pour déterminer le poids des sommets, nous avons eu recours aux mesures de centralités et avons démontré que leur utilisation, en particulier la betweenness centrality, réduit

fortement la taille de l'ensemble dominant tout en permettant d'influer sur cette mesure par l'intermédiaire de la pondération.

La capacité d'un objet à recevoir un service de sécurité a également été modélisée à l'aide de valeurs de priorité sur les sommets afin d'accroitre la qualité des ensembles dominants identifiés. Nous démontrons qu'une classification en deux catégories : priorité forte et faible, nécessite une attention particulière dans le choix des valeurs et qu'elles dépendant fortement du jeu de données utilisé. L'introduction de catégories supplémentaires pourrait, à défaut d'offrir de meilleures performances, faire bénéficier de plus de souplesse dans le processus de positionnement. Cependant, le choix des valeurs de priorité serait exponentiellement plus difficile à effectuer.

Nous démontrons également que l'ajout d'une pondération relative aux capacités des objets augmente significativement la qualité de l'ensemble dominant sans pour autant compromettre sa taille ce qui a pour effet de minimiser les frais de déploiement des services de sécurité. Notre solution présente des résultats intéressants avec un impact minime sur le transit de l'information dans le graphe en supposant que l'information circule de manière optimale.

Par la suite, il serait intéressant d'explorer les différentes techniques de pondération disponibles. Les travaux de Doriguzzi-Corin et al. proposent une méthode de pondération qui pourrait être pertinente à explorer [97]. Il serait également intéressant de pouvoir comparer nos résultats avec d'autres méthodes de placement de services sur des jeux de données similaires.

Chapitre 4

Comparaison des solutions de placement de services pour la sécurité

Dans ce chapitre, la seconde contribution de la thèse est présentée. Elle consiste à décrire l'apport qu'offre une approche sémantique dans la généralisation des problèmes de positionnement de services dans l'IoT ainsi que dans la comparaison de leurs solutions. Deux méthodologies basées sur l'utilisation d'une ontologie sont présentées, l'une portant sur la résolution d'un problème de placement, la seconde décrit un processus de comparaison des solutions de placement offrant la possibilité de faire varier les algorithmes de résolution ainsi que les critères d'évaluation des solutions. Finalement, un squelette de structure sémantique accompagné d'un outil de comparaison des solutions est proposé. Il en résulte que l'utilisation d'une structure sémantique offre des perspectives intéressantes en faisant toutefois attention à ne pas modéliser des infrastructures très volumineuses.

Sommaire

4.1 Introduction 80
4.2 Formalisation d'un problème de placement 82
4.3 Présentation des ontologies 87
4.4 Méthodologie de comparaison 95
4.5 Illustration sur un cas concret 104
4.6 Conception d'un outil de Comparaison de Solutions de Placement de Services (CSPS) 108
4.7 Conclusion et perspectives de poursuite 112

4.1 Introduction

L'essor des paradigmes de edge computing, et plus particulièrement du fog computing, entraine l'apparition de nouveaux verrous scientifiques. Parmi ces derniers, nous retrouvons particulièrement les problèmes de placement de services en bordure du réseau (fog, mist ou encore MEC).

Nous avons discuté, en Section 3.2, de ces problèmes et mis en évidence la grande diversité qu'il existe dans la description des problèmes de placement de services [76]. En effet, nous avons montré, en nous reposant sur les travaux de Salaht et al., que la description d'un problème de placement de services dans le fog nécessite la modélisation des trois modèles suivants : l'infrastructure, l'application et le déploiement [229]. Étonnamment, ces modèles subissent de grandes variations entre les différents travaux sur le positionnement de services.

En Section 3.2.1, nous avons souligné l'existence d'une relation de dépendance entre le modèle d'infrastructure et le réseau IoT alors que les modèles d'application et de déploiement sont quant à eux, fortement liés aux services à placer dans le réseau. Le fait que les travaux sur le positionnement de services n'utilisent pas de réseau IoT commun dans leurs travaux implique une variation importante du modèle d'infrastructure dans la définition de leur problème de placement.

4.1.1 Pourquoi comparer les solutions de placement de services ?

La forte diversité des problèmes de placement de services interroge sur la pertinence à vouloir comparer leurs solutions.

Lors de notre étude sur les problèmes de placement de services en Section 3.2, nous avons relevé que l'ensemble des travaux dans ce domaine suivent la même structure.

Un réseau représentatif de l'IoT est modélisé pour former le modèle d'infrastructure qui recense les ressources (appareils physiques) et caractéristiques (CPU, RAM, latence, bande passante ...) disponibles. Un premier niveau de variation entre les travaux est alors introduit à cette étape. En effet, lors de notre étude, nous n'avons pas identifié de travaux reposant exactement sur le même modèle d'infrastructure.

Les raisons d'une comparaison des solutions de placement de services prennent plus de sens aux regards de la diversité d'algorithmes de résolution utilisés par la communauté. Nous avons mis en évidence en Section 3.2.5, en nous reposant sur les travaux de Brogi et al. [76], que ces algorithmes pouvaient être catégorisés selon trois catégories : les approches mathématiques avec notamment l'ILP, les algorithmes de recherche et heuristiques, mais également des algorithmes plus exotiques à base d'apprentissage ou de backtracking.

Comparer les solutions de placement de services pourrait alors permettre à la communauté scientifique de positionner ses travaux les uns par rapport aux autres. Cette mise en relation pourrait également aider à l'identification des modèles de déploiement les plus appropriés. Actuellement, chaque problème de placement est accompagné d'un ensemble de contraintes et de fonctions objectives liées aux

applications, il serait intéressant de voir leurs effets sur les solutions retournées en ajustant ces paramètres.

Finalement, le plus grand intérêt d'une comparaison entre les solutions de placement de services consiste à identifier les algorithmes les plus adaptés à leurs résolutions. Doriguzzi-Corin et al. ont mis en évidence, dans leur résolution du problème de placement de services de sécurité, qu'il existait une différence entre une approche basée sur MILP et une autre utilisant un parcours heuristique [97].

4.1.2 Notre approche

Comparer des solutions de placement de services soulève un verrou scientifique majeur. Ce dernier concerne la modélisation des problèmes auxquels ces solutions répondent. Afin de proposer une comparaison pertinente, il est nécessaire de bénéficier d'une représentation commune à l'ensemble de ces problèmes. Dans ce chapitre, nous souhaitons généraliser les problèmes de placement de services dans une structure commune et déterminer quel est l'apport d'une telle structure sur la comparaison de leurs solutions.

Pour répondre à ces interrogations, nous formalisons ce que sont les problèmes de placement de services et proposons d'utiliser une structure sémantique pour leur définition. L'intérêt de cette structure est de représenter les trois modèles dont la description d'un problème de placement dépend, à savoir : l'infrastructure, l'application et le déploiement (Illustrés sur la Figure 3.1).

Notre objectif lors de la création de cette structure est de couvrir un maximum de descriptions de problèmes de placement possible. L'utilisation des ontologies semble particulièrement bien adaptée pour la construction de cette représentation en raison des nombreuses modélisations IoT effectuées avec ces outils. Procéder de la sorte contribue à une meilleure coopération entre notre représentation et les modélisations IoT existantes dans la communauté scientifique.

Une fois la structure mise en place, nous définissons, dans un premier temps, les problèmes de placement que nous avons proposés dans notre contribution au Chapitre 3 ainsi que les algorithmes utilisés dans l'ontologie. Nous obtenons alors une structure semblable à celle illustrée en Figure 4.1 et qui correspond à une base de connaissance.

L'ontologie agit comme une base de connaissance relationnelle. Nous sommes alors en mesure de construire un outil de comparaison de solutions de placement en se reposant sur cette structure. Notre objectif est, depuis une infrastructure et un service donnés, de comparer des solutions de positionnement en faisant varier les algorithmes de résolution, et les méthodes d'évaluation. La Figure 4.2 illustre ces propos en montrant l'utilisation de la structure sémantique pour la résolution des problèmes de placement ainsi que pour la comparaison de leurs résultats.

Notre outil, intitulé CSPS (Comparaison de Solutions de Placement de Services), ne permet pas uniquement de déterminer quels sont les meilleurs algorithmes de placement. En effet, en mettant en relation les solutions des algorithmes de placement avec divers fonctions objectives, nous sommes également en mesure d'identifier quels sont les critères d'évaluation les plus discriminant. L'utilisateur de CSPS peut ainsi

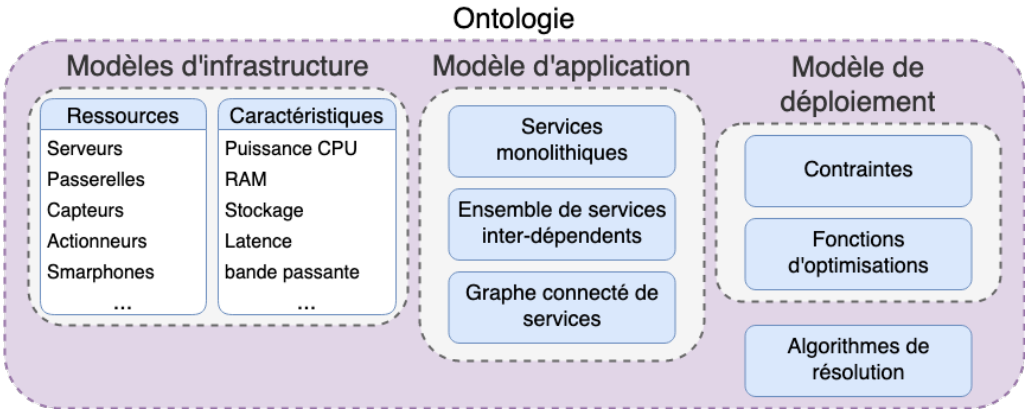


FIGURE 4.1 – Illustration des concepts à définir dans notre ontologie.

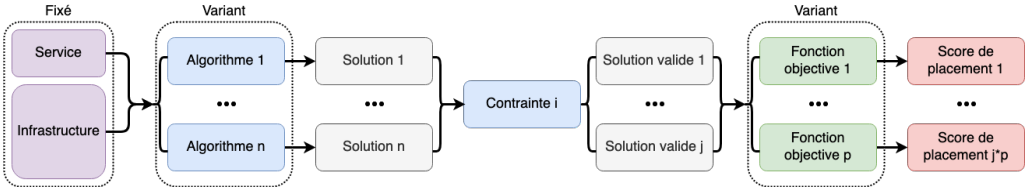


FIGURE 4.2 – Principe et enchainement des éléments de l'ontologie pour l'identification et la comparaison des solutions de placement de services.

évaluer les solutions de différents algorithmes de placement de services selon divers critères.

4.2 Formalisation d'un problème de placement

Un problème de placement de service de sécurité dans l'IoT consiste à identifier sur quels objets de l'IoT un service de sécurité doit être déployé étant donné certaines conditions.

4.2.1 Modélisation d'un réseau IoT

La modélisation d'un réseau IoT est primordiale pour formaliser les problèmes de placement de services dans l'IoT. Au chapitre précédent, nous avions modélisé de tels réseaux sous la forme de graphes. Bien que cette structure soit particulièrement intéressante de par sa capacité à être pondérée, elle possède des limites. En effet, même si il est possible de faire évoluer la structure sous la forme d'un multi-graphe, cela reste insuffisant pour de nombreux problèmes. Une alternative consiste à se reposer sur une même structure issue de la théorie des modèles finis. Nous appellerons cette structure $\mathcal{M} = (\mathcal{D}, \mathcal{R})$ dans laquelle $\mathcal{D}$ représente le domaine, il comprend l'ensemble des éléments de la structure ; dans notre cas cela comprend tous les objets de l'IoT. Le symbole $\mathcal{R}$ correspond au vocabulaire, il contient les relations et fonctions agissant sur les éléments de $\mathcal{D}$ (les objets de l'IoT). $\mathcal{R}$ peut également contenir des

constantes (fonctions d'arité zéro) ainsi que des relations ($=$, $<$ et $>$) pour comparer des valeurs à ces constantes. On parle de $\mathcal{R}$-structure pour préciser le vocabulaire lorsque plusieurs vocabulaires sont possibles.

L'utilisation d'une telle structure offre un grande souplesse en n'imposant pas de limitations dans la quantité d'informations que l'on peut y intégrer. À titre d'exemple, nous pouvons y décrire les fonctions unaires $W : \mathcal{D} \to \mathbb{R}$ qui retourne la puissance de calcul d'un objet, $M : \mathcal{D} \to \mathbb{R}$ qui fournit la mémoire résiduelle d'un objet ou encore $SE : \mathcal{D} \to \{0, 1\}$ qui informe sur la présence d'un élément de sécurité sur un objet spécifique. La structure ne limite pas l'arité des fonctions qui y sont décrites. Ainsi, nous pouvons également décrire des fonctions binaires telles que $B : \mathcal{D}^2 \to \mathbb{R}$ ou encore $L : \mathcal{D}^2 \to \mathbb{R}$ qui renvoient respectivement la bande passante et la latence entre deux objets.

4.2.2 Modélisation d'un placement de service

Définissons les services par des relations unaires, n'appartenant pas à $\mathcal{R}$, que nous appellerons variables unaires. Soit $\mathcal{S}$ l'ensemble de ces variables unaires pour un problème de placement de service fixé, un problème de placement de service de sécurité dans l'IoT consiste à identifier ces variables unaires. Ce problème s'exprime à l'aide de la formule suivante, appartenant à la logique Monadique Existentielle du Second Ordre (MESO) et dans laquelle Φ est une formule du premier ordre

$$\exists \mathcal{S} \Phi(\mathcal{R}, \mathcal{S}).$$

En règle générale, Φ représente une conjonction de contraintes, définies en Section 4.2.2.1, que doivent respecter les services. Ces contraintes peuvent aller d'une simple vérification de capacités d'un objet à des conditions plus complexes regroupant de multiples contraintes. Quelques exemples de contraintes sont donnés dans la section suivante.

La résolution d'un problème de placement consiste à trouver une solution ($\mathcal{R} \cup \mathcal{S}$ $(\mathcal{M}, \mathcal{S}^*)$) qui vérifie la formule suivante dans laquelle les éléments de $\mathcal{S}^*$ constituent des valeurs (ou interprétations) de variables de $\mathcal{S}$

$$(\mathcal{M}, \mathcal{S}^*) \models \Phi(\mathcal{R}, \mathcal{S}).$$

Trouver une solution au problème de placement n'est souvent pas suffisant dans le cadre de l'IoT. En effet, ce problème est communément associé avec des contraintes d'optimisation pour par exemple réduire les coûts d'utilisation de ressources [245] ou encore minimiser les temps de réponses des services [147, 244, 267].

4.2.2.1 Définition d'une contrainte

Les contraintes sont des éléments essentiels pour le placement de services de sécurité. En effet, chaque service ayant des conditions de placement spécifiques, il est important de pouvoir les spécifier afin qu'une solution de placement puisse satisfaire ces conditions. D'un point de vue général, les contraintes sont les conditions qui

doivent être satisfaites pour qu'une solution de placement de service soit valide. Plus formellement, soit $(\mathcal{M}, \mathcal{S}^*)$ une $(\mathcal{M}, \mathcal{S})$-structure solution d'un problème de placement de service de sécurité, une contrainte C_i est une application d'un domaine $\mathcal{D}_i$ sous-ensemble de $(\mathcal{M}, \mathcal{S}^*)$ et retournant un booléen. Dans la suite de nos travaux, nous considérerons que les contraintes sont des formules du premier ordre. Nous offrons ainsi la possibilité de cumuler plusieurs contraintes lors du placement de multiples services ou de services plus complexes regroupant de multiples contraintes (tel qu'un emplacement géographique particulier, une puissance de calcul suffisante et la présence d'un composant matériel spécifique). Nous représentons ainsi une conjonction de contraintes à l'aide de la formule $\Phi(\mathcal{R}, \mathcal{S})$. Plusieurs exemples de contraintes sont abordés en Section 4.2.3 dans l'illustration d'un modèle-jouet.

4.2.2.2 Définition d'une fonction objective

Une fonction objective se représente sur la forme d'une fonction $\mathcal{O}_i : (\mathcal{M}, \mathcal{S}^*) \rightarrow \mathbb{R}$. Son rôle est de permettre la définition d'un problème d'optimisation en évaluant et comparant des solutions de positionnement. Lorsque plusieurs solutions sont disponibles, la fonction objective permet notamment de déterminer laquelle est la meilleure d'après ses critères d'évaluation. Il est alors possible de résoudre le problème d'optimisation en minimisant ou maximisant la valeur retournée par cette fonction. Dans le cas d'une minimisation, par exemple pour minimiser le temps de réponse moyen d'un service [147], étant donné deux solutions $(\mathcal{M}, \mathcal{S}^{*,1})$ et $(\mathcal{M}, \mathcal{S}^{*,2})$, $(\mathcal{M}, \mathcal{S}^{*,1})$ sera considéré comme la meilleure solution d'après la fonction $\mathcal{O}_i$ si $\mathcal{O}_i(\mathcal{M}, \mathcal{S}^{*,1}) < \mathcal{O}_i(\mathcal{M}, \mathcal{S}^{*,2})$. Dans le cas d'une maximisation, la meilleure solution sera celle pour laquelle la fonction objective aura retourné la valeur la plus grande.

Il se peut que dans certains cas, le simple fait d'avoir trouvé une solution est suffisant, et qu'il n'y ait alors pas de problème d'optimisation. C'est notamment le cas lors de l'application de correctifs de sécurité où le correctif doit être déployé sur l'ensemble des solutions retournées (appareils vulnérables dans le cas de cet exemple).

4.2.3 Illustration sur un modèle-jouet

Prenons deux services de sécurité S_1 et S_2 pour lesquels nous définissons les contraintes suivantes :

- C_1 : Chaque service est déployé sur un seul objet.
- C_2 : Le service S_1 requiert une puissance de 10 pour fonctionner.
- C_3 : Le service S_2 a besoin d'une mémoire résiduelle de 5.
- C_4 : Les deux services ne doivent pas être hébergés sur des objets qui interagissent ensemble.
- C_5 : Chaque service doit être relié à un objet avec une capacité de bande passante d'au moins 50.

Les valeurs numériques données constituent des constantes du problème et peuvent être modifiées selon les données réelles. Pareillement, les contraintes énumérées peuvent être changées pour correspondre aux services à positionner. Dans notre

exemple, les contraintes s'expriment sous la forme de formules du premier ordre suivantes dans lesquelles $E(x, y)$ est une relation binaire sur $\mathcal{D}$ indiquant si x et y sont connectés ensemble :

C_1 : $\exists x \forall y\ S_1(x) \wedge S_1(y) \rightarrow x = y.$
$\exists x \forall y\ S_2(x) \wedge S_2(y) \rightarrow x = y.$

C_2 : $\forall x\ S_1(x) \rightarrow W(x) \geq 10.$

C_3 : $\forall x\ S_2(x) \rightarrow M(x) \geq 5.$

C_4 : $\forall x \forall y\ (S_1(x) \wedge S_2(y)) \rightarrow \neg E(x, y).$

C_5 : $\forall x \exists y\ (S_1(x) \wedge E(x, y)) \rightarrow B(x, y) > 50.$
$\forall x \exists y\ (S_2(x) \wedge E(x, y)) \rightarrow B(x, y) > 50.$

Le problème de placement ainsi décrit est représenté par la formule MESO suivante :

$$\exists S_1 \exists S_2\ C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5.$$

Un solution de ce problème est donné par S_1^* et S_2^*, interprétations respectives S_1 et S_2, par la formule suivante :

$$(\mathcal{M}, \{S_1^*, S_2^*\}) \models C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5.$$

La description des problèmes de placement sous la forme de formule de logique MESO ne permet pas de déterminer la difficulté de ces problèmes. En effet, un grand nombre de problèmes exprimables dans cette logique sont faciles à résoudre notamment grâce à l'utilisation d'algorithmes basiques tels que les algorithmes gloutons. Cependant, cette logique permet également d'exprimer des problèmes bien plus complexes dont certains sont NP-complets. Décrire des problèmes de placement de la sorte n'offre peut être pas explicitement d'informations sur leurs difficultés mais cela offre un cadre formel pour leur comparaison.

4.2.4 Autres logiques pour la modélisation

Nous avons décrit, dans les sections précédents, l'utilisation de logique MESO pour l'expression des problèmes de placement de service. Cette dernière, bien que relativement intuitive pour adresser les problèmes de placement, peut ne pas être suffisamment expressive dans la description de certains problèmes.

Ce cas a été rencontré lors de la modélisation du problème de placement proposé par Donassolo et al. dans lequel chaque paire de services doit être reliée par un chemin fixé [96]. La formalisation du problème de cet article met au même niveau (c'est-à-dire dans $\mathcal{D}$) les nœuds, les arêtes et les chemins entre les nœuds. Étant donné la grande quantité de chemins possible, il est préférable de ne pas donner ces informations dans la structure. Dans un tel cas, il est nécessaire de pouvoir quantifier sur les arêtes pour fixer les chemins entre les services.

Soient S_i et S_j deux services et a et b les éléments de $\mathcal{D}$ sur lesquels sont déployés les services S_i et S_j. Un chemin entre a et b est codé par une relation binaire $R_{i,j}$ de la manière suivante :

$$\forall x \forall y \forall z \left((R_{i,j}(x,y) \wedge R_{i,j}(x,z)) \rightarrow y = z \right) \wedge \left((R_{i,j}(y,x) \wedge R_{i,j}(z,x)) \rightarrow y = z \right)$$
$$\left(\exists x \exists y \Big(E(a,x) \wedge R_{i,j}(a,x) \wedge E(y,b) \wedge R_{i,j}(y,b) \Big) \right) \wedge$$
$$\left(\forall x \Big(x \neq b \exists y R_{i,j}(y,x) \Big) \rightarrow \Big(\exists y R_{i,j}(x,y) \Big) \right)$$

La première ligne de cette formule décrit le codage d'un chemin par $R_{i,j}$, la seconde ligne exprime que a peut atteindre un sommet au travers $R_{i,j}$ et que b peut être atteint par un sommet par l'intermédiaire de $R_{i,j}$. La dernière partie exprime que si un sommet différent de b est atteint par $R_{i,j}$ alors il peut atteindre un autre sommet par $R_{i,j}$.

Nous obtenons finalement une variable de relation binaire $R_{i,j}$ pour chaque couple de services (S_i,S_j) de la sorte que pour n services, nous ayons :

$$\mathcal{S} = \{S_1, \ldots, S_n\} \cup \{R_{i,j} \mid (i,j) \in \{1, \ldots, n, i \neq j\}.$$

La formule $\exists \mathcal{S} \Phi(\mathcal{R}, \mathcal{S})$ appartient alors à la logique Existentielle du Second-Ordre (ESO). Le fait de pratiquer une quantification sur les chemins rend le problème beaucoup plus difficile et l'existence d'une solution plus incertaine. De ce fait, certains compromis peuvent être envisagés notamment sur les contraintes de placement à satisfaire.

Dans le cas de l'article de Donassolo et al. [96], les auteurs contournent cette difficulté en découpant leur structure (graphes) sous la forme de plusieurs composantes au sein desquels des solutions locales sont recherchées. Aborder ce problème de manière globale est beaucoup trop complexe en particulier dans le domaine de l'IoT où la taille de la structure atteint facilement plusieurs milliers d'objets.

4.2.5 Formalisation du problème abordé au Chapitre 3

Illustrons désormais ce formalisme des problèmes de placement de services de sécurité au problème introduit au Chapitre 3. Nous définissons ainsi une structure $\mathcal{M} = (\mathcal{D}, \mathcal{R})$ dans laquelle $\mathcal{D}$ représente les sommets du graphe de réseau IoT et $\mathcal{R} = \{E, C, P\}$. L'ensemble des arêtes du graphe est représenté par E alors que P et C, notées $P : \mathcal{D} \rightarrow \mathbb{R}$ et $C : \mathcal{D} \rightarrow \mathbb{R}$, correspondent aux fonctions qui fournissent respectivement les valeurs de priorité et centralité pour chaque sommet du graphe de réseau IoT. Pour rappel, dans le Chapitre 3, nous devions placer un service de chiffrement bout-en-bout sur un ensemble dominant du graphe de réseau IoT. Cela se traduit par la réduction de $\mathcal{S}$ en une seule relation unaire S. Nous pouvons alors exprimer la contrainte que S soit un ensemble dominant (*dominating set* ou DS) à l'aide de la formule :

$$\phi_{DS} = \forall x \notin S, \exists y \in S, E(x,y).$$

Nous avions par la suite introduit un problème d'optimisation en nécessitant que cet ensemble dominant soit de taille minimal (minimal local). Cette nouvelle contrainte peut alors s'exprimer à l'aide de la formule suivante :

$$\phi_{minimal} = \forall z \in S, \exists x \notin S, \forall y \in S \setminus \{z\}, \neg E(x, y).$$

Trouver une solution $(\mathcal{M}, \mathcal{S}^*)$ revient alors à vérifier $\phi_{DS} \wedge \phi_{minimal}$. Modélisons ϕ_{DS} et $\phi_{minimal}$ par les contraintes $C_1, C_2 : (\mathcal{D}, E) \rightarrow \{Vrai, Faux\}$. Cela signifie que trouver une solution $(\mathcal{M}, \mathcal{S}^*)$ revient à satisfaire la conjonction de ces contraintes $(C_1 \wedge C_2)$.

Afin de déterminer la solution la plus appropriée à la résolution de notre problème, nous avions introduit quatre fonctions objectives qui sont les suivantes.

— Taille de l'ensemble dominant :
$\mathcal{O}_1 : S^* \rightarrow |S^*|$

— Qualité :
$\mathcal{O}_2 : (P, S^*) \rightarrow \mathbb{R}$

— Protection :
La protection et la protection normalisée constituent des fonctions objectives qui reposent sur le calcul de l'ensemble des plus courts chemins dans le graphe. En raison de la taille exponentielle de cet ensemble par rapport au nombre de sommets du graphe, l'ensemble des chemin les plus courts ne sont pas intégrés à la structure de départ $(\mathcal{M} = (\mathcal{D}, \mathcal{R}))$. De manière plus générale, l'usage de ces fonctions objectives est discutable. En effet, le coût important du calcul de chemins les plus courts rendent ces fonctions peu appropriées à une utilisation lors du placement d'un service. La protection et sa version normalisée offrent cependant des informations très importantes sur les solutions de placement retournées par les algorithmes de placement. L'usage de ces fonctions pour l'évaluation des méthodes de placement semble donc plus approprié.

4.3 Présentation des ontologies

Il existe de nombreux outils permettant de modéliser la structure formelle que nous avons décrite en section précédente. Parmi ces outils, ceux qui reposent sur l'utilisation de structures sémantiques semblent particulièrement appropriés.

Depuis maintenant plusieurs années, les structures sémantiques sont de plus en plus utilisées en relation avec les technologies de l'IoT. La communauté scientifique a nommé ces structures SWoT, un acronyme qui désigne Semantic Web of Things [128]. Les SWoT sont communément représentées à l'aide d'ontologies.

Une ontologie est une description formelle et explicite de concepts dans un domaine précis. Plus spécifiquement dans le domaine de l'informatique, les ontologies constituent un moyen de modéliser formellement la structure d'un système, à savoir, les entités et les relations pertinentes qui émergent de son observation, et qui sont utiles à des objectifs spécifiques [126]. Les ontologies se constituent de classes qui

représentent les concepts, de propriétés qui décrivent leurs différents attributs et de restrictions sur ces propriétés [201]. Une ontologie associée à un ensemble d'instances individuelles de classes constitue une base de connaissance. Les ontologies pouvant contenir initialement plusieurs instances de classes, la séparation entre ontologie et base de connaissance est discutable.

4.3.1 Ontologies IoT existantes

Une ontologie IoT modélise formellement la structure d'un système IoT. En dehors de sa capacité de modélisation, une ontologie doit contribuer à l'interopérabilité des systèmes. Pour ce faire, il faut qu'elle soit facilement lisible et compréhensible aussi bien par un humain ainsi que par une machine. Sa structure doit être partagée, de préférence publiquement, mais également extensible afin de favoriser l'évolution son évolution. Nous entrons plus en détail sur ces notions en Section 4.3.3 où nous abordons la construction d'ontologies. Dans cette section, nous porterons notre attention sur les ontologies IoT les plus populaires et couramment utilisées.

L'ontologie la plus communément reprise dans les travaux sur le SWoT, est l'ontologie Semantic Sensor Network (SSN [1]). Cette ontologie décrit l'ensemble des concepts qui gravitent autour des capteurs et de leurs observations. L'ontologie SSN repose sur une seconde ontologie, SOSA [2] (Sensor, Observation, Sample, and Actuator), plus légère, qui décrit des classes et propriétés élémentaires. Malgré le fort niveau de détails apporté dans la conception de l'ontologie SSN, cette dernière couvre principalement les applications qui reposent sur les capteurs. Cette ontologie reste cependant une référence pour d'autres travaux dans lesquels certaines classes de SSN sont reprises. La classe "Device" de SSN est particulièrement intéressante dans sa définition puisqu'elle permet la description d'un "Device" comme une composée de plusieurs "Devices" plus petits.

L'organisme de standardisation des technologies M2M et de l'IoT, oneM2M propose une structure ontologique permettant aux autres ontologies de s'interfacer avec oneM2M [3]. L'objectif de cette ontologie est de permettre aux systèmes compatibles oneM2M d'interagir avec des systèmes extérieurs. Cette ontologie décrit certains concepts particulièrement intéressants dans notre contexte dont notamment les classes "Device" et "Service". En dépit de son fort potentiel descriptif, l'ontologie oneM2M rencontre des limitations pour la description de concepts extérieurs aux standards oneM2M.

Les travaux de Bermudez-Edo et al. ont mené à la conception de l'ontologie IoT-Lite, une ontologie allégée qui décrit de manière sémantique l'IoT [61]. En effet, l'ontologie IoT-Lite se compose de 149 axiomes alors que SSN comporte 572 et oneM2M 232. Dans leurs travaux, les auteurs ont démontré que la taille de l'ontologie, le nombre de concepts et de propriétés définis, avaient un impact significatif sur leurs utilisations. Cette constatation est d'autant plus importante dans un contexte IoT où les temps de traitement jouent un rôle primordial et où les capacités peuvent

1. http://www.w3.org/ns/ssn/
2. http://www.w3.org/ns/sosa/
3. https://git.onem2m.org/MAS/BaseOntology

être limitées. L'intérêt de cette ontologie repose sur sa légèreté, mais également sur les concepts définis. IoT-Lite reprend la définition de "Device" faite dans SSN et décrit également une classe "Service" avec les propriétés associées. La particularité de IoT-Lite réside dans sa capacité à décrire de manière très concise les différents concepts liés à l'IoT. Contrairement aux autres ontologies qui visent à apporter une modélisation la plus précise possible au détriment de leurs tailles, IoT-Lite propose une approche inverse en généralisant les concepts. De plus, sa légèreté et sa documentation en font une ontologie particulièrement compréhensible et donc propice à être réutilisé.

D'autres ontologies proposent des approches sous la forme de taxonomies. C'est le cas de l'ontologie M3 [130] qui est une extension de l'ontologie SSN et plus particulièrement du concept "ObservationValue" de SSN. Elle permet d'unifier les données des capteurs pour fournir une base aux raisonneurs en décrivant les concepts tels que le domaine d'intérêt, les phénomènes physiques ou encore les unités de mesure. En raison de sa taille importante, une version allégée de l'ontologie M3 a été proposée et intitulée M3-lite. Cette dernière possède les mêmes ambitions et décrit 60 types de capteurs, 78 phénomènes physiques, 64 unités de mesure ainsi que 12 domaines d'intérêts [43].

L'ontologie Fiesta-IoT [43] reprend plusieurs ontologies décrites précédemment, à savoir, SSN, IoT-Lite, et M3-Lite. Cette ontologie vise à regrouper les objets de l'IoT et à supporter leur interopérabilité ainsi qu'aux données qu'ils produisent. L'ontologie Fiesta-IoT se focalise principalement sur la donnée, elle s'applique à toutes les applications IoT qui nécessitent une annotation sémantique des données. Fiesta-IoT propose une approche descriptive approfondie de l'ensemble des concepts associés à l'annotation des données et englobe ainsi de nombreuses notions intéressantes. Cependant, contrairement à l'ontologie IoT-Lite, Fiesta-IoT définit 2433 axiomes contre 149 pour IoT-Lite. En dépit de sa qualité dans la description des concepts liés à l'IoT, et d'après nos besoins applicatifs, une ontologie plus légère est à privilégier. De plus, notre application n'étant pas axée principalement sur l'annotation de données, la pertinence d'utiliser l'ontologie Fiesta-IoT serait réduite.

Il existe une multitude d'autres structures ontologiques qui décrivent les concepts de l'IoT avec des degrés de détails très variés. L'ontologie IoT-O semble par exemple très intéressante en raison de sa composition sous la forme de cinq modules : *sensing*, *acting*, *lifecycle*, *service*, *energy*, qui regroupent au total neuf ontologies [236]. IoT-O décrit, dans son module *service* plusieurs concepts qui pourraient nous intéresser s'ils ne décrivaient pas principalement des services web. De plus, l'un des défauts majeurs de IoT-O réside dans sa disponibilité. En effet, au moment de la rédaction de ce texte, le fichier décrivant la structure ontologique n'est plus disponible ce qui porte à s'interroger sur la maintenance de cette ontologie. Comme de nombreuses bonnes pratiques le recommandent, et comme nous allons le mentionner par la suite en Section 4.3.3.1, il est préférable d'éviter les ontologies dont la disponibilité est compromise.

Afin de recenser et de classifier l'ensemble des projets IoT qui reposent sur l'utilisation de sémantique, Gyrard et al. ont proposé la plateforme LOV4IoT [128]. Cette dernière compte, à la date d'écriture de ces lignes, près de 590 projets de

recherche basés sur les ontologies IoT et leurs domaines d'applications. Nous avons décrit dans cette section les ontologies qui sont les plus couramment utilisées et les plus largement répandues.

4.3.2 Travaux de placement de services basés sur ontologies

L'allocation et plus particulièrement, le placement de services dépendent fortement de l'environnement dans lesquels ils sont effectués. De manière plus générale, ces applications bénéficient fortement des principes de *context awareness* qui décrivent la capacité d'un système à récolter de l'information sur son environnement pour adapter son comportement en conséquence.

En section 3.2, nous avons discuté des nombreux travaux sur le positionnement de services et avons constaté qu'il était préférable de bénéficier d'une connaissance la plus étendue possible sur l'infrastructure afin d'identifier l'emplacement d'un déploiement idéal. L'utilisation des ontologies permet notamment de décrire de manière formelle cette infrastructure et contribue ainsi à structurer les informations récoltées.

Les travaux de Tao et al. [253] illustrent l'apport d'une structure sémantique pour la représentation du modèle d'infrastructure. Dans leur papier, les auteurs présentent une architecture cloud à plusieurs couches. La couche haute correspond aux infrastructures de cloud public suivi des instances de cloud privées spécifiques à chaque vendeur et finalement, la couche la plus basse représente les objets de l'IoT. Malgré le fait que ces travaux se focalisent sur le cloud, l'analogie avec le fog computing est toujours possible grâce à cette structure en couches. L'étude faite par Tao et al. introduit une nouvelle ontologie composée de cinq catégories, *Security*, *Home_device*, *Environment*, *Data_communication* et *Entertainment*. Chacune de ces catégories se scinde par la suite en de nombreuses autres branches pour former une ontologie complexe qui décrit le domaine *smart home*. Bien que l'ontologie proposée soit intéressante, elle rencontre cependant plusieurs limites. Lors de sa conception, aucune autre ontologie n'a été réutilisée. De plus, l'ontologie ne semble pas être accessible publiquement ni répertoriée sur la plateforme LOV4IoT [128] rendant alors sa réutilisation et son utilisation avec de nouvelles ontologies difficile.

Les travaux de Nezami et al. [199] quant à eux, se basent IoT-O [236] qu'ils associent avec une seconde ontologie, Cloud-O [218] formant ainsi une troisième ontologie. Dans leur approche, les auteurs proposent d'utiliser les ontologies pour allouer des ressources dans le paradigme du edge computing. Les auteurs proposent une méthodologie composée de quatre étapes, la découverte des ressources, la modélisation, la sélection et l'allocation. Lors de la première étape, l'ensemble des noeuds de l'infrastructure se partagent leurs informations et leurs requêtes. Elles sont ensuite relayées à l'ensemble des objets du réseau et modélisées dans une ontologie partagée à l'ensemble des noeuds du réseau. La troisième étape consiste à déterminer quelles ressources allouer pour satisfaire les besoins décrits dans l'ontologie. Cette opération est effectuée par les noeuds fog au travers l'utilisation de règles sémantiques et logiques décrites avec le langage SWRL (Semantic Web Rule Language). Finalement, les répercutions de l'allocation de ressources sont propagées sur l'ensemble des autres noeuds du réseau afin de préserver la validité de leurs informations.

La méthodologie de Nezami et al. [199] propose une approche intéressante concernant l'allocation de ressources, cependant, comme les travaux de Tao et al., l'ontologie résultante de leur approche n'est pas mise à disposition. Les auteurs proposent d'exploiter les capacités de raisonnement inhérent à l'utilisation d'ontologie dans la sélection des ressources. Nous avons discuté en Section 3.2.5 de la multitude d'algorithmes utilisables pour la résolution des problèmes de placement dans le fog. L'utilisation de règles sémantiques en fait partie et il serait intéressant de pouvoir introduire plus de variété dans cette étape de sélection. Finalement, en dépit des limitations de leur approche, les auteurs proposent une méthodologie pouvant être reprise et améliorée pour adresser les problèmes de placement.

Parmi les méthodologies de la littérature qui reposent sur des structures sémantiques, les travaux de Petrovic et Tosic se démarquent particulièrement [205]. Dans leur étude, les auteurs décrivent un composant structurel, nommé SMADA-fog, qui permet la génération de code automatique pour la gestion d'infrastructure de fog computing. Ils démontrent l'apport d'une approche sémantique pour le traitement automatique de connaissances complexes. Ce traitement est ensuite appliqué à la génération de code automatisée pour le déploiement de services. L'ensemble de ces opérations reposent sur l'utilisation de méta-modèles. Les auteurs décrivent un méta-modèle comme une forme de modélisation définissant la structure et les contraintes d'une famille de modèles pour deux aspects différents du déploiement dans une architecture de fog computing. En d'autres mots, les méta-modèles de Petrovic et Tosic permettent la modélisation de la topologie statique (modèle de déploiement) et du comportement dynamique de leur application (stratégie d'adaptation). Cette modélisation est au centre de SMADA-fog puisqu'elle permet à la fois l'annotation du modèle de déploiement dans l'ontologie, mais également l'utilisation des stratégies d'adaptations par le générateur de code pour s'adapter aux changements de son environnement. Afin d'optimiser les paramètres du générateur de code, les auteurs proposent d'utiliser des algorithmes d'optimisation linéaire. Le reste du composant structurel consiste à déployer le code ainsi généré et répercuter les effets observés sur la structure sémantique.

L'approche de SMADA-fog est particulièrement intéressante dans son utilisation des structures sémantiques pour la génération de code automatisée. Cette opération s'apparente en grande partie aux principes de déploiement de services en raison de la nature personnalisable du code déployé. Les travaux de Petrivic et Tovic [205] proposent une approche fortement approfondie sur l'implémentation démontrant l'apport des structures sémantiques dans la représentation de données complexes. Les services de sécurité pouvant requérir une multitude de contraintes spécifiques, ces travaux justifient l'utilisation de structures sémantiques dans notre contexte. SMADA-fog rencontre cependant ses limites dans la définition de sa structure sémantique et dans son modèle d'optimisation. En effet, l'étude présente brièvement l'ontologie utilisée sans identifier ses capacités d'interopérabilité avec d'autres ontologies. Dans une volonté de traiter de connaissances complexes, il serait intéressant d'être en mesure de lier ces concepts à d'autres ontologies pour accroitre la base de connaissance et ses capacités de raisonnement. L'ontologie définie dans SMADA-fog ne semble pas être répertoriée dans LOV4IoT ni être disponible publiquement. De plus, cette

structure repose uniquement sur l'utilisation d'optimisation linéaire et des capacités de raisonnement des ontologies pour effectuer son déploiement de code. Il serait intéressant de constater l'apport des structures ontologies auprès d'autres algorithmes de résolution.

En dépit des limites de sa structure ontologique, SMADA-fog décrit habilement les principes de déploiement de code personnalisé dans une infrastructure fog.

4.3.3 Construction des ontologies

La construction d'une ontologie est une opération complexe qui requiert une expertise à la fois dans le domaine de l'ontologie, mais également dans la sémantique du web. La principale difficulté lors de la construction de ces structures réside dans le fait qu'un individu bénéficie rarement de ces deux domaines d'expertises. Pour remédier à ce problème tout en facilitant la construction, plusieurs bonnes pratiques et méthodologies sont recensées dans la littérature. Ces recommandations permettent notamment de préserver les propriétés des ontologies telles que l'interopérabilité, l'utilisabilité, la lisibilité et l'extensibilité.

4.3.3.1 Bonnes pratiques de construction

Les ontologies sont des structures sémantiques comprises dans la sémantique du web. De ce fait, l'ensemble des conseils et recommandations sur ce dernier domaine s'appliquent également aux ontologies. Berners-Lee [62] liste quatre règles qui fournissent les bases à respecter pour la construction de structures sémantiques. Ces règles impliquent l'utilisation d'URI (Uniform Resource Identifier) avec de préférence le format HTTP des URI lors du nommage d'un élément pour faciliter son accès en ligne. La présence d'une documentation ainsi qu'une liaison des ressources décrites avec des ressources externes sont également des pratiques qui sont fortement recommandées.

Plus spécifiquement sur les ontologies, les travaux de Gyrard et al. [131, 17] ainsi que ceux de Bermudez-Edo et al. [61] définissent chacuns des listes de bonnes pratiques à respecter lors de la conception d'ontologies. Les travaux de Gyrard et al. proposent une liste de seize bonnes pratiques alors que ceux de Bermudez-Edo et al. en listent dix. D'un point de vue général, ces recommandations se résument à six catégories : Unification, Méthodologie et respect des bonnes pratiques, Documentation, Réutilisation des connaissances existantes, Validation et pour finir, Disponibilité et partage.

Unification

Les représentations sémantiques, les données et les termes utilisés doivent être unifiés afin d'assurer l'interopérabilité de chacun des composants définis. L'utilisation de termes unifiés lève des potentielles ambiguïtés entre le nom des classes. Par exemple, la présence de classes nommées “t” pour le temps et “t” pour température serait confuse. L'utilisation d'un vocabulaire unifié est adapté est alors important pour prévenir les ambiguïtés et faciliter l'interopérabilité.

Méthodologie et respect des bonnes pratiques

Le respect des recommandations et bonnes pratiques contribue à l'amélioration de l'interopérabilité des ontologies ainsi que les applications qui en dépendent. En plus de faciliter la construction des ontologies, le respect des méthodologies permet à des experts de domaines de construire leurs ontologies sans nécessiter une expertise élevée en sémantique du web.

Documentation des données et des représentations sémantiques

La construction d'une ontologie dépend fortement de l'individu qui l'a conçue et l'utilise. Une documentation détaillée des données, mais également de la représentation sémantique permet à de nouveaux utilisateurs d'appréhender l'ontologie, de comprendre les concepts définis, mais surtout d'utiliser l'ontologie correctement.

Réutilisation des connaissances existantes

La réutilisation des connaissances existantes, souvent sous la forme d'ontologies externes, permet d'éviter au maximum possible de redéfinir des concepts déjà définis. Afin de préserver la stabilité de l'ontologie, il est recommandé de privilégier la réutilisation des ressources populaires et correctement maintenues.

Validation

La validation d'une ontologie est une étape importante qui permet la détection de potentielles erreurs de syntaxe et d'éviter les pièges récurrents. Plusieurs outils en ligne existent pour effectuer ces opérations telles que RDF Triple-Checker [14] pour la syntaxe et OOPS ! [211, 11] pour la détection de pièges.

Disponibilité et partage

La disponibilité et le partage des ontologies sont primordiaux pour procurer une représentation sémantique réutilisable. Pour ce faire, plusieurs aspects doivent être considérés. Un premier concerne le nom de l'ontologie ainsi que son adresse. Les ontologies sont communément nommées d'après leurs adresses, dans ces conditions, une ontologie pourrait alors ne plus être accessible dans le cas d'un changement de nom de domaine. Une pratique courante pour remédier à ce problème consiste à utiliser des adresses URL permanentes [13]. Par la suite, un second aspect concerne le partage de l'ontologie. Même si une ontologie est disponible, elle n'est pas forcément visible.

Un site internet est disponible dès sa mise en ligne ; cependant, il n'est pas forcément visité, car aucun moteur de recherche ne l'a référencé pour le moment. Ce principe est identique pour les ontologies. Une fois l'ontologie mise en ligne, il est nécessaire de la partager et de la référencer sur diverses plateformes sémantiques telles que Linked Open Vocabularies (LOV) [8].

4.3.3.2 Méthodologies de construction

Nous avons souligné précédemment qu'un manque d'expertise dans le domaine de la sémantique du web avait un impact important dans la création d'ontologies. Pour remédier à ce problème, la communauté scientifique qui travaille autour de cette thématique propose une multitude de méthodologies de conception d'ontologies.

Un élément récurrent de ces méthodologies implique la réutilisation des connaissances existantes pouvant être modélisées ou non sous la forme d'ontologies [201, 250]. La réutilisation d'ontologies accroit l'interopérabilité de l'ontologie construite et améliore les applications qui en dépendent.

Dans leur papier, Noy et al. [201] fournissent trois règles fondamentales pour la conception d'ontologies. La première stipule qu'il n'existe pas d'unique manière de modéliser un domaine, cela signifie que chaque ontologie dépend de son application et de son concepteur. Deux individus qui modélisent les mêmes concepts vont produire deux ontologies différentes. La seconde règle souligne l'aspect itératif de la conception des ontologies. Il est peu probable que la première version d'une ontologie réponde à toutes les attentes d'une application spécifique. Plusieurs itérations sont attendues avant de produire une version satisfaisante. La dernière règle déclare que les concepts décrits dans l'ontologie doivent être proches des objets (physiques ou logiques) et des relations de son domaine de prédilection.

De manière similaire, la méthodologie On-To-Knowledge (OTKM) [249] aborde cette notion de conception itérative. Cette méthodologie comporte cinq étapes : "Feasability study", "Kickoff", "Refinement", "Evaluation" et "Application and Evolutions". Lors du processus de conception de l'ontologie, on rencontre communément plusieurs cycles entre les étapes de "Refinement" et "Evaluation". Ces cycles ont pour objectif d'accroitre l'adéquation de l'ontologie avec son application cible. La notion d'évaluation d'ontologies reste cependant très relative. En effet, Bajaj et al. expliquent dans leur papier qu'il n'existe aucune méthode concrète pour évaluer une ontologie [55]. De ce fait, l'évaluation est alors basée sur le respect de l'ontologie vis-à-vis des attentes de son concepteur. Sure et al. ont mis en évidence que, en règle générale, une bonne première version d'ontologie doit fournir suffisamment de matière pour la construction d'un prototype d'application [249]. Une fois le prototype opérationnel, un nouveau processus itératif peut être relancé pour améliorer l'ontologie.

L'une des tâches essentielles de la conception d'ontologies consiste à décrire les différentes classes qui la composent ainsi que leur hiérarchie. Cette tâche est décrite comme la quatrième étape de la méthodologie "Ontology Development 101" de Noy et al. [201].

Uschold et Gruninger décrivent [257], dans leur papier, trois catégories de hiérarchies différentes : "top-down", "bottom-up" et "middle-out".

La première, "top-down" définit les concepts généraux en premier et les spécifie ensuite jusqu'à atteindre la représentation souhaitée. La seconde, "bottom-up", à l'inverse, décrit les concepts les plus précis en premier et généralise par la suite pour atteindre le niveau d'abstraction requis. La dernière catégorie de hiérarchie, "middle-out" consiste à définir en premier les concepts les plus importants et construire le

reste de l'ontologie autour (au-dessus et en dessous) en fonction de la représentation désirée.

L'approche "top-down" a tendance à imposer l'ajout de classes avec un haut niveau hiérarchique dans l'ontologie ce qui l'encombre et l'alourdit inutilement. De la même manière, une approche "bottom-up" introduite de nombreux concepts qui ne sont pas systématiquement nécessaires dans l'application de notre ontologie. Une hiérarchie "middle-out" correspond cependant parfaitement à nos besoins. Cette dernière permet de construire l'ontologie en se focalisant sur les concepts essentiels à décrire tout en offrant de la souplesse pour en ajouter d'autres si le besoin s'en fait sentir. De plus, comme l'ont démontré Bermudez-Edo et al. la taille d'une ontologie est importante lors d'une utilisation dans un contexte IoT [61]. Avoir une ontologie restreinte à ses classes essentielles semble donc pertinent dans notre situation.

4.4 Méthodologie de comparaison

Comparer des solutions de placement requiert d'avoir une certaine conscience du contexte dans lequel la résolution est effectuée, mais également, d'avoir connaissance des problèmes de placement relatifs à ces solutions.

Pour ce faire, nous nous appuyons sur la formalisation des problèmes de placement de service de sécurité dans l'IoT effectuée en Section 4.2. Nous proposons ainsi de représenter l'ensemble des connaissances relatives à ce problème sous la forme d'une structure sémantique et plus particulièrement, à l'aide d'une ontologie.

Cette ontologie est ainsi utilisée pour constituer une base de connaissance et agir comme pierre angulaire dans notre méthodologie de résolution des problèmes de placement et des comparaisons de leurs solutions.

4.4.1 Création d'une base de connaissance

Une base de connaissance regroupe un ensemble d'informations sur un domaine particulier sous un format accessible et interprétable. Dans le cas des ontologies, une base de connaissance est constituée d'un ensemble d'instances individuelles de classes définies dans une ontologie.

4.4.1.1 Avantages d'une base de connaissance ontologique

L'utilisation d'ontologies pour la représentation de bases de connaissance comporte deux atouts. Le premier repose sur l'aspect formel de la structure. En effet, comme le démontre Shimizu, les langages OWL (Web Ontology Language), communément utilisé pour la rédaction d'ontologie est directement lié aux notions de logique descriptive [239]. Les logiques descriptives sont des sous-ensembles décidables de logiques du premier ordre enrichis de sémantique formelle permettant l'interprétation des connaissances renseignées dans une ontologie de manière précise et sans ambiguïté par les humains, mais surtout par les machines.

Le second avantage d'utiliser des ontologies repose sur les inférences. Les structures ontologiques sont régulièrement agrémentées de raisonneurs qui, au travers l'utilisation

d'outils logiques, sont en mesure de déterminer des inférences. Les inférences sont des informations implicites de la base de connaissance qui n'ont pas été explicitement décrite dans l'ontologie. Par exemple, je peux décrire dans mon ontologie le concept de *grand-père* comme le *père* du *père* d'un individu. Si je renseigne dans cette ontologie trois individus : Alice, Bob et Chuck et que je formule explicitement que Bob est le *père* de Alice et Chuck le *père* de Bob, le raisonneur est en mesure de déterminer que Chuck est également le *grand-père* de Alice sans que cette information n'ait été explicitement renseignée. Les inférences sont particulièrement intéressantes dans notre contexte puisqu'elles permettent d'identifier de nouvelles informations depuis celles qui ont été transcrites dans l'ontologie. Cet apport est d'autant plus conséquent dans le domaine de l'IoT et de la sécurité où la quantité d'informations est importante et où une connaissance conséquente du contexte est souhaitée.

4.4.1.2 Constitution de la base de connaissance

La construction d'une base de connaissance, comme la création d'une ontologie, est une procédure itérative. Dans un premier temps, la base est créée et un minimum de connaissances y est renseigné. Par la suite, cette base a vocation à être partagée et enrichie par la communauté scientifique afin de bénéficier d'une structure commune dans l'étude des problèmes de placement dans l'IoT. La construction de notre base de connaissance s'effectue en trois étapes dont l'enchainement est illustré en Figure 4.3.

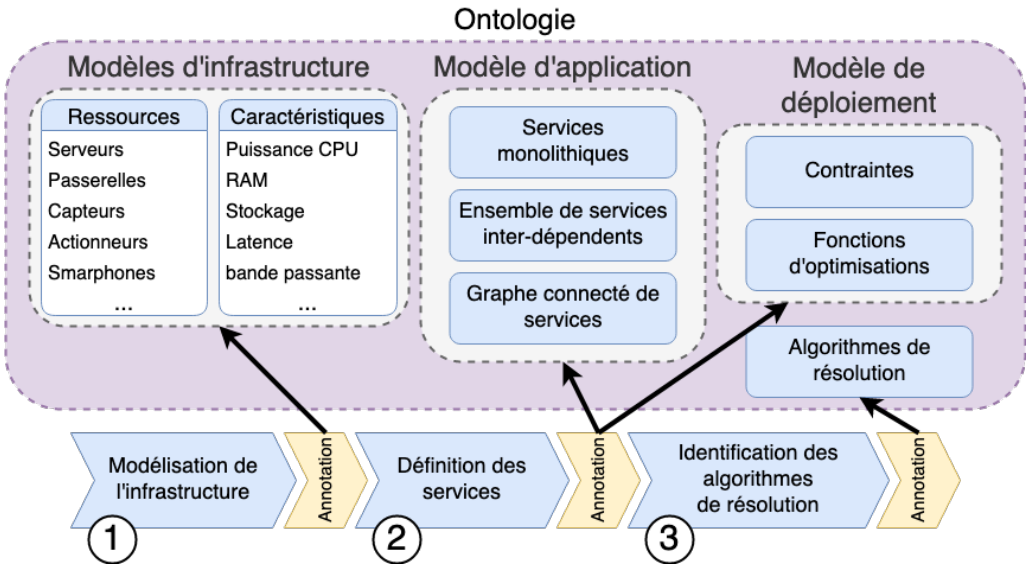


FIGURE 4.3 – Processus de constitution de notre base de connaissance.

Modélisation de l'infrastructure

La première étape consiste à modéliser l'infrastructure sur laquelle les problèmes vont être appliqués. Cette modélisation doit reprendre l'ensemble des informations disponibles sur le réseau IoT traité. Cela comprend les ressources physiques comme

les dispositifs (PCs, serveurs, capteurs...) présents, leurs caractéristiques (CPU, RAM...) et celles du réseau. Modéliser l'infrastructure consiste à définir la structure $\mathcal{M} = (\mathcal{D}, \mathcal{R})$ introduite dans notre formalisation des problèmes de placement en Section 4.2.

La modélisation de l'infrastructure requiert un certain niveau de connaissance sur cette dernière. Dans notre cas, nous avons fait le même choix que celui évoqué en Section 3.2.7, à savoir, celui d'avoir une connaissance totale de la topologie du réseau, de sa constitution et de ses caractéristiques. Cette connaissance implique donc que nous nous retrouvions dans un modèle d'orchestration centralisé. Une approche décentralisée est toutefois envisageable en partageant la structure sémantique telle que le propose Nezami et al. dans leurs travaux [199]. Une telle procédure n'est pas souhaitable dans notre cas en raison du fort nombre d'interactions nécessaires pour préserver les représentations de l'infrastructure sur chaque entité.

La modélisation de l'infrastructure se conclut par l'annotation du modèle dans l'ontologie sous la forme d'instances individuelles. Le processus d'annotation est semblable à celui de l'insertion de données dans les services de gestion de base de données classiques. La donnée est saisie dans l'ontologie ainsi que son type, ses propriétés et l'ensemble des autres relations nécessaires à sa modélisation.

Définition des services

La seconde étape de constitution de notre base de connaissance repose sur la définition des services à placer dans l'infrastructure IoT. Nos travaux se focalisent principalement sur les services de sécurité cependant, la méthodologie proposée s'étend également aux services plus généraux.

La définition des services requiert que chaque service soit défini individuellement selon son modèle de déploiement. Le modèle de déploiement d'un service comprend les contraintes de positionnement du service et au moins une fonction objective dont leurs définitions formelles sont introduites en Section 4.2.2.1 et 4.2.2.2.

Les contraintes permettent de déterminer si une solution de placement du service est satisfaisante d'après certains critères prédéfinis. L'exemple le plus couramment utilisé dans la littérature concerne les caractéristiques du dispositif d'accueil. Une contrainte peut alors stipuler qu'une solution de placement est valide si et seulement si le dispositif d'accueil possède suffisamment de capacité pour exécuter ce service. Cette contrainte, d'apparence très simple, peut être complétée d'une multitude de critères afin de personnaliser les conditions de positionnement du service. Les conditions pour qu'une contrainte exigeante soit satisfaite peuvent impliquer qu'il n'existe pas de solution de placement satisfaisante. Dans un tel cas, il peut être parfois intéressant de relaxer les contraintes afin d'effectuer un compromis entre le résultat et les exigences et d'être en mesure de positionner le service dans le réseau.

Le principe de la fonction objective est d'endosser le rôle d'évaluateur sur une solution de placement. Cette fonction associe couramment un score à une solution de placement et tend à maximiser ou minimiser ce score en fonction des besoins du service. L'intérêt de cette fonction objective réside dans sa capacité à optimiser

le positionnement en déterminant, parmi les solutions de placement satisfaisantes, laquelle est la plus adaptée.

Lorsque plusieurs services doivent être positionnés, leurs définitions dans la base de connaissance doivent également décrire comment ces services interagissent, c'est le rôle du modèle d'application. Le modèle d'application détermine si les services sont monolithiques, c'est-à-dire que les services forment des blocs indépendants qui n'interagissent pas, si certains sont interdépendants ou si la totalité des services est dépendante les unes des autres. La description du modèle d'application est une étape très importante dans la conception de notre base de connaissance et détermine en grande partie la capacité d'un ensemble de services à être positionnés. En effet, on distingue facilement que le positionnement de services interdépendants sera plus complexe que de positionner des services indépendants où seules les contraintes de ce service seront à prendre en compte.

De la même manière que lors de la modélisation de l'infrastructure, l'étape de définition des services se termine par l'annotation des services et connaissances associées dans la base de connaissance.

Conception des algorithmes de résolution

La recherche de solutions à des problèmes de placement de services nécessite l'utilisation d'algorithmes de résolution. Ces algorithmes peuvent être de différentes natures comme nous l'avons présenté en Section 3.2.5. Cela peut être des algorithmes mathématiques reposant majoritairement sur de la programmation linéaire (ILP ou MILP), des algorithmes de recherche constitués d'heuristiques, des algorithmes génétiques ou même des algorithmes d'apprentissage.

L'ensemble de ces algorithmes peuvent ainsi être renseignés dans la base de connaissance et mis en relation avec les services avec lesquels ils sont compatibles. Lors de l'annotation des algorithmes de résolution dans la base de connaissance, l'objectif est de faire en sorte qu'ils soient compatibles avec le modèle de l'infrastructure, mais également de renseigner les services auxquels ils peuvent contribuer à la découverte de solutions de placement. L'objectif étant qu'il soit possible d'appliquer ces algorithmes à l'ensemble des services décrits dans la base de connaissance sans pour autant que les solutions retournées soient optimales.

4.4.2 Résolution d'un problème de placement

Avant d'être en mesure de comparer des solutions de placement de services, il faut préalablement être en mesure de résoudre un problème de placement de services. La méthodologie de résolution que nous proposons généralise celle abordée dans le Chapitre 3 et se décline en quatre étapes représentées dans la Figure 4.4 : la récupération des informations, l'identification des solutions, la validation et la sélection de la meilleure solution. L'ensemble de ces étapes sont décrites plus en détail dans les paragraphes suivants.

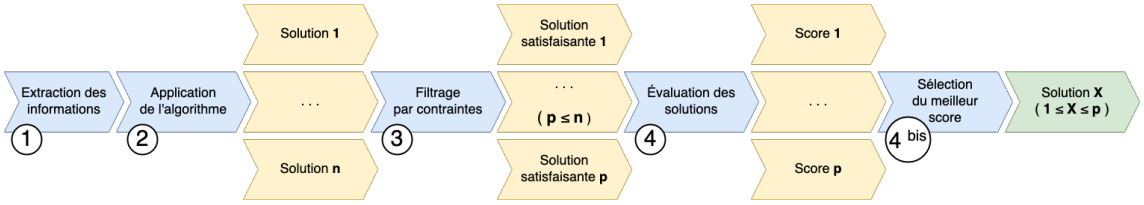


FIGURE 4.4 – Résolution d'un problème de placement utilisant la base de connaissance.

Extraction des informations

La première démarche à effectuer pour résoudre un problème de placement à l'aide de notre base de connaissance consiste à acquérir une base de connaissance. Son acquisition peut s'effectuer soit par sa construction comme détaillée en Section 4.4.1.2 soit en la récupérant depuis d'autres travaux, ou encore en enrichissant une base déjà existante.

La base de connaissance acquise, il est possible d'interpréter les informations relatives à l'infrastructure du réseau, les données des services, leur modèle de déploiement, les algorithmes de résolution ainsi que l'ensemble des relations entre les entités décrites dans l'ontologie.

Application de l'algorithme

La seconde étape du processus de résolution consiste à appliquer un algorithme de résolution pour identifier des solutions de placement. Ces algorithmes se basent sur les informations extraites à l'étape précédente et plus généralement sur l'ensemble de la base de connaissance (contraintes incluses) pour l'identification de solutions. En fonction des algorithmes, une ou plusieurs solutions peuvent être retournées. Il arrive même que certains algorithmes puissent directement retourner la solution optimale cependant, cela signifie que l'algorithme décrit des notions d'optimisation comme abordé dans les étapes suivantes.

Filtrage par contraintes

Le filtrage par contraintes consiste à appliquer les contraintes du service sur l'ensemble des solutions retournées par l'algorithme. Pour un ensemble de n solutions initiales, le filtrage par contrainte détermine la satisfaisabilité de chacune des solutions. Ainsi, pour les n solutions en entrée, cette opération identifie p solutions satisfaisables avec $p \leq n$. Les solutions qui ne satisfont pas les contraintes ne sont pas conservées. Il peut arriver qu'il n'y ait aucune solution satisfaisante, dans ce cas, un relâchement des contraintes est envisageable pour proposer des conditions de placement moins strictes. Dans le cas où les algorithmes de résolution utilisent les contraintes dans leur identification de solution, le filtrage peut constituer une étape redondante. Elle reste cependant nécessaire pour valider les solutions retournées par l'algorithme.

Évaluation et sélection

Les étapes d'évaluation et de sélection de solutions sont communément associées à l'utilisation de la fonction objective. Le processus d'évaluation des solutions consiste à appliquer l'évaluateur sur chacune des solutions de placement satisfaisantes afin de déterminer leurs scores respectifs. La résolution des problèmes de placement de services implique régulièrement des notions d'optimisation sous la forme de maximisation ou de minimisation de scores. L'aspect optimisation est représenté sur la Figure 4.4 par l'étape 4 bis et consiste simplement à appliquer une fonction de maximisation ou de minimisation sur l'évaluateur.

Une fois le meilleur score identifié, la solution associée est communiquée à l'entité en charge d'effectuer le déploiement de services.

4.4.3 Processus de comparaison

Le processus de comparaison des solutions de placement possède de nombreuses similarités avec la méthodologie de résolution décrite précédemment. En effet, comme illustré dans la Figure 4.5, la majorité des étapes du processus sont similaires.

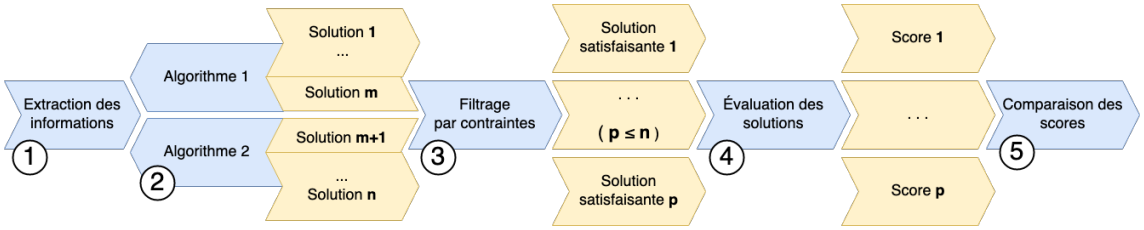


FIGURE 4.5 – Processus de comparaison des solutions de placement.

La première phase de la procédure consiste encore une fois à interpréter l'information présente dans la base de connaissance afin d'appréhender l'infrastructure. La seconde étape subit quelques modifications avec l'introduction de plusieurs algorithmes de résolution. Contrairement à la méthode de résolution précédente, la comparaison des solutions implique l'utilisation de plusieurs algorithmes. Les étapes suivantes restent invariantes ; le filtrage des contraintes identifie les solutions satisfaisantes parmi l'ensemble des solutions que retournent les algorithmes de résolution. La phase d'évaluation analyse les solutions satisfaisantes et leur détermine chacune un score. Le dernier changement dans le processus intervient après l'attribution de ces scores. Dans la résolution des problèmes de placement, nous récupérions la meilleure solution, dorénavant, l'analyse des scores est plus minutieuse et constitue l'étape finale du processus de comparaison.

Le premier atout de notre processus de comparaison réside dans sa capacité à identifier les algorithmes qui retournent les meilleures solutions. Lors de l'utilisation d'un unique évaluateur comme illustré dans la Figure 4.5, on distingue facilement l'algorithme qui retourne le meilleur résultat en retraçant l'origine de la solution qui a obtenu le meilleur score selon les critères de l'évaluation.

Dans son état actuel, nous définissons les évaluateurs comme des fonctions qui retournent une unique valeur numérique. D'un point de vue pratique, un adminis-

trateur de réseau IoT pourrait souhaiter mettre en concurrence ces évaluateurs afin de comparer les solutions de positionnement selon plusieurs critères et déterminer son choix d'après ces derniers. Pour ce faire, une variation des évaluateurs est possible lors de la quatrième étape. Soit p le nombre de solutions satisfaisantes et q le nombre d'évaluateurs, il en résulte un nombre $p * q$ de scores sur ces solutions de positionnements.

Cette nouvelle diversité permet, au travers la comparaison des solutions retournées, d'identifier quels algorithmes retournent les solutions les plus intéressantes, mais également quels critères d'évaluation discriminent le mieux ces dernières. Les conditions principales pour assurer un bon fonctionnement de cette méthodologie résident dans la constitution de la base de connaissance ainsi que dans la bonne définition de l'ontologie initiale.

4.4.4 Présentation de notre ontologie

L'objectif de cette ontologie est de permettre la modélisation des problèmes de placement de services dans l'IoT, d'uniformiser les travaux sur cette thématique et de comparer les solutions de placement identifiées. Nous avons nommé cette ontologie FSPlacementOntology en référence au *Fog Service Placement Problem* (FSPP) défini par Skarlat et al. [244].

Afin que l'ontologie modélise au mieux les problèmes de placement de services, cette dernière doit contenir l'ensemble des modèles nécessaires pour la description de ces problèmes et illustré en Figure 3.1, à savoir : le modèle d'infrastructure, d'application et de déploiement. De plus, afin de correspondre à la méthodologie proposée dans les sections précédentes, l'ontologie doit également comporter une représentation des algorithmes de résolution et des solutions de placement. La Figure 4.6 illustre l'aspect actuel de l'ontologie FSPlacementOntology d'après la notation visuelle VOWL [174].

4.4.4.1 Concepts importés

Les problèmes de placement de services et plus particulièrement le FSPP s'axent majoritairement autour de la notion de service. Nous avons ainsi décidé de focaliser notre modélisation autour d'une classe *Service* représentant cette notion. Afin de respecter les bonnes pratiques de conception d'ontologie décrites en Section 4.3.3.1, nous avons réutilisé la classe *Service* provenant de l'ontologie IoT-Lite [61]. Suivant une approche hiérarchique de type *middle-out*, nous avons modélisé le reste des concepts essentiels à notre application autour de cette classe *Service*.

L'ontologie IoT-Lite modélise l'exposition d'un service par un dispositif à l'aide des propriétés *exposes* et *exposedBy* reliant les services à la classe *Device* définie dans l'ontologie SSN [16]. Dans le cas de notre application, les services sont hébergés sur les objets. La modélisation de la classe *Device*, représentative de ces objets est alors nécessaire. Ainsi, les classes *Device* et *System* (super-classe de *Device*) sont importées dans FSPlacementOntology, de même que les propriétés *exposes* et

4. `http://vowl.visualdataweb.org/webvowl.html`

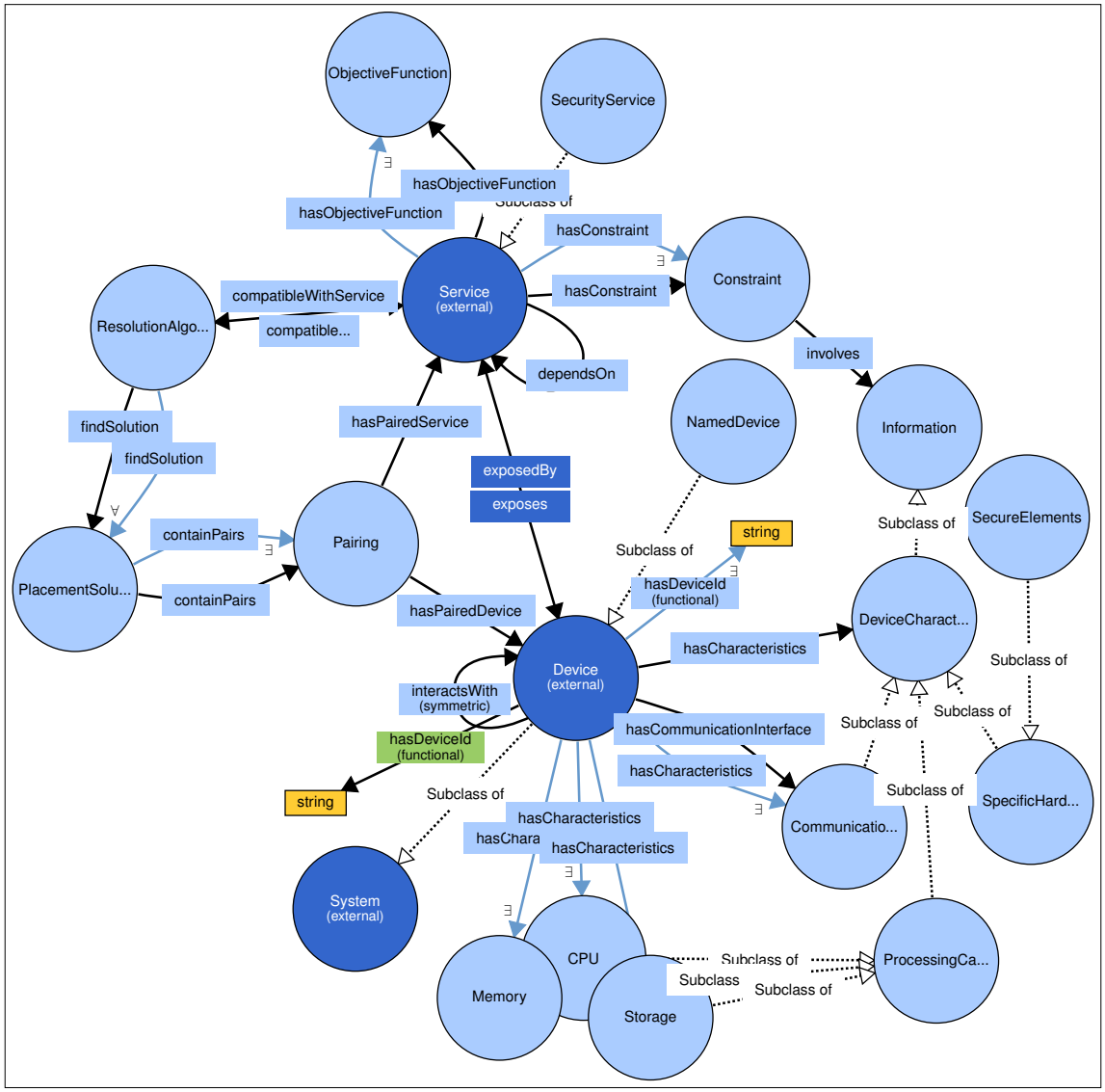


FIGURE 4.6 – Visualisation de FSPlacementOntology d'après l'outil de visualisation WebVOWL [4]

exposedBy. L'import de concepts externes est illustré en bleu foncé dans la Figure 4.6.

4.4.4.2 Concepts Principaux

Représenter le modèle de déploiement dans l'ontologie requiert la définition des concepts associés aux contraintes et aux fonctions objectives. Pour ce faire, nous avons défini une classe *Constraint* représentant les conditions à satisfaire pour qu'une solution de placement soit valide. Pour des raisons de réalisme, lors de l'annotation d'un *Service* dans l'ontologie, au moins une contrainte de placement doit lui être renseignée au travers de la propriété existentielle *hasConstraint*. Nous avons également

introduit une seconde classe, *Information* reliée à *Constraint* par la propriété *involves* pour modéliser les informations qui sont utilisées pour la définition d'une contrainte. Cette classe *Information* recense notamment dans notre cas, les caractéristiques des appareils (*DeviceCharacteristics*) ou encore les composants matériels spécifiques (*SpecificHardwareComponent*). Afin de compléter le modèle de déploiement, les fonctions objectives constituent des instances de la classe *ObjectiveFunction*. Cette classe est mise en relation avec *Service* par l'intermédiaire d'une propriété existentielle *hasObjectiveFunction* stipulant qu'un *Service* doit impérativement être associé à au moins une fonction objective lors de sa création.

Le modèle d'application est modélisé au travers la propriété *dependsOn* reliant plusieurs instances de la classe *Service* entre elles afin de refléter les potentielles interdépendances des services.

Pour correspondre le plus fidèlement à l'application souhaitée, les algorithmes de résolution sont modélisés dans l'ontologie (*ResolutionAlgorithm*) et mis en relation avec les solutions (*PlacementSolution*) par l'intermédiaire de la propriété universelle *findSolution*. Cette propriété représente la possibilité des algorithmes de résolution à retourner exclusivement des instances de *PlacementSolution* ou à ne rien retourner. Les solutions de placement comportent au minimum, déterminé grâce à la propriété existentielle *containPairs*, une instance de la classe *Pairing*. Cette dernière associe exclusivement une instance de *Service* avec une instance de *Device* au travers des propriétés *hasPairedService* et *hasPairedDevice*.

Pour finir, la classe *Device* est mise en relation avec de nombreux autres concepts relatifs aux performances de l'appareil (*CPU*, *Storage*, *Memory*), ses interfaces de communication (*CommunicationInterface*) ou plus généralement ses caractéristiques (*DeviceCharacteristic*).

4.4.4.3 Évolution de l'ontologie

Nous avons fait le choix de décrire une ontologie simplifiée afin de limiter le nombre d'axiomes qu'elle comporte et de permettre une plus grande fluidité d'utilisation dans un contexte IoT. Bermudez-Edo et al. [61] ont montré l'impact que peut avoir la taille d'une ontologie sur son temps de réponse et le fait qu'une petite ontologie était plus appropriée dans un contexte d'IoT. L'ontologie FSPlacementOntology comporte actuellement 93 axiomes respectant ainsi leurs recommandations. De plus, un nombre d'axiomes réduit facilite la compréhension de l'ontologie et accroit sa capacité à être réutilisée.

Les avantages de cette ontologie résident dans la capacité d'annotation des problèmes de placement ainsi que dans sa flexibilité pour décrire les contraintes de placement à l'aide de restrictions sémantiques. Horridge et al. [141] définissent une restriction comme une catégorie d'individus basée sur les relations auxquelles les membres de la catégorie participent. Pour une application de placement de fonctions de sécurité dans le fog, une telle restriction pourrait être la suivante :

Le groupe d'individus de type *Device* pouvant héberger une fonction virtuelle (VirtualFunction) telle qu'une Virtual Network Function (VNF) et qui comportent au moins une relation *hasComponent* avec un membre de la classe *SecureElement*.

Formulée autrement, cette restriction regroupe tous les appareils qui bénéficient d'éléments de sécurité et qui sont capables d'accueillir une fonction virtuelle comme une VNF. La restriction précédente comporte des concepts qui ne sont pas décrits actuellement dans l'ontologie. Le principe est de fournir une ontologie basique qui puisse être reprise et enrichie par la communauté afin de correspondre aux différentes applications.

Idéalement, pour que cette ontologie soit enrichie et qu'elle puisse servir à l'élaboration de bases de connaissance à la communauté scientifique, elle doit être mise publiquement à disposition. Malheureusement, l'opération n'est pas encore autorisée au moment de la rédaction de cette thèse.

Pour permettre à la communauté scientifique d'exploiter la méthodologie proposée dans cette contribution, il est nécessaire de partager la structure de cette ontologie. Nous avons décrit, dans les sections précédentes, les concepts essentiels de l'ontologie, ainsi que le raisonnement effectué lors de sa conception. Une telle description n'est pas suffisante pour permettre une bonne reproductibilité de l'ontologie réalisée. Pour pallier à ce problème, nous avons illustré notre ontologie sous le format VOWL [174] en Figure 4.6. Nous complétons cette visualisation à l'aide des tables B.1 et B.2, présentes en annexe B, qui décrivent l'ensemble de l'ontologie (les classes et propriétés) sous la forme logique. Ces formulations ont été générée à l'aide de l'outil OWL Syntax Converter [5] à partir de notre fichier d'ontologie.

Nous espérons ainsi permettre à la communauté scientifique d'accéder à l'ontologie décrite dans ces travaux, de construire leurs bases de connaissance et d'exploiter les travaux de cette contribution.

4.5 Illustration sur un cas concret

Dans cette section, nous illustrons les méthodologies présentées précédemment en nous appuyant sur les travaux du Chapitre 3. La conception de la base de connaissances étant l'étape la plus délicate, nous focalisons ainsi notre présentation sur celle-ci. À l'issue de cette illustration, une brève analyse des mesures de notre base de connaissances est présentée afin d'alimenter la discussion sur l'apport de ces travaux.

4.5.1 Modélisation du dataset

La première étape et probablement la plus importante de la création de cette base de connaissance résident dans la modélisation d'une infrastructure. Nous avons fait le choix de représenter les travaux abordés dans la contribution précédente ce qui implique, en plus des divers concepts de ces travaux, de modéliser ce dataset dans la base de connaissance.

Nous avons ainsi détaillé les données récoltées du projet Social IoT [25] et plus précisément, la matrice d'adjacence OOR. Cette matrice représente un graphe d'objet connecté. Les interactions entre ces objets sont déduites d'après les interfaces de

5. `https://www.ldf.fi/service/owl-converter/`

communication des dispositifs et de leur emplacement géographique de sorte qu'une liaison entre deux appareils représente une communication potentielle. De plus amples détails sur ce jeu de données sont apportés en Section 3.5.1 de ce manuscrit. La particularité de ces données réside dans leur absence d'informations relatives aux objets. Seule une catégorisation des dispositifs est explicitement mentionnée. Notre attention se porte en particulier sur les dispositifs de la catégorie *Street Lights* en raison d'une potentielle capacité à accueillir des fonctions de sécurité.

La Figure 4.7 propose une visualisation générale de la base de connaissance réalisée. À première vue, les changements effectués sur les classes sont relativement faibles. On observer l'apparition de sous-classes de *NamedDevice* qui illustrent chacune des catégories d'objets présents dans le jeu de données. Les valeurs numériques apparaissant dans les bulles représentent le nombre d'instances de leur type. La grande majorité des instances ainsi observables sont des éléments de la classe *Device*. Ces instances correspondent à chacun des 1 458 appareils recensés dans la grande composante connexe du jeu de données SIoT. Chacune de ces instances est reliée entre elles par la propriété *interactsWith* afin de représenter les arêtes dans le graphe initial.

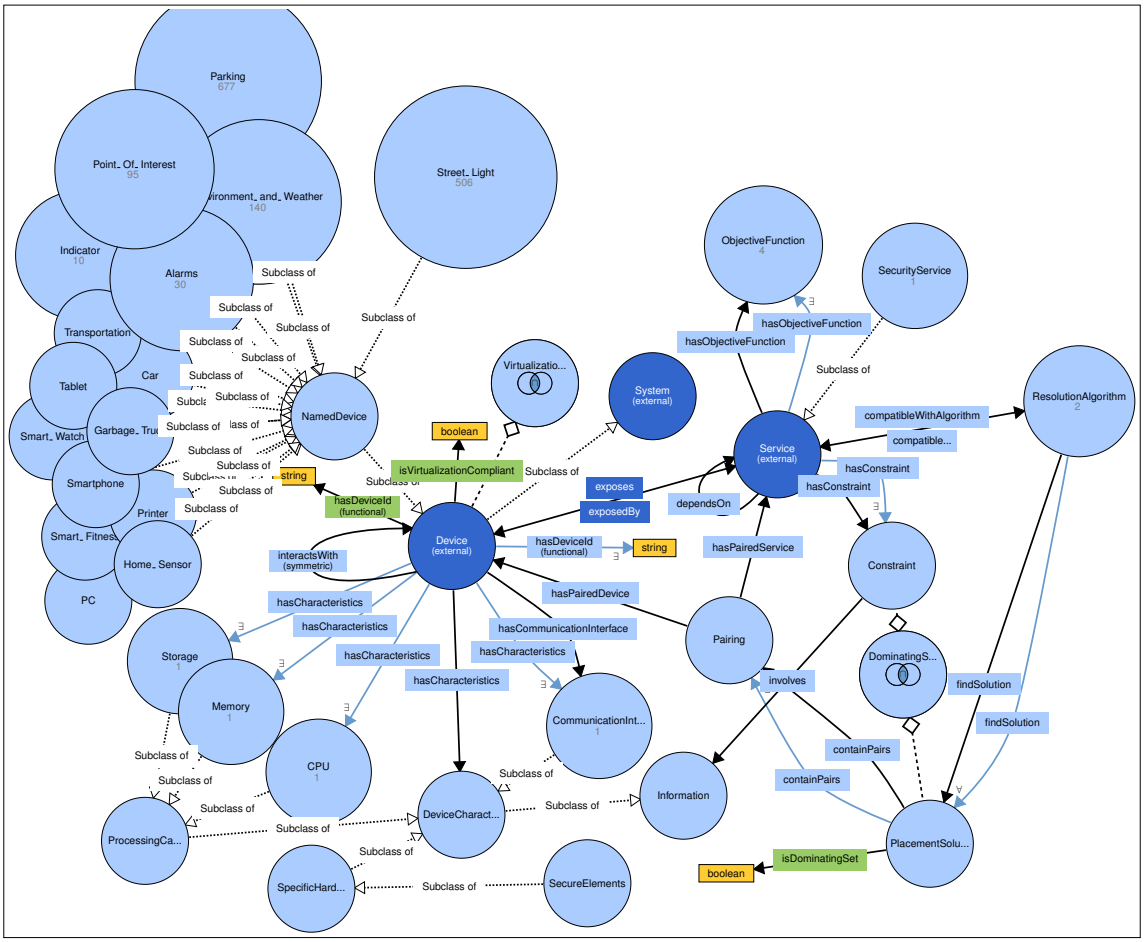


FIGURE 4.7 – Visualisation de la base de connaissance en VOWL [174].

4.5.2 Définition des services

Les travaux que nous modélisons dans cette base de connaissance n'introduisent pas de changements majeurs dans notre ontologie. En effet, comme l'on peut l'observer en Figure 4.7, seule une instance de service a été introduite (*SurrogateEndToEndEncryptionService*). En l'absence d'un modèle d'application divergeant de services monolithiques, aucune notion de dépendance (via *dependsOn*) ne nécessite d'être définie pour le moment.

De manière similaire, les contraintes et fonctions objectives de notre problème de placement se définissent sans altération de l'ontologie initiale. Ces contraintes consistent à déterminer si une solution représente un ensemble dominant du graphe. La procédure consiste à définir une propriété de données indiquant si une instance de *PlacementSolution* constitue un ensemble dominant ou non. Les instances des fonctions objectives sont plus nombreuses, au nombre de quatre, elles représentent chacune un critère d'évaluation abordé dans nos travaux précédents, à savoir, la taille de l'ensemble dominant, la protection et protection normalisée (voir Figure 3.9), et la qualité de la solution correspondant à la proportion d'appareils en mesure d'héberger le service de sécurité souhaité dans l'ensemble dominant.

4.5.3 Description des algorithmes de résolution

La saisie des instances d'algorithmes de résolution dans la base de connaissance, comme pour les autres instances, n'impacte pas la structure de l'ontologie. Nous avons hébergé deux algorithmes de résolution, l'algorithme d'identification d'ensemble dominant depuis la centralité de graphe (*CentralDominatingSetWeightlessAlgorithm*) et sa version pondérée (*CentralDominatingSetWeighted1.2Algorithm*). Lors de la description de ces deux algorithmes, nous avons également stipulé qu'ils étaient compatibles avec le service *SurrogateEndToEndEncryptionService* via la propriété *compatibleWithAlgorithm* et son inverse *compatibleWithService*.

4.5.4 Mesures de la base de connaissance

En raison de la taille volumineuse du jeu de données, on distingue nombre d'instances crées qui correspondent majoritairement aux objets de notre graphe. Ces instances sont représentées dans l'ontologie sous la forme d'axiomes déclaratifs. Dans l'outil Protégé [197], les axiomes déclaratifs sont des axiomes créés de manière automatique lors de la saisie de nouvelles classes, propriétés ou individus. Les axiomes logiques quant à eux sont des axiomes qui sont déduits des axiomes déclaratifs.

Dans le cadre de notre graphe, nous déclarons chaque sommets par une instance. Dans cette instance, nous renseignons de multiples informations telles que le type de l'instance (sa classe) ainsi que ses interactions qui correspondent aux arêtes du sommet. Ces informations sont stockées dans notre base de connaissances sous la forme de triplet (sujet,prédicat,objet). Lors de la création d'une nouvelle instance, Protégé va créer un nouvel axiome déclaratif et autant d'axiomes logiques qu'il faut de triplets pour définir notre instance. Soit un graphe $G = (V, E)$, nous allons

représenter ce graphe dans notre base de connaissance en créant autant d'instances que nous avons de sommets dans G ($|V|$ instances). Étant donné que nos sommets représentent des objets IoT, nous déclarons que ces instances appartiennent à la classe *Device* de notre ontologie (en pratique elles appartiennent à une sous-classe de *Device* relative à la catégorie d'objet). Cette déclaration requiert l'ajout dans la base de connaissance d'un triplet du type ($instance\ SubClassOf\ Device$) et donc à l'ajout d'un nouvel axiome logique pour chaque instance ($|V|$ axiomes logiques). Protégé va par la suite interpréter chacune de ces instances sous la forme d'un axiome déclaratif ce qui va générer $|V|$ axiomes déclaratifs. Nous avons défini dans notre ontologie une propriété $interactsWith$ qui représente les arêtes de notre graphe sous la forme d'un prédicat ($Device\ interactsWith\ Device$) et qui relie deux instances de la classe *Device*. Créer un instance pour chaque sommet de notre graphe nécessite donc d'ajouter $|V|$ axiomes déclaratifs et $|V| + |E|$ axiomes logiques à notre base de connaissance. Notre jeu de données correspond à un graphe composé de 1 458 sommets et de 35 657 arêtes. Sa définition dans notre base de connaissances implique donc la création de 1 458 axiomes déclaratifs et de 37 115 axiomes logiques.

Avec exactement 1 527 axiomes déclaratifs et 38 802 axiomes logiques, la base de connaissance recense un volume important d'information. Une base de connaissance volumineuse soulève tout de même quelques préoccupations notamment concernant sa capacité à opérer efficacement dans un contexte IoT [61]. L'ouverture de notre base de connaissance dans l'outil Protégé [197] offre plus de visibilité concernant sa structure et sa composition. La Figure 4.1 reprend quelques métriques illustrant la composition de cette base. Les axiomes cumulent les axiomes logique et non-logiques. Les propriété d'objets relient deux objets par l'intermédiaire d'une propriété alors que les propriétés de données associent un objet à une donnée (exemple : *xsd :boolean* pour une donnée booléenne). Pour finir, le nombre d'individus décrit le nombre d'instances de classes décrites dans l'ontologie.

Nom	Compte
Axiomes	40 343
Axiomes logiques	38 802
Axiomes déclaratifs	1 527
Classes	37
Propriété d'objets	15
Propriété de données	3
Individus	1 470

TABLE 4.1 – Quelques métriques de la base de connaissance (obtenues avec Protégé [197])

En analysant ces données, on distingue que malgré un très grand nombre d'axiomes, la quantité de classes et de propriétés reste très raisonnable. En effet, en regardant

de plus près les données que nous avons saisi dans notre base de connaissance comprennent un jeu de données lui-même important.

Une attention particulière est donc nécessaire à apporter aux données qui sont utilisées pour constituer la base de connaissance au risque de voir sa taille croitre exponentiellement avec des réseaux IoT encore plus importants.

4.6 Conception d'un outil de Comparaison de Solutions de Placement de Services (CSPS)

Nous proposons de compléter notre contribution sur la comparaison des solutions de placement de services par la réalisation d'un outil que nous avons intitulé CSPS (Comparaison de Solutions de Placement de Services). Ce dernier permet, à partir d'une base de connaissance ontologique, d'appliquer les méthodologies proposées précédemment dans cette contribution. Les sections suivantes décrivent le principe de CSPS ainsi que son fonctionnement.

4.6.1 Principe de CSPS

CSPS permet, au travers une base de connaissance définie, de résoudre des problèmes de placement de service à l'aide de plusieurs algorithmes et de comparer les solutions à ces problèmes selon de multiples critères d'évaluation.

Pour ce faire, nos travaux reposent sur la capacité à nous interfacer avec une base de connaissance. Pour ce faire, nous avons utilisé Owlready2 développé par Jean-Baptiste Lamy [162]. Owlready2 est une librairie Python qui permet d'interagir en Python avec une structure ontologique, typiquement un fichier OWL. Pour des questions pratiques, nous avons fait le choix de cette librairie plutôt que son homologue Java, Jena [6], pourtant complet. En effet, la plateforme que nous utilisons pour notre rendu visuel (Dash [7]) ainsi que l'ensemble de nos travaux précédents sont en Python. L'utilisation d'une librairie Python était donc un choix naturel. La Figure 4.8 propose un aperçu de l'interface de CSPS dans lequel la topologie extraite de notre base de connaissance y est illustrée.

Dans la partie gauche de l'interface, illustré en Figure 4.9, nous retrouvons les différentes options de configuration. La partie droite affiche les résultats sous la forme d'un tableau dont les résultats sont repris et affichés dans le tableau Table 4.2.

4.6.2 Base de connaissances

Lors de la réalisation de CSPS, nous avons repris la base de connaissances que nous avions construite en Section 4.5. Cette base reprend notre ontologie FSPlacementOntology décrite précédemment dans laquelle nous avons renseigné les informations du jeu de données SIoT. Ce jeu de données, issu de la plus grande composante

6. `https://jena.apache.org/index.html`
7. `https://plotly.com/dash/`

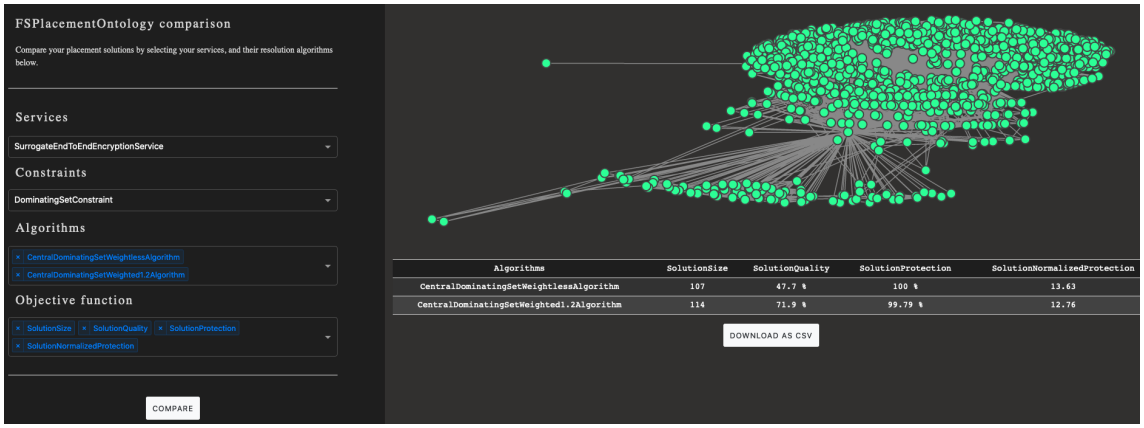


FIGURE 4.8 – Aperçu de l'interface de CSPS.

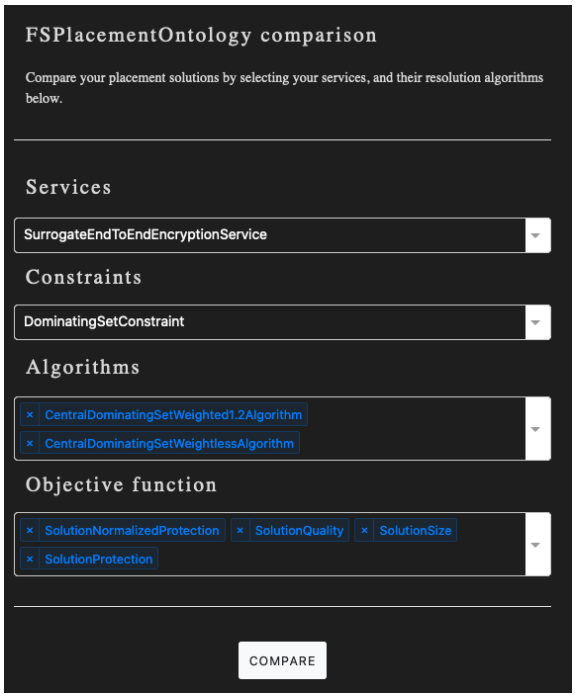


FIGURE 4.9 – Panneau de configuration de CSPS.

connexe du graphe OOR du projet SIoT [25], comporte 1 458 sommets et 35 657 arêtes. Il est évident que l'ensemble des individus représentatifs de ces sommets et leurs relations n'ont pas été renseignés manuellement. L'ensemble des informations relatives au modèle d'infrastructure ont ainsi été renseignées automatiquement dans notre ontologie FSPlacementOntology à l'aide de Owlready2 [162].

De manière plus générale, toutes les interactions entre notre outil et la base de connaissances sont effectuées à l'aide de la bibliothèque Python Owlready2.

4.6.3 Utilisation

L'utilisation de CSPS repose sur les menus déroulants situés sur la gauche de l'interface (Figure 4.9). Lors de l'accès d'un utilisateur sur la page, la base de connaissance est chargée, la topologie du réseau IoT est ensuite extraite et affichée sous la forme d'un graphe.

En même temps, l'ensemble des services contenus dans la base de connaissance sont proposés à la sélection dans le menu de gauche. Lors du choix d'un service, l'indicateur de contrainte est automatiquement actualisé pour faire correspondre la contrainte au service sélectionné. Simultanément, la liste des algorithmes disponibles est mise à jour pour proposer l'ensemble de ceux qui sont compatibles avec le service à analyser.

Par défaut, l'ensemble des fonctions objectives sont proposées à la sélection. De plus, la sélection de plusieurs fonctions objectives est permise de même que pour les algorithmes de résolution.

Le choix de multiples algorithmes de résolution et de critères d'évaluation des solutions offre une plus grande latitude dans l'analyse des solutions proposées. Il est ainsi possible d'identifier quel algorithme de résolution offre de meilleurs résultats et de faire varier cette notion de "meilleur résultat" pour correspondre le plus fidèlement au besoin d'évaluation de l'utilisateur.

Une fois l'ensemble de ces paramètres renseignés et le bouton pressé, un récapitulatif des scores est affiché sous la forme d'un tableau. Ce tableau met en comparaison les résultats des algorithmes de résolution ainsi que les scores obtenus par chaque fonction objective. Finalement, le tableau est accompagné d'une fonctionnalité d'export de données au format CSV pour faciliter la récupération de ces données.

4.6.4 Analyse d'un résultat de comparaison

Nous pouvons observer le résultat d'une comparaison en Table 4.2. Dans cette table, nous comparons les performances de placement de nos deux algorithmes, décrits dans le Chapitre 3, avec les différentes méthodes d'évaluation que nous avons également définies dans ce chapitre.

Algorithmes	Taille	Qualité	Protection	Protection normalisée
CentralDominatingSetWeightless	107	47.7 %	100 %	13.63
CentralDominatingSetWeighted1.2	114	71.9 %	99.79 %	12.76

TABLE 4.2 – Exemple d'une comparaison entre plusieurs algorithmes de résolution et scores de fonctions objectives.

Cette comparaison permet de mettre en correspondance les différents algorithmes et fonctions objectives définis dans notre base de connaissances. Au travers de cette comparaison, le rôle de notre outil est d'aider son utilisateur à prendre une décision sur la méthode de placement de service à adopter. En effet, en fonction des résultats des scores de fonctions objectives, un utilisateur pourra préférer d'utiliser

un algorithme de placement spécifique sachant que les solutions de cet algorithme se rapprochent le plus des attentes de l'utilisateur. Prenons l'exemple des résultats présentés dans la Table 4.2. Si l'objectif de l'utilisateur est de maximiser la *Qualité* d'une solution de placement, ce dernier privilégiera l'utilisation de l'algorithme *CentralDominatingSetWeighted1.2*. À l'inverse, si l'utilisateur n'est pas intéressé par la qualité de la solution et qu'il souhaite privilégier la *Taille* de la solution, l'algorithme *CentralDominatingSetWeightless* sera préféré.

Cette comparaison apporte également une aide au choix des fonctions objectives. Cet aspect n'est pas facilement observable dans la Table 4.2 étant donné que l'ensemble de ces fonctions objectives ont été définies en relation avec les algorithmes illustrés dans la table. Supposons qu'une fonction objective retourne les mêmes scores pour plusieurs résultats d'algorithmes de placement. Il serait alors difficile de départager ces algorithmes en se basant uniquement sur une telle fonction objective. Dans de telles conditions, l'utilisateur peut ajouter d'autres fonctions objectives dans sa comparaison afin de mieux discriminer les solutions des algorithmes de placement et ainsi effectuer son placement de services plus judicieusement.

4.6.5 Intérêt d'utiliser une ontologie

Outre le fait que notre ontologie nous serve de base de connaissances, cette dernière nous offre divers avantages.

Le premier avantage d'utiliser une ontologie dans le cadre de notre outil réside dans la formalisation et dans une meilleure structuration des problèmes de placement de services de sécurité. En effet, comme nous l'avons décrit précédemment, l'utilisation d'une structure sémantique comme une ontologie nous permet de modéliser les problèmes de placement de services avec un formalisme similaire à celui présenté en Section 4.2. Grâce à cette modélisation généraliste, nous sommes en mesure de modéliser divers problèmes dans notre base de connaissances.

En complément des informations introduites dans cette base de connaissances lors de la modélisation, de nouvelles informations peuvent être déduites par l'intermédiaire d'inférences. Le principe d'inférence repose sur l'utilisation de règles de raisonnement et de raisonneurs communément associés aux ontologies. Nous pouvons notamment accentuer ce principe en liant notre ontologie avec d'autres ontologies. Une des qualités majeures des ontologies est qu'elles permettent aux systèmes d'accroitre leur interopérabilité en partageant des structures communes. En associant plusieurs ontologies, il est alors possible d'inférer de nouvelles connaissances depuis leurs bases de connaissances associées. Nous pourrions ainsi, en associant notre base de connaissance à une base de vulnérabilités, détecter quels appareils de notre infrastructure sont vulnérables grâce à l'utilisation de notre raisonneur et de règles de raisonnements.

La grande quantité d'informations que nous pouvons déduire grâce aux inférences est particulièrement intéressante pour la définition des contraintes de placement où nous pourrions, par exemple, définir qu'un service de sécurité doit être positionné sur un appareil qui est considéré comme vulnérable (cas d'un correctif de sécurité) ou un un appareil situé à proximité de ce dernier.

4.7 Conclusion et perspectives de poursuite

Au travers cette contribution nous avons proposé une approche plus généraliste aux problèmes de placement de services dans l'IoT comparé à celle du Chapitre 3. Nous avons proposé une structure généraliste pour modéliser des problèmes de placement de service. Pour ce faire, nous nous sommes appuyés sur des outils sémantiques et plus particulièrement sur les ontologies.

Deux méthodologies basées sur l'utilisation d'une telle structure ont été proposées. La première décrit une procédure de résolution d'un problème de placement de services. La seconde détaille le procédé de comparaison des solutions de placement.

En soutien à ces méthodologies, une ontologie que nous avons appelée FSPlacementOntology a été proposée. Cette ontologie modélise l'ensemble des concepts nécessaires à l'application des deux méthodologies décrites.

Finalement, l'ensemble des apports de cette contribution sont fédérés au sein d'un outil de comparaison de solutions de placement de services de l'IoT.

Cette contribution démontre l'apport de l'utilisation d'une structure sémantique dans la génération des problèmes de placement de service dans l'IoT et plus particulièrement dans la comparaison de leurs solutions.

En dépit des résultats intéressants de l'approche proposée, nous avons identifié une première limitation concernant la taille de la base de connaissance générée. L'étape de comparaison ne s'effectue pas obligatoirement sur des périphériques contraints, cependant la taille de la base de connaissance influence ses performances de fonctionnement. Plus une base de connaissances sera volumineuse, plus il lui sera difficile de répondre aux requêtes en temps raisonnable. Afin de limiter de potentiels surcouts dans les temps de traitement, il serait intéressant de pouvoir limiter la taille de l'infrastructure à décrire dans l'ontologie. Pour ce faire, une approche décentralisée avec une description restreinte aux environnements proches pourrait être envisagée. Une possible méthode consisterait à séparer un réseau en sous-réseaux, lesquels seraient gérés chacun par une entité qui bénéficierait d'une base de connaissance restreinte à son sous-réseau. Une telle approche offrirait plus de souplesse dans la gestion de réseaux volumineux pour proposer un meilleur passage à l'échelle. Afin de garantir l'interaction entre les sous-réseaux, il serait alors essentiel d'étudier la coordination entre les différentes bases de connaissance dans le positionnement de services complémentaires ainsi que l'impact de ce découpage sur la définition du problème de placement. En supposant que nous adaptions notre approche centralisé pour chaque sous-réseau, nous aurions alors une formule pour chaque composante. Nous devrions alors introduire de nouvelles formules pour formaliser les liaisons entre ces composantes et ainsi faire évoluer notre ontologie en conséquence.

Chapitre 5

Ancrage réseau de fonctions cryptographiques white-box

Ce chapitre présente les concepts de la cryptographie white-box, les attaques utilisées à leur encontre et propose une méthode pour se prémunir contre une partie de ces attaques. Notre méthode repose sur l'ancrage d'une implémentation de cryptographie white-box sur son environnement local dans un réseau IoT. Nous décrivons une méthodologie d'ancrage qui repose sur la construction d'une signature locale de graphes. Ce principe d'ancrage est ensuite mis en application auprès d'un exemple simplifié confronté à plusieurs scénarios d'attaques comme l'extraction du code de l'implémentation ou encore le vol de l'appareil. Le mécanisme d'ancrage ouvre des perspectives de sécurisation intéressantes en particulier concernant la prévention des attaques consistant à s'emparer de l'implémentation white-box.

Sommaire

5.1 Introduction 113
5.2 Présentation de la cryptographie white-box 115
5.3 Méthodologie d'ancrage d'implémentation boîte blanche 123
5.4 Exemple d'un ancrage 132
5.5 Conclusion et améliorations 140

5.1 Introduction

Un grand nombre d'objets de l'IoT ne sont pas en mesure de chiffrer leurs données de manière sécurisée. Lorsqu'ils sont en mesure d'effectuer des opérations de chiffrement, ces derniers se retrouvent confrontés aux problèmes de gestion de clé cryptographiques.

Plusieurs approches existent à ce problème, la première repose sur l'utilisation de composants matériels dédiés au stockage et l'utilisation de clés cryptographiques telles que les Secure Elements. Cette approche propose un niveau de sécurité reconnu et souvent certifié. Cependant, il n'est pas toujours possible, pour des raisons budgétaires. Une alternative repose sur l'utilisation de procédures logicielles pour utiliser et surtout stocker les clés cryptographiques avec une sécurité suffisamment importante. Parmi ces méthodes de sécurisation logicielle, on retrouve le principe de cryptographie *white-box*. La cryptographie white box consiste à implémenter des algorithmes de chiffrement dans lesquels la clé de cryptographique est embarquée. Ces données sont alors fortement obfusquées afin de cacher au mieux possible les informations relatives à cette clé.

En règle générale, les implémentations de cryptographie white box sont vulnérables à deux catégories d'attaques. Les attaques par *code-lifting* dans lesquels l'objectif de l'attaquant est d'extraire le code de l'implémentation afin de la réutiliser dans un contexte différent. Et les attaques de récupération de clé. Ces dernières consistent à analyser le code de l'implémentation, son comportement et divers facteurs environnants afin d'extraire des informations sur la clé de chiffrement utilisée et de pouvoir la reconstruire. Les attaques de récupération de clé sont communément perpétrées par des experts dans le domaine de la sécurité dont les compétences sont très importantes. À l'inverse, le profil nécessaire pour effectuer des attaques par extraction de code est amplement moins exigeant. Une méthode fréquente pour se prémunir contre l'extraction de code consiste à lier l'implémentation au dispositif sur lequel elle est exécutée. Par exemple, une implémentation peut être personnalisée pour être utilisée sur un smartphone et être dépendante d'une fonction physique inclonable liée à l'appareil photo de l'appareil [214].

Ces méthodes sont intéressantes, mais se limitent à se prémunir contre l'extraction du code de la *white-box* de non de la fonctionnalité offerte. En effet, si l'on étend le principe de l'attaque par *code-lifting* de la *white-box* à l'extraction de la fonction de sécurité de son contexte normal, cela comprend le déplacement du composant qui héberge la fonction en elle-même. Dans l'exemple mentionné précédemment, cela se résume à subtiliser l'appareil hébergeant cette fonction, à savoir le smartphone, afin de bénéficier de la capacité d'utiliser la fonction de sécurité (en faisant abstraction des protections externes de l'appareil comme son verrouillage).

Cette contribution constitue une étude préliminaire pour étendre l'ancrage des implémentations *white-box* à la topologie d'un réseau pouvant apporter une sécurisation accrue pour un service de sécurité. L'objectif de nos travaux consiste à empêcher qu'un service de sécurité ne soit utilisable dans environnement autre que celui pour lequel il est destiné. Dans un contexte d'IoT, cela comprend le déplacement d'un objet dans un environnement différent.

Dans notre contribution, nous décrivons les principes fondamentaux de la cryptographie *white-box*. Par la suite, nous détaillons notre méthodologie d'ancrage en nous basant sur le postulat de l'existence d'un compilateur d'implémentations *white-box* capable de générer des implémentations dans lesquels un calcul de distance et une comparaison de valeurs peuvent être effectués. Notre méthodologie consiste à construire une signature de l'environnement local d'une implémentation et de saisir

cette donnée de référence lors de sa compilation. Une fois l'implémentation déployée sur leurs objets respectifs, ces implémentations vont requérir que la signature locale des environnements leur soit renseignée lors de chaque appel au programme afin de déterminer si cette nouvelle signature est suffisamment proche de celle de référence. Cette méthodologie est ensuite appliquée à un exemple simplifié comportant plusieurs scénarios d'attaque tels que l'extraction de code ou le vol de l'objet d'accueil. En complément, les apports de notre mécanisme d'ancrage sont discutés.

Le principe d'ancrage local des implémentations *white-box* offre des perspectives intéressantes pour accroitre la sécurité de ces implémentations.

5.2 Présentation de la cryptographie white-box

La notion de cryptographie boîte blanche (*white-box*) a été introduite en 2002 par Stanley Chow [82, 83]. Elle vient de la constatation que les implémentations de la cryptographie conventionnelle (boîte noire) deviennent vulnérables lorsque l'environnement d'exécution est compromis.

Contrairement au contexte en boîte noire où un attaquant doit se contenter d'observer uniquement les données d'entrée et sortie de la boîte, le contexte boîte blanche (ou *whitebox*) suppose qu'un attaquant peut observer l'intérieur de la boîte. L'objectif de la cryptographie boîte blanche est de préserver la sécurité d'un système cryptographique dans un tel contexte et de répondre aux problèmes de stockage des clés cryptographiques. Chow et al. [82] ont d'abord proposé une implémentation de l'algorithme DES dans le contexte *whitebox*. Cette dernière a été compromise peu de temps après sa publication par une attaque introduite par Jacob et al. [146]. Chow et al. ont proposé un second papier [83] où ils proposent une implémentation de AES en cachant la clé à l'intérieur de *lookup tables* tout en encodant ces dernières à l'aide de bijections aléatoires (exemple Figure 5.1). Il est alors difficile pour un attaquant de retrouver la clé de chiffrement. Cette nouvelle implémentation résiste à l'attaque de Jacob et al.

Contrairement à la cryptographie théorique, Chow et al. [83] ne proposent pas une sécurité prouvée, mais offrent une protection accrue dans un environnement purement logiciel. Quelques années plus tard, en 2004, Billet et al. [65] proposent une attaque contre l'implémentation AES de Chow avec une complexité maximale en temps de 2^{30}. À la suite de cela, Karroumi [151] reprend les travaux de Chow et soumet une implémentation utilisant un double chiffrement afin de modifier la représentation des clés à la fin de chaque tour. Cette nouvelle implémentation augmente la complexité maximale de l'attaque de Billet et al. [65] à 2^{91}.

Les implémentations de cryptographie *whitebox* se répartissent principalement en deux catégories présentées par Brecht Wyseur [266] et illustrées dans la Figure 5.2. Dans la première catégorie d'implémentations, dite "à clé fixe" ou *fixed-key*, la clé de chiffrement est codée en dur dans l'implémentation (par exemple incluse dans les *lookup-tables*). La seconde catégorie, plus adaptée à un environnement de production, est à clé dynamique ou *dynamic-key*. Dans cette dernière, une version chiffrée de

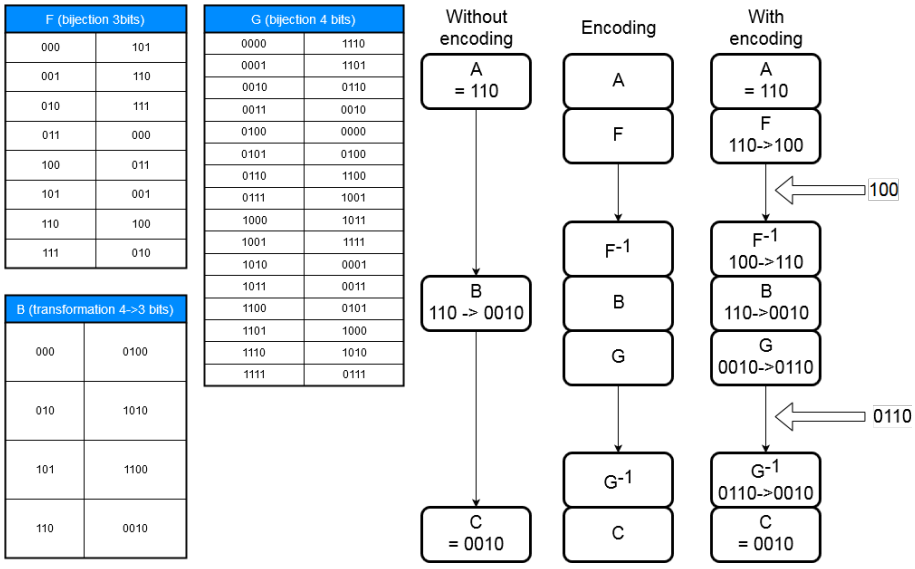
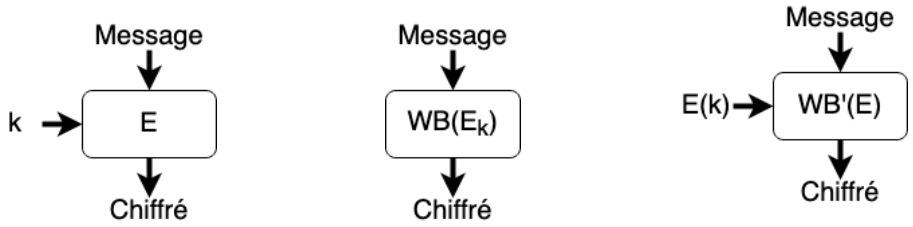


FIGURE 5.1 – Exemple d'encodage via des bijections (F et G) aléatoires.



(a) Chiffrement classique (b) Boîte blanche à clé fixe (c) Boîte blanche à clé dynamique

FIGURE 5.2 – Catégories d'implémentations cryptographiques boîte blanche (source : [266])

la clé est fournie et l'implémentation va déchiffrer la clé et chiffrer / déchiffrer le message avec cette clé.

5.2.1 Attaques sur les implémentations whitebox

Le niveau de sécurité offert par l'utilisation de la cryptographie boîte blanche n'est pas garanti. Sanfelix et al. [230] illustrent ce propos dans leur papier par deux attaques sur des implémentations publiques de cryptographie boîte-blanche. Ces attaques appliquent les mêmes notions que celles par canaux auxiliaires sur un élément matériel, mais ici dans un environnement logiciel. L'attaquant observe la consommation électrique de l'implémentation (DPA) et injecte des fautes (DFA) afin de découvrir de l'information sur la clé cryptographique.

Les implémentations de cryptographie *whitebox* sont principalement exposées à deux vecteurs d'attaques, les attaques par extraction de code (*code lifting*) et les attaques par récupération de clés (*key recovering*) [230, 266].

5.2.1.1 Extraction de code (Code lifting)

L'implémentation *whitebox* étant purement logicielle, un attaquant n'arrivant pas à récupérer la clé cryptographique, s'emparera de l'ensemble de l'implémentation afin d'utiliser les fonctions intégrées de chiffrement ou de déchiffrement. Afin de se prévenir contre ce type d'attaques, il est possible de mettre en place des contre-mesures effectuant du *node locking* (verrouillage de l'implémentation sur un appareil précis) en se basant par exemple sur les *Physical Unclonable Functions* (PUF, dont les premiers travaux remontent à 2000 [173]). Des verrous scientifiques restent cependant à lever pour rendre possible l'intégration dans une implémentation *white-box* de cet ancrage matériel. Il est également possible de tracer l'origine d'une implémentation à l'aide de mécanismes de *software fingerprinting* [185] intégrés à l'encodage interne de l'implémentation [172, 200, 91, 124]. Cette dernière méthode ne prévient pas du *code lifting* mais permet de retrouver la personne à l'origine de la diffusion d'une implémentation de *whitebox* et de sanctionner éventuellement en conséquence.

5.2.1.2 Récupération de clés (Key recovering)

Les attaques de type *key recovering* consistent à récupérer des informations sur la clé de cryptographique utilisée afin de la reconstruire. Le principe de la cryptographie boîte blanche est qu'il ne permet pas de récupérer la clé cryptographique par analyse de la mémoire de l'appareil, il est alors nécessaire d'utiliser d'autres catégories d'attaques.

Attaques par analyses statistiques Les attaques par analyses statistiques consistent à reproduire, sur une implémentation entièrement logicielle, les attaques par canaux auxiliaires habituellement effectués sur des composants *hardware*. Pour ce faire, l'attaquant analyse les données intermédiaires lors de l'exécution de l'implémentation et effectue une analyse statistique pour tenter d'extraire de l'information sur la clé cryptographique. L'attaque d'analyse différentielle de consommation électrique (DPA) proposée par Sanfelix et al. [230] appartient à cette catégorie. Cette dernière consiste à analyser les différences de consommation électrique lors de l'exécution de plusieurs chiffrements. Cela requiert d'être en possession des paires *plaintext* et *ciphertext* pour chaque chiffrement analysé.

Attaques par manipulation de données Les attaques par manipulation de données visent à reproduire les attaques par injection de fautes (DFA) dans un environnement purement logiciel. Elles consistent à modifier les données lors de l'exécution de l'algorithme de chiffrement ou déchiffrement par implémentation *whitebox* afin de comparer les résultats d'une opération correcte et ceux d'une opération perturbée. L'attaquant va analyser les différences entre ces résultats pour tenter d'en déduire de l'information sur la clé de chiffrement utilisée. L'attaque présentée par Jacob et al. [146] ainsi que l'attaque d'analyse différentielle de fautes présentée par Sanfelix et al. [230] rentrent dans cette catégorie.

Attaques par contrôle du flux d'exécution L'attaque par contrôle de flux d'exécution s'assimile également aux attaques par injection de fautes sur les éléments *hard-*

ware. Elle consiste à modifier le flux d'exécution de l'algorithme (sauter des instructions par exemple) afin de faire découvrir de l'information sur la clé cryptographique utilisée.

5.2.2 Différentes notions de la cryptographie boîte blanche

L'obfuscation générale des programmes a connu une multitude de progrès durant les dernières années [116], cependant sa mise en pratique généralisée n'est pas toujours envisageable. En effet, certains cas d'usages ne sont pas compatibles avec les limitations de cette technique en termes de performances [277] et de niveau de sécurisation offert [133]. Étant une forme d'obfuscation, la cryptographie boîte blanche est sujette aux mêmes limitations.

Lors de la construction d'implémentations de cryptographies boîte blanche, certaines notions sont à envisager afin d'offrir et préserver les propriétés de sécurité voulue.

Sens-unique (*One-wayness*)

La notion de sens-unique ou *one-wayness* a été proposée par Delerablée et al. en 2014 [91]. La même année, Biryukov et al. [67] introduisent une notion semblable sous le nom de *strong white box*.

La notion de sens unique consiste à permettre à une implémentation de cryptographie boîte blanche de chiffrer à volonté, mais de ne pas permettre d'effectuer le déchiffrement. Un compilateur capable de satisfaire cette notion possède deux intérêts. Le premier consiste à partiellement se prémunir des attaques par extraction de code. En effet, si un attaquant s'empare de l'implémentation, il ne sera en mesure que de chiffrer ou déchiffrer des messages, mais ne pourra pas effectuer l'opération inverse. Le second intérêt consiste à concevoir un système de chiffrement asymétrique à partir d'implémentations symétriques [149].

Incompressibilité (*Incompressibility*)

Objectif plus réaliste, la notion d'incompressibilité (*incompressibility*) a été introduite par Delerablée et al. [91]. Elle est également appelée *weak whitebox* par Biryukov et al. [67] ou encore *space hardness* par Bogdanov et Isobe [68]. La notion d'incompressibilité consiste à ne pas être en mesure de réaliser un programme à fonction équivalente, dont la taille est inférieure au programme initial. Cela signifie qu'il est difficile pour un adversaire d'extraire une clé cryptographique de taille inférieure à celle de l'implémentation. À titre d'exemple, Delarablée et al. [91] comparent l'implémentation AES de Chow et al. [83] constituée de 800 ko à une clé cryptographique de 128 bits.

Cette notion est particulièrement intéressante dans le cadre de l'Internet des Objets où les capacités de transfert de données sont limitées. Cependant, il semble qu'il n'existe pas aujourd'hui d'implémentations d'algorithmes cryptographiques AES ou DES *whitebox* qui remplissent la propriété d'incompressibilité [109]. Afin de remplir cette propriété, les algorithmes cryptographiques doivent être conçus

avec cette préoccupation tels que l'ont été les algorithmes WhiteKey et WhiteBlock proposés par Fouque et al. [109] dont le respect de cette propriété est prouvé.

Traçabilité (*traceability*)

La traçabilité est une notion particulièrement appréciée de la cryptographie boîte blanche [265]. Le principe consiste à ce qu'un compilateur d'implémentation boîte blanche soit capable de dériver plusieurs implémentations différentes qui introduisent de légères perturbations tout en offrant un fonctionnellement identique [91]. Cette notion justifie grandement l'utilisation de la cryptographie boîte blanche dans les DRMs. En effet, en proposant plusieurs versions d'implémentation boîte blanche traçables pour de déchiffrement d'un bien numérique, il est possible d'identifier quelle version a déchiffré le contenu [215].

Prenons l'exemple d'un film, chaque utilisateur se voit attribué un programme d'accès au film comportant une implémentation boîte blanche traçable. Si le film est mis à disposition publiquement sur internet, il est possible, à partir des perturbations introduites pas l'algorithme de traçage, d'identifier le programme qui a déchiffré ce film et de sanctionner l'utilisateur en conséquence. Cette notion fait écho au *software fingerprinting* abordé en Section 5.2.1.1.

Incassabilité (*Unbreakability*)

La notion d'incassabilité est une notion fondamentale de la cryptographie boîte blanche introduite par les Chow et al. [82, 83]. Le principe de cette notion consiste à ce qu'il soit impossible pour un attaquant de briser le secret de la clé de chiffrement utilisée dans une implémentation de cryptographie boîte blanche [91]. Cette notion est très difficile à satisfaire en pratique. En effet, la moindre fuite d'information concernant la composition de la clé de chiffrement constitue une brèche dans son secret.

5.2.3 Robustesse des implémentations boîte blanche

De nombreuses implémentations boîte blanche ont été mises à l'épreuve lors de deux compétitions : WhibOx contest édition 1 [1] et WhiBox contest édition 2 [2] qui se sont déroulées respectivement lors des conférences CHES 2017 et CHES 2019 (Conference on Cryptographic Hardware and Embedded Systems).

Lors de la première édition, les règles pour qu'une implémentation soit acceptée étaient les suivantes :

— Le code source ne doit pas dépasser 50Mo.

— La compilation doit au plus utiliser 500Mo de RAM.

— La compilation ne doit pas dépasser 100secs.

— L'exécutable ne doit pas faire plus de 20Mo.

1. `https://whibox-contest.github.io/2017/`
2. `https://whibox-contest.github.io/2019/`

— L'exécutable ne doit pas utiliser plus de 20Mo de RAM.

— Chaque appel de fonction doit s'effectuer en moins d'une seconde.

Au cours de cette première édition de WhiBox contest, 94 implémentations ont été proposées. Aucune de ces implémentations n'a résisté jusqu'à la fin de la compétition. Un grand nombre d'implémentations ont été cassées en moins d'une semaine, dont plusieurs en seulement quelques heures après leur publication. L'implémentation qui a durée le plus longtemps a été intitulée *adoring_poitras* par le serveur. Cette dernière a résisté pendant 28 jours avant d'être cassée. Goubin et al. décrivent la procédure qui a permis de compromettre cette implémentation [124].

La seconde édition de cette compétition a bénéficié de 27 implémentations de cryptographie boîte blanche. Sur les 27 implémentations publiées, 16 ont été inversées et seulement 3 n'ont pas été cassé ni inversées. À l'issue de ce concours, seules 3 implémentations ont résisté : *googy_lichterman*, *elegant_turing* et *hopful_kirch*. Ce résultat est à mettre en perspective avec la durée du concours. En effet, ce dernier à duré 158 jours cependant les 3 implémentations ont été publiées respectivement 21, 21 et 24 jours avant la fin du concours. Le fait de ne pas avoir été ni cassées ni inversées ne signifie pas forcément que ces implémentations sont incassables, mais uniquement qu'aucun candidat n'a réussi à compromettre ces implémentations durant le temps imparti.

Les compétitions telles que WhiBox sont particulièrement intéressantes pour évaluer la robustesse des implémentations boîte blanche en raison du profil des candidats. En effet, la compétition étant organisée à l'occasion de la conférence CHES, les candidats bénéficient de compétences très avancées dans le domaine de la cryptographie. Habituellement utilisée dans le domaine de l'industrie, les implémentations de cryptographie boîte blanche sont rarement divulguées au public afin de réduire leurs risques de compromission. Les initiatives telles que WhiBox permettent d'ouvrir ce domaine à la communauté scientifique et de fournir un aperçu de l'état actuel de la technologie.

Bien que certaines implémentations aient résisté à la seconde édition de la compétition, on remarque que la robustesse des implémentations n'est toujours pas garantie. La notion d'incassabilité n'est alors pas satisfaite pour les implémentations boîte blanche connue à ce jour. L'évolution entre la première et seconde édition de WhiBox contest reste encourageante sur l'état de la recherche dans le domaine.

5.2.4 Avantages et inconvénients de la cryptographie boîte blanche

Le modèle boîte noire souvent utilisé en cryptographie, dans lequel l'algorithme de chiffrement ainsi que sa clé sont protégés n'est pas toujours applicable, surtout dans le domaine de l'IoT. Dans ce domaine, la présence d'un élément de sécurité n'est pas toujours possible. De ce fait, une solution de sécurité boîte noire sera contournée en s'attaquant au stockage de la clé. Le *whitebox attack context* (WBAC) introduit pas Chow et al. [82] représente la situation où l'environnement d'exécution est fortement exposé et où le stockage des clés cryptographiques n'est pas sécurisé. L'utilisation de

cryptographie *whitebox* permet d'augmenter le niveau de protection générale dans un tel contexte, fréquent dans le monde de l'IoT. Il est important de mentionner que son utilisation ne permet pas de garantir un fort niveau de sécurité et que le temps d'exécution ainsi que la taille de l'implémentation sont plus élevés.

L'avantage principal de la cryptographie *whitebox* est de pouvoir se passer des mécanismes de stockage sécurisés de clés cryptographiques. Il est ainsi possible de se prémunir de l'ensemble des attaques relatives au stockage de ces clés. L'utilisation de la cryptographie *whitebox* permet de protéger le système contre les menaces liées à la gestion des clés de la taxonomie de Babar et al. [54]. Habituellement, pour se prémunir contre ce type d'attaques, un élément de sécurité matériel est utilisé. L'utilisation de cryptographie *whitebox* permet de se passer d'un tel matériel en permettant un accès plus facile à la sécurité et des coûts de fabrication et de déploiement potentiellement moins élevés. Dans son papier, Marc Joye [149] énumère d'autres avantages de l'utilisation de la cryptographie boîte-blanche. Il met en avant le côté logiciel de la technologie qui permet une distribution, une installation et un renouvellement plus faciles des implémentations, des avantages d'autant plus importants dans le contexte de l'IoT.

La cryptographie *whitebox* possède également quelques inconvénients dont le principal est que la sécurité des implémentations de cryptographie *whitebox* n'est à ce jour toujours pas prouvée. Ainsi, les implémentations industrielles utilisant cette cryptographie se basent sur des solutions propriétaires mettant en œuvre des techniques conservées secrètes. Ces solutions contredisent le principe de Kerckoff en se reposant sur de la sécurité par l'obscurité. En outre, comme déjà indiquée, l'utilisation de cryptographie *whitebox* engendre des contraintes sur les performances du système. Une implémentation de cryptographie *whitebox* nécessitera toujours davantage de puissance du processeur et de mémoire comparée à une implémentation normale.

5.2.5 Utilisations de la cryptographie boîte blanche

De nos jours, les principales utilisations de la cryptographie *whitebox* résident dans la protection de biens numériques via DRM (*Digital Rights Management*) telles que les films, musiques, jeux vidéo, livres numériques, logiciels ou tous autres biens non matériels vendus sur internet. La présence de l'utilisation de cryptographie *whitebox* pour protéger ces services n'est pas facilement vérifiable. On peut noter cependant, de nombreuses solutions industrielles, comme whitebox Cryptography(Arxan), Cloakware(Irdeto), SafeNet(Gemalto), whiteCryption(Intrust) qui utilisent de la cryptographie *whitebox*. D'autres domaines utilisent également les technologies de chiffrement en boîte blanche, c'est le cas du paiement mobile qui utilise la cryptographie *whitebox* dans la protection de certaines transactions [33].

La cryptographie *whitebox* permet de chiffrer ou déchiffrer des données sans avoir les problématiques du stockage de clés. De ce fait, il est envisageable d'utiliser ce type de cryptographie dans une multitude d'usages.

Remplacer AES classique Il est possible de remplacer l'utilisation d'un AES classique et de sa clé par une implémentation *whitebox* à clé fixe. Cette opération

permet de réduire les risques associés au stockage de la clé de chiffrement. Le système peut toutefois rencontrer un ralentissement causé par l'utilisation de la technologie.

Additionner avec un algorithme de chiffrement standard Cho et al. [81] proposent d'utiliser la cryptographie *whitebox* en complément de la cryptographie standard au sein de deux schémas de chiffrement (*F-CTR-WBC double block length* et *UF-CTR-WBC*). Les auteurs affirment qu'une implémentation "hybride" permet d'obtenir des performances en temps plus intéressants (seulement 1.3 fois plus lent que AES) du fait que l'algorithme *whitebox* n'est utilisé que pour chiffrer une petite partie de la donnée.

Améliorer des architectures de sécurité Les algorithmes cryptographiques standards actuellement présents dans certaines structures de sécurité pourraient être remplacés par une implémentation de cryptographie *white-box*. Dans le cas d'OSCAR [260], il serait possible de remplacer le secret partagé par l'*autorization server* et présent dans les objets, par une implémentation *white-box* qui permettrait de chiffrer et déchiffrer la ressource(figure 5.3).

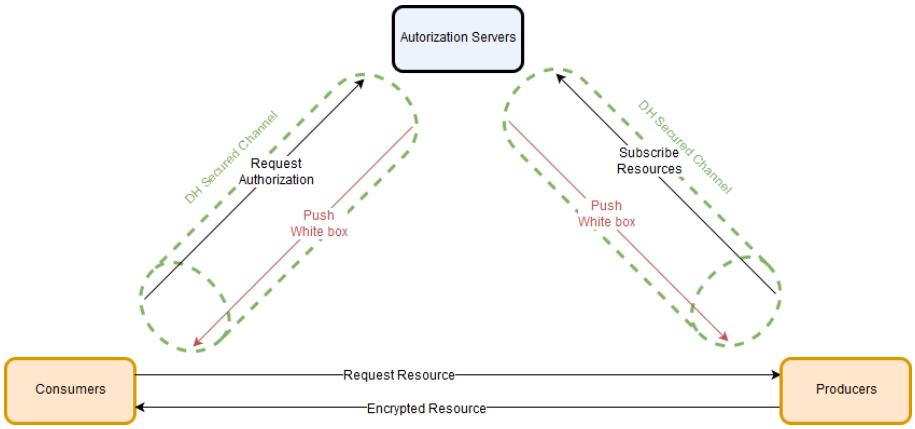


FIGURE 5.3 – Exemple d'utilisation de white box sur l'architecture OSCAR

Signature digitale Beunardeau et al. [63] affirment qu'utiliser une implémentation *white-box* mettant en œuvre un MAC (Message Authentication Code) permettrait à ceux qui possèdent cette implémentation de vérifier une signature MAC. Seul celui qui connait la clé pourra signer les messages.

Cryptographie asymétrique Il est envisageable d'effectuer de la cryptographie asymétrique depuis des primitives symétriques. Biryukov et al. [67] étudient cette possibilité tout en proposant deux implémentations *whitebox* basées sur des schémas ASASA (transformation affine - substitution - transformation affine - substitution - transformation affine).

5.3 Méthodologie d'ancrage d'implémentation boîte blanche

La méthodologie décrite ci-après propose de prémunir une implémentation boîte blanche contre les attaques par extraction de code décrites en Section 5.2.1.1 et consiste à implémenter une méthode d'ancrage de l'implémentation à son environnement d'exécution.

Un mécanisme de protection couramment utilisée dans de telles circonstances consiste à effectuer un verrouillage de l'implémentation sur l'appareil (*node locking*). Notre méthodologie consiste à appliquer ce principe d'ancrage à l'appareil ainsi qu'à son environnement local.

5.3.1 Principe général d'ancrage

L'ancrage d'une implémentation *white-box* sur son environnement local requiert la présence d'une signature locale pour chaque appareil d'un réseau ainsi que la capacité de l'implémentation à comparer ces signatures.

Lors d'une présentation [219], Matthieu Rivain mentionne une catégorie particulière d'implémentation boîte blanche protégée par un mot de passe (Figure 5.4). L'implémentation vérifie si le mot de passe donné en paramètre du programme ($\hat{\pi}$) correspond à celui de référence (π) et si tel est le cas, effectue le chiffrement du message.

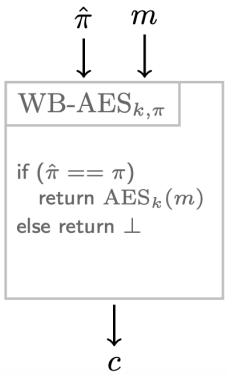


FIGURE 5.4 – Implémentation *white-box* protégée par un mot de passe (source : [219])

Cette structure est particulièrement intéressante en raison de sa capacité à effectuer des opérations de comparaison à l'intérieur de l'implémentation. Nous supposerons, dans la suite de ce chapitre, être en mesure de générer des implémentations de cryptographie boîte blanche aptes à comparer des signatures.

5.3.1.1 Implémentation de comparaison de signatures

Le principe d'une comparaison de signatures consiste à analyser la distance entre deux signatures. Si la distance entre ces signatures est nulle, on parle alors d'égalité

de signatures. Cela correspond au cas présenté en Figure 5.4.

Dans le cas contraire, la distance correspond à la différence qu'il existe entre ces deux signatures. Il est alors possible de déterminer un seuil de distance au-delà duquel deux signatures seront considérées différentes en raison d'une distance trop importante. La Figure 5.5 représente une implémentation boîte blanche qui intègre ces mécanismes de distance.

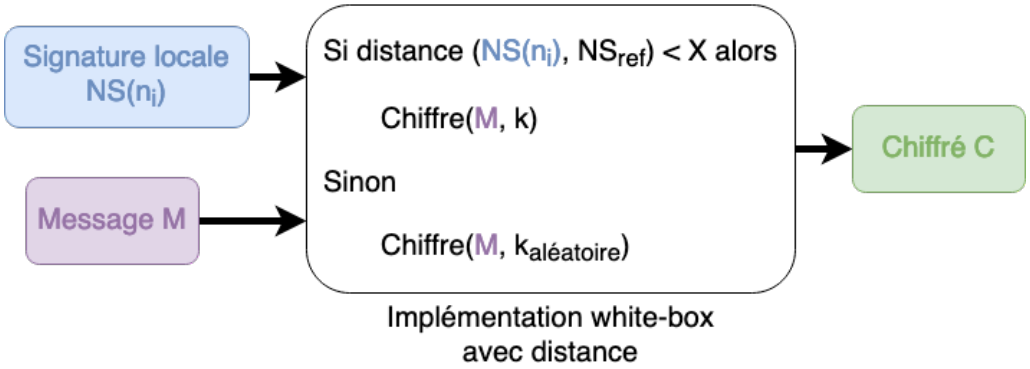


FIGURE 5.5 – Présentation d'une implémentation white-box avec calcul de distance intégré.

Soit $NS(n_i)$ et NS_{ref} deux signatures qui correspondent respectivement à la signature locale de l'objet n_i et la signature de référence. Le programme chiffre un message M avec une clé k si et seulement si la distance entre ces signatures est inférieure à un seuil X. Dans le cas contraire, le programme peut : chiffrer le message avec une clé aléatoire, difficilement réalisable en pratique, autrement le programme peut se terminer et ne pas effectuer le chiffrement du message comme présenté dans le cas de l'implémentation boîte blanche protégée par mot de passe en Figure 5.4.

5.3.1.2 Signature locale d'ancrage

Notre approche s'inspire des travaux de Jouili et al. dans lesquels les auteurs constituent une signature locale d'un graphe $(G = (V, E, A))$ où A comprend les attributs a_i de chaque nœud $n_i \in V$ et $a_{i,j}$ de chaque arête $e_{i,j} \in E$ reliant les nœuds n_i et n_j [148]. Les auteurs représentent une signature locale à l'aide de la formule suivante où $d(n_i)$ correspond au degré du nœud n_i :

$$NS(n_i) = \{\{a_i\}, d(n_i), \{a_{i,j}\}_{\forall n_j \in E}\}. \quad \text{(Signature locale d'après Jouilli et al. [148])}$$

Soit $G = (V, E)$ un graphe non orienté dont V est l'ensemble des noeuds du graphe représentant les objets connectés d'un réseau IoT et E l'ensemble des arêtes représentant les interactions des noeuds. Ainsi soient $v, v' \in V$ tels que ses noeuds sont en interaction, alors l'arête $(v, v') \in E$.

Nous proposons de définir les fonctions suivantes. Soit $N(v)$ une fonction qui retourne l'ensemble des objets v' avec lesquels v interagit

$$\begin{aligned} N : V &\longrightarrow V^* \\ v &\longmapsto \{v' \ : \ \forall (v, v') \in E\}. \end{aligned}$$

Décrivons deux fonctions : $Att_{obj}(v)$ qui fourni l'ensemble des attributs de l'objet v et $Att_{int}(v, v')$ qui contient les attributs de l'interaction entre v et v' (attributs de l'arête $(v, v') \in E$). Définissons l'environnement local d'un objet IoT v par la fonction $Env(v)$ comme l'association des attributs de v et ceux de ses interactions avec ses voisins. Ainsi :

$$Env(v) = (Att_{obj}(v), \{Att_{int}(v, v') \ : \ \forall v' \in N(v)\}).$$

Finalement, nous définissons une fonction $Sign(Env(v))$ qui, d'après la connaissance de l'environnement local d'un objet v, retourne la signature locale de cet environnement dans l'espace des signatures que nous dénoterons Σ. Nous appellerons la résultante de cette fonction, une signature locale de l'objet v. À noter que la connaissance globale des environnement du réseau implique que l'ensemble des environnements locaux soient connus, permettant ainsi de calculer leurs signatures respectives.

Afin de respecter son rôle, le mécanisme de signature locale doit satisfaire les propriétés suivantes.

Propriété 5.3.1. Deux signatures de sommets distincts ne doivent pas être égales

$$\forall v \in V, \nexists v' \in V, v \neq v' \wedge Sign(Env(v)) = Sign(Env(v'))$$

Justification. L'objectif de notre mécanisme de signature est de permettre l'identification d'un environnement local. L'existence d'une collision (même signature pour deux environnements différents) remettrait alors en cause le principe même de ce mécanisme de signature.

Il n'est pas nécessaire d'être en mesure de prouver l'unicité du mécanisme de signature. Il est suffisant de pouvoir faire en sorte que les signatures de sommets distincts ne puissent être égales. En effet, si l'ensemble des signatures sont générées par une même entité, cette dernière peut, lors de la détection d'une collision, décider de modifier son mécanisme de signature en introduisant, par exemple, une once aléatoire. Cette technique est notamment utilisée dans le cas des mots de passe et des fonctions de hachage.

Une autre technique possible pour différencier les signatures de sommets distincts consiste à s'appuyer sur les identifiants des objets de l'IoT. D'après Minerva et al. [188], un système IoT est composé d'éléments (*things*) qui bénéficient d'identifiants uniques. Construire un mécanisme de signature qui utilise ces identifiants semble donc être une bonne idée pour assurer cette propriété.

Propriété 5.3.2. Un objet doit pouvoir calculer sa signature locale

Justification. Chaque appareil du réseau doit, à partir des informations qu'il possède sur son environnement, être en mesure de calculer sa propre signature locale. De la sorte, chaque appareil peut renseigner sa signature locale lors de l'utilisation de l'implémentation *white-box* comme illustré en Figure 5.5.

Propriété 5.3.3. La connaissance de tout le réseau doit permettre de calculer toutes les signatures locales

Justification. Il est nécessaire qu'une entité qui bénéficie de la connaissance globale du réseau (telle qu'un orchestrateur) soit en mesure de calculer chaque signature locale afin d'assurer la génération des implémentations et leur distribution. Cette notion est plus détaillée en Section 5.3.2. De manière plus générale, connaitre tout ou partie du réseau implique la connaissance de plusieurs environnement locaux du réseau. D'après la propriété 5.3.2, cela implique que la connaissance de plusieurs environnements locaux permette de calculer les signatures respectives de ces environnements.

Propriété 5.3.4. La méthode de calcul des signatures doit être déterministe

Justification. Le principe même de notre mécanisme de signature repose sur sa capacité à effectuer des comparaisons de signatures. Leurs calculs doivent être déterministes afin de permettre l'identification d'environnement identique, sans quoi, deux signatures différentes pourraient être attribuées à un même environnement local.

À la fonction de signature $Sign$ est associée une fonction D qui permet de mesurer la distance entre deux signatures. Un seuil de distance d_{thresh} est également défini en complément de D. Soit deux temps t_1 et t_2, nous définissons l'état d'un environnement local $Env(v)$ à un instant t_1 par $Env_{t_1}(v)$.

Propriété 5.3.5. Deux signatures d'un même environnement, à des instants différents, doivent être à faible distance

$$\forall v \in V, D(Sign(Env_{t_1}(v)), Sign(Env_{t_2}(v))) < d_{thresh}$$

Justification. Le mécanisme de signature utilisé s'inspire du domaine de la biométrie et plus particulièrement des empreintes digitales. En biométrie, deux empreintes digitales d'un même doigt provenant du même individu auront une très forte similitude. Ces deux captures ne seront pas forcément identiques en raison des paramètres de capture telles que la présence de micro poussières sur le capteur, modification du positionnement du doigt ou encore de la qualité du capteur. Ce phénomène de distorsion de la donnée est également observable dans le domaine de l'IoT. En effet, un environnement local est potentiellement voué à subir de faibles variations suite à l'aspect dynamique de ces réseaux. Un objet peut se joindre à l'environnement comme d'autres peuvent disparaitre (déplacement ou désactivation par exemple). Prendre en compte une légère variation dans ces environnement permet leur identification malgré ces légères variations.

Propriété 5.3.6. Deux signatures d'environnements différents doivent être différentes avec une distance élevée

$$\forall v, v' \in V \Rightarrow D(Sign(Env(v)), Sign(Env(v'))) > d_{thresh}$$

Justification. En biométrie, deux empreintes digitales provenant de deux individus différents auront une distance plus importante que dans le cas mentionné précédemment. Dans le domaine de l'IoT, ce phénomène est également présent lors de l'analyse d'environnements différents. Notre mécanisme de signature doit ainsi considérer de telles circonstances. À noter cependant que, lorsque la variation entre deux environnements à des instants différents d'un même objet est trop importante, ces environnements peuvent être considérés comme différents. C'est typiquement le cas lorsque un objet est déplacé et que l'ensemble de ses voisins ont changés.

L'intérêt de notre approche est que, d'après l'ensemble des propriétés décrites précédemment, chaque objet est en mesure de calculer sa propre signature depuis les informations qui lui sont accessibles. En complément, une entité connaissant l'ensemble ou partie des informations du réseau est également en mesure de calculer les signatures des environnements connus.

Ainsi, nous représentons une signature locale $NS(v)$ d'un objet v à l'aide de l'équation suivante :

$$\begin{aligned} NS(v) &= Sign(Env(v)) \\ &= Sign(Att_{obj}(v) \cup \{Att_{int}(v, v') \ : \ \forall v' \in N(v)\}). \end{aligned}$$

Cette représentation a la particularité de généraliser l'approche proposée par Jouili et al. [148]. En effet, les fonctions $Att_{obj}(v)$ et $Att_{int}(v, v')$ permettent de représenter respectivement les attributs des objets et de leurs interactions. Les auteurs effectuent cette représentation au travers de leurs variables $a_i eta_{i,j}$. Les degrés des nœuds du graphe pouvant, par la suite, être considérés comme des attributs des nœuds ou directement depuis notre fonction N puisque le degré d'un nœud correspond à la taille de N soit $|N|$.

5.3.1.3 Ancrage d'une implémentation

Le principe de l'ancrage est de prévenir des attaques par extraction de code. Cet ancrage est effectué de manière locale sur l'environnement proche de l'objet. Pour ce faire, nous nous utilisons le mécanisme de signature locale présenté en section précédente. Ce mécanisme prend en considération les attributs du nœud et de ses arêtes adjacentes. Dans notre contexte d'IoT, cette signature comprend les attributs de l'objet qui héberge l'implémentation et les attributs des interactions avec ses voisins directs. Ces informations sont alors condensées pour former une signature de l'environnement local de l'objet qui héberge l'implémentation *white-box* comme illustrée dans la Figure 5.6 et décrit en Section 5.3.1.2. On distingue la capacité de l'objet à calculer sa propre signature locale depuis les informations qui lui sont accessibles.

Lors de chaque appel au programme de chiffrement, la signature locale est renseignée et comparée avec une signature de référence enregistrée dans l'implémentation boîte blanche lors de sa compilation.

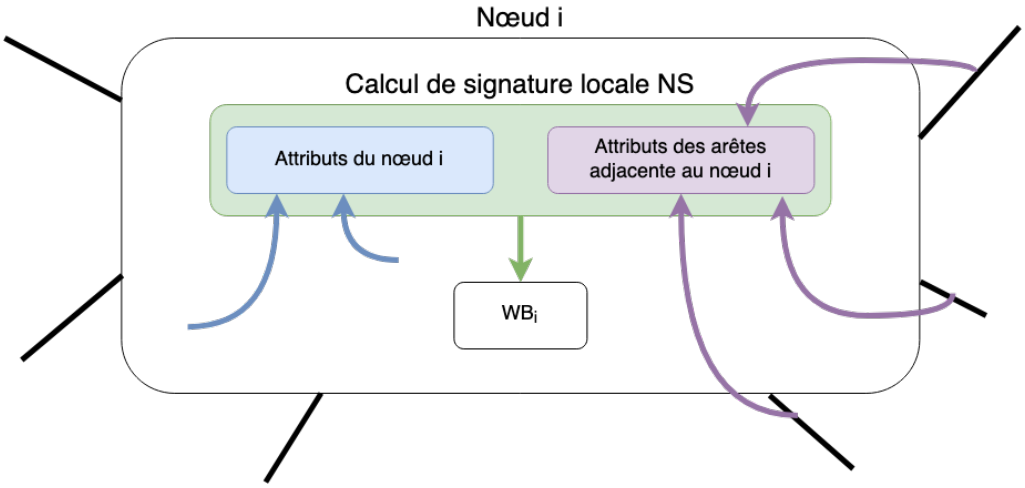


FIGURE 5.6 – Ancrage local d'une implémentation white-box.

Au cours d'une attaque par extraction de code, seule l'implémentation *white-box* est subtilisée. Cette implémentation est ensuite réintroduite dans un nouvel environnement dont les attributs de l'hôte, des interactions avec ses voisins directs et leurs nombres peuvent fortement différer de l'environnement de référence.

Cette forte différence se reflète ainsi lors de la comparaison entre la signature attendue par le programme et celle renseignée lors de son appel.

La particularité de notre mécanisme d'ancrage réside dans sa capacité à prémunir également contre le rapt de l'objet dans son ensemble (implémentation *white-box* et son hôte).

En effet, si un objet qui bénéficie d'une implémentation *white-box* ancrée est subtilisé et introduit dans un nouvel environnement, le mécanisme de signature locale permet l'identification de la situation. Étant donné que l'implémentation n'a pas été extraite de son hôte, les attributs de l'objet, composants de la signature locale de l'objet, correspondent avec ceux utilisés dans le calcul de la signature de référence de l'implémentation *white-box*. Il en résulte alors une signature locale de l'environnement plus similaire à la signature de référence que dans le cas d'une attaque par extraction de code.

La différence entre ces signatures reste cependant identifiable étant donné que l'environnement autour de l'objet est différent. Le nombre de voisins directs et les attributs des interactions de l'objet hôte avec ses voisins sont différents de l'environnement de référence.

5.3.2 Génération et déploiement des implémentations

Une notion particulièrement importante pour le bon fonctionnement de notre mécanisme d'ancrage s'appuie sur la capacité des signatures locales d'environnements à être calculables de manière globale. La méthode présentée en section 5.3.1.2 repose sur l'utilisation des attributs des objets et les attributs des interactions adjacentes à ces objets. La connaissance, par l'orchestrateur, de l'ensemble des objets, de leurs

interactions ainsi que leurs attributs respectifs permet à cette entité de calculer l'ensemble des signatures locales pour chaque nœud du graphe.

Le processus de génération d'implémentation *white-box* et leurs déploiements débutent par l'identification des appareils qui requièrent le déploiement de ces programmes. Soit N l'ensemble des nœuds pour lesquels un tel déploiement est requis, pour chaque nœud de cet ensemble, sa signature locale est calculée. Cette signature est ensuite renseignée auprès du compilateur d'implémentation cryptographique *white-box* en complément d'un seuil de distance. Le seuil de distance correspond à la limite au-delà de laquelle deux signatures seront considérées trop distantes. Cela signifie que si la distance entre deux signatures est supérieure à ce seuil, le programme considère que l'environnement diffère de celui de référence et n'effectue pas son opération normale de chiffrement. La Figure 5.5 représente ce principe dans lequel la variable X correspond au seuil de distance. Une fois ces deux informations en possession du compilateur d'implémentations *white-box*, une clé de chiffrement doit lui être renseignée. Cette clé est utilisée pour le chiffrement effectué dans l'implémentation. La majorité des implémentations de cryptographie white-box reposant sur des algorithmes de chiffrement symétriques, cette procédure permet à l'orchestrateur de générer deux (ou plus si besoin) implémentations *white-box* avec des ancrages différents dont les fonctions cryptographiques sont identiques. De la sorte, les hôtes de ces implémentations sont en mesure de chiffrer et dé-chiffrer respectivement leurs messages à l'aide de ces implémentations. La Figure 5.7 illustre la procédure de génération et personnalisation d'implémentation *white-box*.

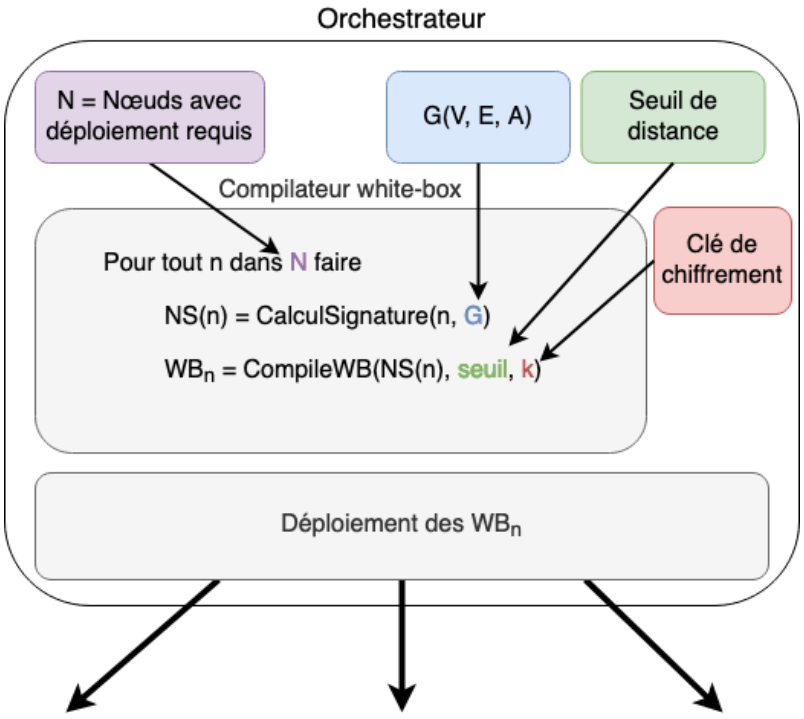


FIGURE 5.7 – Principe de génération d'une implémentation white-box ancrée.

L'orchestrateur déploie ensuite ces implémentations de la même sorte qu'il le

ferait lors de déploiement de services conventionnels avec une attention particulière à respecter les hôtes pour lesquels les implémentations ont été générées.

5.3.3 Fonctionnement de l'ancrage

Une fois les implémentations white-box déployées sur les appareils du réseau, ces derniers sont en mesure de faire appel aux fonctionnalités de chiffrement / déchiffrement de l'algorithme cryptographique présent dans les implémentations afin de sécuriser leurs communications.

Lorsqu'un appareil souhaite communiquer un message de manière sécurisé, il calcule la signature locale de son environnement et la donne, en complément de son message, dans l'appel au programme *white-box*. Si la distance entre la signature renseignée et celle de référence du programme n'est pas trop importante, le message est chiffré correctement, et peut ensuite être communiqué vers son destinataire.

Le destinataire du message est confronté à deux cas possibles. Dans le premier cas, illustré en Figure 5.8, le destinataire possède une copie de l'implémentation *white-box* ainsi que la connaissance totale ou partielle du réseau lui permettant de calculer la signature locale de l'environnement de l'implémentation. Ce cas exige d'être en mesure de conserver une représentation du réseau ou au moins une partie de ce dernier, ce qui n'est pas systématiquement possible pour chaque appareil.

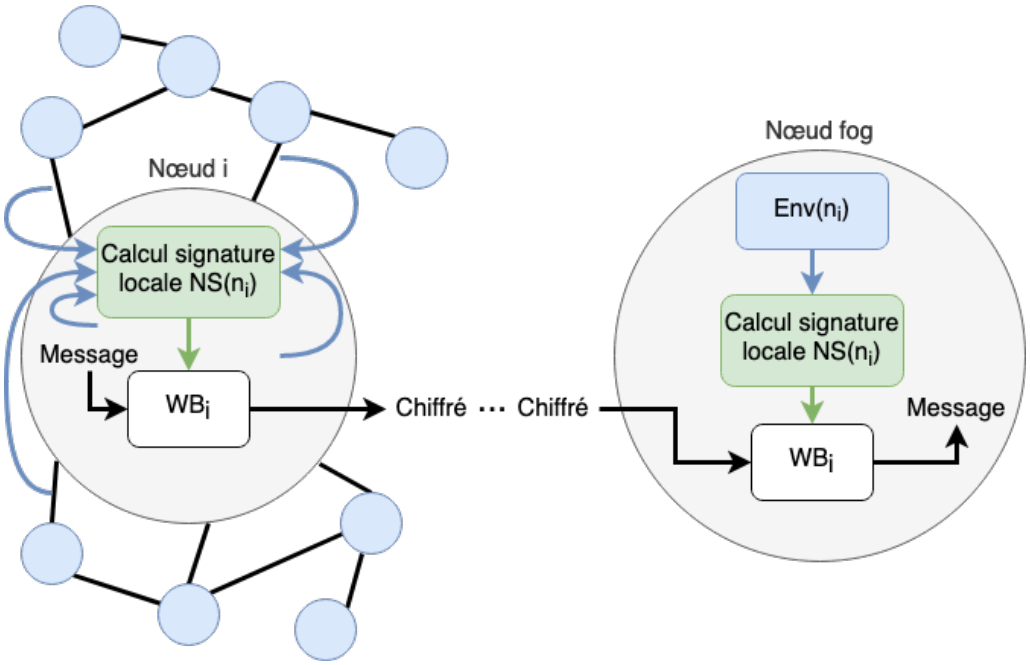


FIGURE 5.8 – Présentation du fonctionnement de l'ancrage d'implémentation white-box avec partage du même programme.

Afin de remédier à ce problème, un second cas est possible. Ce dernier consiste à faire bénéficier le destinataire d'une implémentation white-box dont l'ancrage est effectué sur l'appareil destinataire tout en partageant la même clé de chiffrement

qu'utilisée dans l'implémentation de chiffrement du message. La Figure 5.9 illustre ce cas.

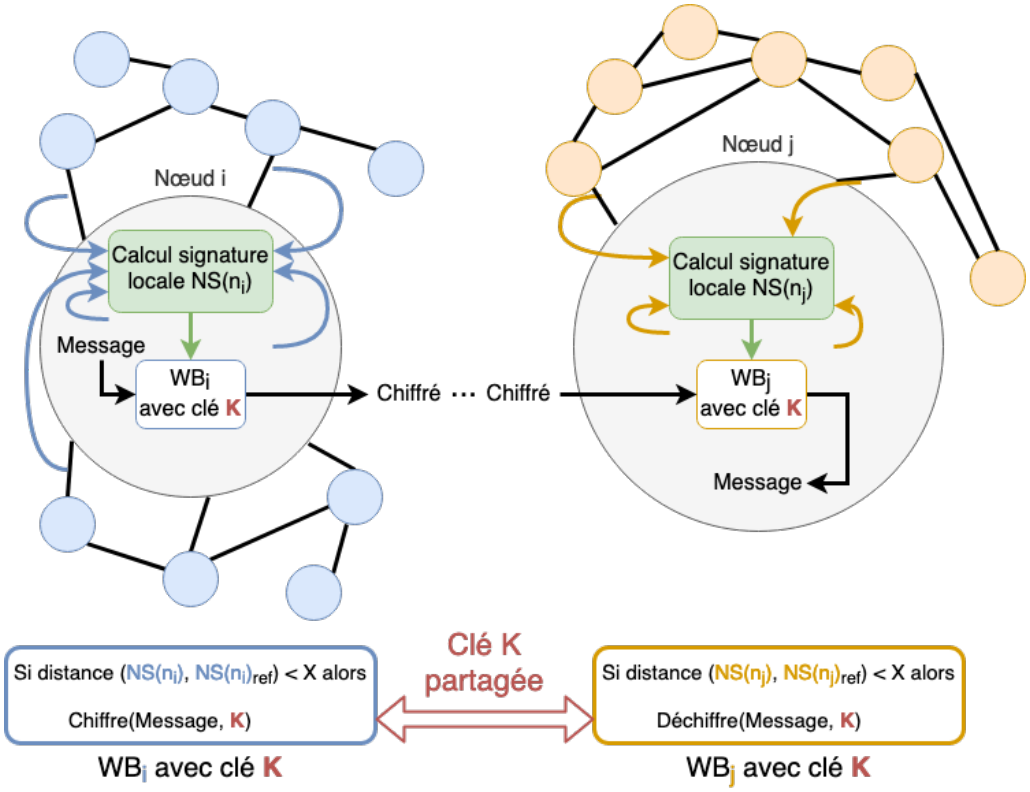


FIGURE 5.9 – Présentation du fonctionnement de l'ancrage d'implémentation white-box avec partage de la même clé de chiffrement.

Le second cas est particulièrement intéressant de par sa souplesse dans la gestion de clés de chiffrement et permet également de prévenir les attaques par extraction de code, et vol de l'objet pour les noeuds destinataires contrairement au premier cas.

5.4 Exemple d'un ancrage

Cette section illustre le principe d'un ancrage sur un exemple simplifié et démontre plusieurs scénarios d'attaques que le mécanisme d'ancrage présenté permet de prévenir.

5.4.1 Présentation du réseau

Pour notre exemple, nous avons défini un réseau simplifié composé de sept objets connectés. Ces objets sont représentés sous la forme d'un graphe dans lequel les sommets correspondent aux objets et les interactions entre objets sont illustrées par les arêtes. L'ancrage de l'implémentation *white-box* est effectué sur le nœud 1 mis en évidence dans la Figure 5.10.

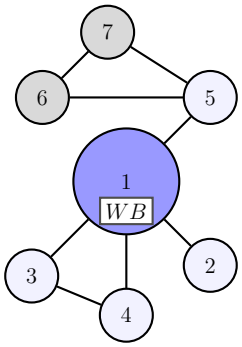


FIGURE 5.10 – Topologie du réseau initial

Par choix de simplification, nous avons limité les attributs des nœuds à leurs identifiants. De la même sorte, les attributs des arêtes correspondent aux identifiants de leurs extrémités. Bien évidemment, dans un contexte plus réaliste, ces attributs pourraient correspondre aux caractéristiques des objets (CPU, RAM, modèles des composants ...), mais également celles de leurs interactions (interface de communication, versions, protocoles, adresses MAC ...).

5.4.2 Construction d'une signature

Nous avons décrit en Section 5.6 les données requises pour la constitution d'une signature locale. Elles sont constituées des attributs de l'objet pour lequel la signature est calculée ainsi que ceux de ses interactions avec ses voisins. Dans le cas de notre exemple, les données requises pour former une signature sont l'identifiant de l'objet et les identifiants des extrémités de ses arêtes, ce qui correspond ainsi à fournir les identifiants de ses voisins. Visuellement, cela revient à renseigner les informations l'environnement local de l'objet, illustré en vert dans la Figure 5.11. Il est important de préciser que les attributs des voisins ne sont pas considérés dans la signature, seules les informations relatives aux interactions avec ceux-ci sont prises en compte. Cette approche considère uniquement les informations que l'objet dont la signature doit être calculé est en mesure de connaitre.

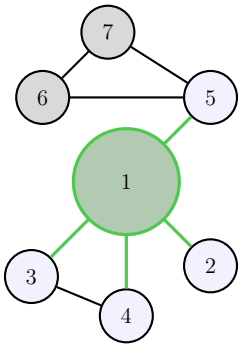


FIGURE 5.11 – Environnement local (vert) du noeud 1.

La signature locale de l'environnement local d'un objet est calculée à l'aide de la fonction $Sign$ telle que décrite en Section 5.3.1.2. Pour simplifier notre exemple, nous supposons que cette fonction prend en entrée les informations de l'environnement local d'un objet et retourne un vecteur composé de l'identifiant de cet objet suivi des identifiants de ses voisins triés par ordre croissant. Ainsi, dans le cas du nœud 1 de notre exemple, sa signature est la suivante :

$$\begin{aligned} NS(1) &= Sign(Env(1)) \\ &= Sign(Att_{obj}(1), \{Att_{int}(1,v) \ : \ \forall v \in N(1)\}) \\ &= Att_{obj}(1) \cup_{v \in N(1)} Att_{int}(1,v) \\ &= Att_{obj}(1) \cup Att_{int}(1,2) \cup Att_{int}(1,3) \cup Att_{int}(1,4) \cup Att_{int}(1,5) \\ &= \{1\} \cup \{2,3,4,5\} \\ &= \{1,2,3,4,5\} \end{aligned}$$

Notons que nous préservons la distinction entre attributs de l'objet Att_{obj} et les attributs des interfaces Att_{int} pour des questions de lisibilité lors de notre calcul de distance entre les signatures.

5.4.3 Distance entre signatures

Soit deux signatures locales $NS(n_i)$ et $NS(n_i)_{ref}$ qui correspondent respectivement aux signatures de l'environnement du nœud n_i et sa référence. Si l'on représente ces signatures sous la forme d'ensembles, la distance entre ces deux signatures correspond à la taille (cardinal) de leurs différences symétriques également notée $\triangle$. La Figure 5.12 représente visuellement ce principe.

Une distance au sens mathématique doit satisfaire des conditions particulières. Dans notre cas, il faut que, pour une fonction de distance D et trois ensembles A, B et C, les conditions suivantes soient satisfaites.

1. $D(A,B) = D(B,A)$
2. $D(A,B) = 0 \Leftrightarrow A = B$

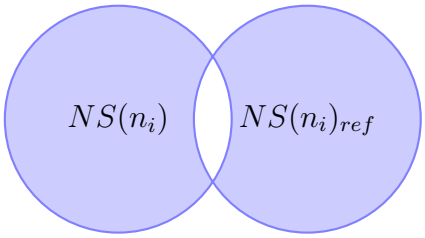


FIGURE 5.12 – Représentation visuelle de la différence symétrique (bleu) entre deux signatures locales pour un nœud n_i.

3. $D(A,C) \leq D(A,B) + D(B,C)$.

Nous démontrons en annexe C que le cardinal de la différence symétrique entre deux ensembles constitue effectivement une distance au sens mathématique.

La mesure de distance dépend grandement du mécanisme de signature utilisé. Pour cet exemple, nous avons fait le choix de simplifier une signature par l'ensemble des sommets qu'elle contient. Lors du calcul de la distance entre deux signatures, une distinction pourrait être effectuée entre les attributs des objets Att_{obj} et les attributs des interactions Att_{int} afin de privilégier certains attributs par rapport au autres. Dans le cas de notre exemple, pour illustrer nos travaux, nous avons fait le choix de la simplicité. Par exemple, dans le cas du sommet 1, sa signature $NS(1)$ devient alors l'ensemble des sommets $\{1\} \cup \{2,3,4,5\}$.

Soit $NS(a)$ et $NS(b)$ deux signatures d'environnement locaux composés de la manière suivante :

$$NS(a) = \{Att_{obj}(a)_1, ..., Att_{obj}(a)_n\} \cup \{Att_{int}(a,x)_{\forall x \in N(a)}\}$$
$$NS(b) = \{Att_{obj}(b)_1, ..., Att_{obj}(b)_m\} \cup \{Att_{int}(b,y)_{\forall y \in N(b)}\}.$$

Nous définissons notre fonction de distance D qui calcule le cardinal de la différence symétrique entre deux ensembles. Pour ce faire, nous calculons la différence symétrique entre les deux ensembles formant les deux signatures. La distance est alors mesurée d'après la taille de cet ensemble. L'équation suivante illustre ce calcul.

$$\begin{aligned} NS(a) &= \{Att_{obj}(a)_1, ..., Att_{obj}(a)_n\} \cup \{Att_{int}(a,x)_{\forall x \in N(a)}\} \\ NS(b) &= \{Att_{obj}(b)_1, ..., Att_{obj}(b)_m\} \cup \{Att_{int}(b,y)_{\forall y \in N(b)}\} \\ D(NS(a), NS(b)) &= |NS(a) \triangle NS(b)| \end{aligned}$$

Prenons maintenant l'exemple de deux signatures $NS(1)$ et $NS(5)$; la distance entre ces deux signatures ($D(NS(1), NS(5))$) se calcul de la manière suivante.

$$
\begin{aligned}
NS(1) &= \{1\} \cup \{2,3,4,5\} \\
NS(5) &= \{5\} \cup \{1,6,7\} \\
D(NS(1), NS(5)) &= |NS(1) \triangle NS(5)| \\
&= |\{1,2,3,4,5\} \triangle \{5,1,6,7\}| \\
&= |\{2,3,4,6,7\}| \\
&= 5
\end{aligned}
$$

La méthode de calcul de distance entre signatures doit être choisie minutieusement afin qu'elle soit compatible avec les propriétés du mécanisme de signature. Le principe d'ancrage s'applique alors sur ce mécanisme par l'intermédiaire d'un seuil de distance. Introduit en Section 5.3.2, ce seuil détermine si deux environnements sont suffisamment proches pour être considérés égaux. L'ancrage s'appuie sur une comparaison par rapport à un seuil distance et non une égalité en raison de la variation potentielle de ces réseaux dans le temps. Un cas normal d'utilisation est détaillé à la section suivante.

5.4.4 Scénarios d'attaque

Le principe d'ancrage d'implémentations *white-box* offre de nombreux atouts concernant la prévention contre les attaques par extraction de code, mais également contre le vol de l'objet physique. Dans les sections suivantes, nous décrivons quelques scénarios d'attaques tels que l'extraction de code, le vol du périphérique, mais également son remplacement en relation avec notre exemple initial. Nous calculons les distances entre leurs environnements respectifs et discutons de l'apport du mécanisme d'ancrage dans de telles circonstances. Avant de décrire ces scénarios d'attaque, décrivons celui d'un fonctionnement normal.

Le fonctionnement normal du mécanisme d'ancrage signifie que la distance calculée entre la signature de l'environnement identifié est relativement proche de celle de référence. Prenons l'ancrage sur l'objet 1 de notre exemple dont la topologie est illustrée en Figure 5.10. La signature de référence saisie lors de la compilation de l'implémentation *white-box* ancrée sur l'objet 1 correspond à l'ensemble $NS(1)_{ref} = \{1\} \cup \{2,3,4,5\}$.

Lors de l'appel au programme de chiffrement ancré, une nouvelle signature de l'environnement local de l'objet 1 est calculée et renseignée en complément du message à chiffrer auprès de l'implémentation *white-box*. Les réseaux et en particulier les réseaux de l'IoT sont sujets à des variations dans leurs environnements. Ces variations sont causées par l'apparition de nouveaux périphériques, leurs déplacements ou alors leur disparition. De ce fait, l'état d'un réseau IoT au temps t_1 risque de ne pas être identique à celui au temps t_2. La Figure 5.13 illustre les changements d'une telle variation sur l'environnement local de l'implémentation *white-box* ancrée. Dans cette nouvelle topologie de réseau, on observe la disparition de l'objet 4 et donc également

des interactions de ce dernier avec le reste du réseau. On remarque également l'apparition d'un nouveau périphérique dans le réseau, l'objet 8 qui interagit avec l'hôte de notre implémentation.

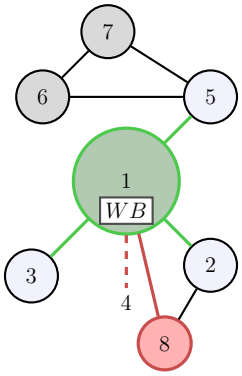


FIGURE 5.13 – Scénario d'ancrage avec faible variation de signature locale.

Calculons maintenant la distance entre la signature de l'environnement local de référence ($NS(1)_{ref}$) et ce nouvel environnement ($NS(1)$).

$$\begin{aligned} NS(1)_{ref} &= \{1\} \cup \{2,3,4,5\} \\ NS(1) &= \{1\} \cup \{2,3,5,8\} \\ D(NS(1)_{ref}, NS(1)) &= |NS(1)_{ref} \triangle NS(1)| \\ &= |\{1,2,3,4,5\} \triangle \{1,2,3,5,8\}| \\ &= |\{4,8\}| \\ &= 2 \end{aligned}$$

Nous observons une distance de 2 encore ces deux signatures. Cela signifie qu'une petite partie de l'environnement a varié cependant la majorité de ce dernier reste stable. Afin que le mécanisme d'ancrage reste fonctionnel, il est nécessaire que le seuil de distance renseigné lors de la compilation de la *white-box* soit strictement supérieur à cette valeur. Définir un seuil de distance trop faible, augmente la sécurité du mécanisme d'ancrage cependant cela risque de réduire la tolérance à la variation de l'environnement et ainsi induire de multiples faux rejets (environnement légitime rejeté par le mécanisme).

5.4.4.1 Extraction de l'implémentation *white-box*

Le principe d'une attaque par extraction de code est de séparer l'implémentation white-box du reste de code associé au périphérique. De la sorte, un attaquant est apte à extraire cette portion de code pour la réintroduire auprès d'un autre périphérique.

Dans ce scénario, illustré en Figure 5.14, l'attaquant extrait l'implémentation *white-box* initialement présente dans l'objet 1 pour l'introduire dans un nouveau réseau et plus spécifiquement sur l'objet 9. De manière plus imagée, ce scénario

peut correspondre à l'extraction d'une implémentation *white-box* hébergée auprès d'un serveur d'une entreprise et réintroduite sur le périphérique d'une entreprise concurrente.

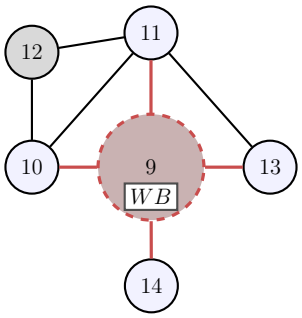


FIGURE 5.14 – Scénario d'ancrage avec implémentation subtilisé.

Visuellement on remarque dans la Figure 5.14 que l'environnement local de l'implémentation *white-box* diffère fortement de celui de référence en Figure 5.10. Le calcul suivant quantifie cette différence.

$$\begin{aligned} NS(1)_{ref} &= \{1\} \cup \{2,3,4,5\} \\ NS(9) &= \{9\} \cup \{10,11,13,14\} \\ D(NS(1)_{ref}, NS(9)) &= |NS(1)_{ref} \triangle NS(9)| \\ &= |\{1,2,3,4,5\} \triangle \{9,10,11,13,14\}| \\ &= |\{1,2,3,4,5,9,10,11,13,14\}| \\ &= 10 \end{aligned}$$

Le calcul de distance entre la signature de l'environnement actuel et celle de référence renseignée lors de la compilation de la *white-box* confirme nos impressions à l'observation de la Figure 5.14. En fixant notre seuil de distance à 3, notre implémentation ancrée serait en mesure de détecter que l'environnement diffère fortement de celui de référence et préviendrait alors son utilisation.

Bien évidemment, cette illustration n'est qu'un exemple simplifié. En pratique, un attaquant pourrait capter la signature locale de l'environnement de l'implémentation avant de l'extraire. Ainsi, il serait en mesure de rejouer cette signature pour contourner ce mécanisme de sécurité. De manière similaire, si un attaquant ne se contente pas de subtiliser l'implémentation, mais extrait l'ensemble de la topologie locale de cette dernière, les objets 1,2,3,4 et 5 dans notre exemple, ce dernier peut contourner l'ancrage.

Il est important de rappeler que les implémentations *white-box* ne sont pas prouvées sécurisées. Notre objectif n'est pas de prévenir toute forme d'attaque par extraction de code, mais uniquement de complexifier l'exécution d'une telle attaque. L'identification d'une implémentation *white-box* n'est pas une opération aisée. Cela demande de grandes connaissances en cryptographie et dans les algorithmes de

chiffrement utilisés dans l'implémentation. Une attaque plus facile à réaliser consiste simplement à s'emparer de l'objet qui héberge l'implémentation.

5.4.4.2 Hôte subtilisé

L'attaque alternative à l'extraction de code consiste à subtiliser l'implémentation ainsi que l'objet qui l'héberge. Cette attaque, plus facile à réaliser, est illustrée dans la Figure 5.15. Dans ce scénario, l'objet 1 est dérobé avec son implémentation *white-box* et introduit dans un nouveau réseau.

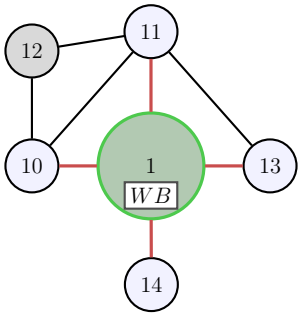


FIGURE 5.15 – Scénario d'ancrage avec hôte subtilisé.

Pareillement au scénario précédent, on remarque une forte différence entre la topologie du réseau de référence et celle de ce nouveau réseau. Cependant, à l'inverse de la situation précédente, l'hôte n'a pas changé ce qui suggère que la distance devrait être moins importante.

$$\begin{aligned} NS(1)_{ref} &= \{1\} \cup \{2,3,4,5\} \\ NS(1) &= \{1\} \cup \{10,11,13,14\} \\ D(NS(1)_{ref}, NS(1)) &= |NS(1)_{ref} \triangle NS(1)| \\ &= |\{1,2,3,4,5\} \triangle \{1,10,11,13,14\}| \\ &= |\{2,3,4,5,10,11,13,14\}| \\ &= 8 \end{aligned}$$

Avec une distance de 8 comparée à une distance de 10 lors de l'extraction de l'implémentation, on distingue une légère réduction des valeurs. Dans le cas de notre exemple, cette réduction n'altère pas la sélection du seuil de distance. Dès lors que ce seuil est inférieur à 8, les deux attaques décrites sont alors identifiées par le mécanisme d'ancrage.

Dans un tel scénario, les mécanismes d'ancrage qui reposent uniquement sur l'objet qui accueille l'implémentation *white-box* sont remis en cause. En effet, les solutions de *node locking* telles que celle proposée par Rasoamiaralanana [214] sont particulièrement efficaces pour prévenir l'extraction du code d'un objet cependant elle n'adresse pas les problèmes relatifs au potentiel vol du périphérique.

Le mécanisme d'ancrage sur l'environnement local de l'objet permet d'identifier les scénarios dans lesquels l'objet hôte est subtilisé. Bien évidemment, le cas présenté est idéal. En pratique, la distinction entre signature de référence et signature obtenue peut être inférieure si le nombre d'interactions des hôtes est fortement réduit. De plus, cette distance peut être davantage réduite si l'attaquant subtilise également d'autres objets du réseau initial, ce qui complique tout de même la réalisation de l'attaque.

5.4.4.3 Remplacement de l'hôte

Le remplacement de l'hôte est un scénario particulier. En effet, l'attaquant extrait l'implémentation *white-box* de l'objet, le remplace par un autre périphérique pour y ré-introduire l'implémentation. À première vue, et d'après l'illustration en Figure 5.16, seul l'hôte de l'implémentation a été altéré.

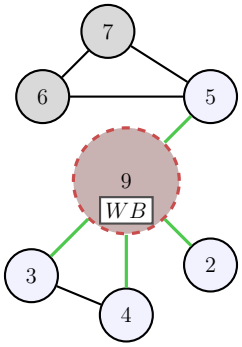


FIGURE 5.16 – Scénario d'ancrage avec remplacement de l'hôte.

En dépit d'une différence conséquente entre l'environnement de référence et celui calculé, notre méthode de calcul de distance des signatures atteste le contraire.

$$
\begin{aligned}
NS(1)_{ref} &= \{1\} \cup \{2,3,4,5\} \\
NS(9) &= \{9\} \cup \{2,3,4,5\} \\
D(NS(1)_{ref}, NS(9)) &= |NS(1)_{ref} \triangle NS(9)| \\
&= |\{1,2,3,4,5\} \triangle \{9,2,3,4,5\}| \\
&= |\{1,9\}| \\
&= 2
\end{aligned}
$$

Notre méthode de calcul de distance entre la signature de l'environnement actuel et celle de référence ne souligne pas distinctement le remplacement de l'hôte. En effet, avec une distance de 2, nous avons le même résultat que lors d'un fonctionnement normal du mécanisme d'ancrage abordé en Section 5.4.4.

Ce scénario est particulièrement intéressant puisqu'il met en évidence le besoin de pondérer les informations qui sont utilisées dans la construction des signatures

d'environnements ainsi que dans le calcul de distance. On pourrait supposer que la correspondance de l'hôte d'une implémentation *white-box* avec celui renseigné lors de sa compilation serait pertinente. Afin de mieux illustrer notre étude préliminaire, nous avons fait le choix de présenter un exemple avec un minimum d'attributs. En pratique, lorsque de multiples attributs seront considérés, il est fort probable que ces derniers n'aient pas la même importance. Si l'on prend l'analogie d'une tour d'ordinateur, il peut arriver que les disques de stockages soient changés, de même que la carte graphique. En revanche, un changement de carte mère est beaucoup plus rare. Ainsi, les informations relatives à la carte mère du poste pourraient être plus importantes dans le calcul de la signature de l'appareil que celles concernant ses périphériques de stockage.

Afin de nous prémunir contre les attaques décrites dans les scénarios précédents, il serait nécessaire de définir un seuil de distance de 3. De la sorte, les scénarios d'extraction d'implémentation *white-box* et de vol de l'objet seraient identifiés par le mécanisme d'ancrage. L'identification d'un seuil de distance optimal est une tâche particulièrement délicate puisqu'elle requiert de bénéficier de nombreuses données afin de minimiser le nombre de faux positifs (attaques qui ne sont pas identifiées), mais également de minimiser les faux négatifs (cas légitimes qui sont détectés comme frauduleux).

5.5 Conclusion et améliorations

Cette contribution présente les principes de la cryptographie *white-box*, les attaques auxquelles cette catégorie de cryptographie est confrontée et présente une méthode pour se prémunir contre une partie de ces attaques. La méthode proposée consiste à ancrer une implémentation de cryptographie *white-box* sur son environnement local. Contrairement à un ancrage sur un appareil spécifique, cette pratique offre l'avantage de permettre l'identification des attaques consistant à voler l'objet hôte.

Notre travail repose sur l'hypothèse qu'il existe un compilateur d'implémentation *white-box* qui soit en mesure d'effectuer des comparaisons de distances dans le domaine *white-box*. Sur cette supposition, nous proposons une méthodologie d'ancrage qui repose sur l'utilisation d'une signature de l'environnement local d'un objet. La fonction de calcul des signatures doit respecter certaines propriétés que nous décrivons dans ce chapitre. Nous abordons brièvement le mécanisme de compilation des implémentations ancrées et leur déploiement au sein du réseau.

Finalement, plusieurs scénarios d'attaques sont illustrés à l'aide d'un exemple simplifié afin de démontrer l'intérêt que peut avoir un tel mécanisme d'ancrage. Avec un choix minutieux des paramètres, ce mécanisme permet aussi bien de prévenir les attaques par extraction de code que celles où l'objet est subtilisé.

Ces travaux peuvent être enrichis sur de multiples aspects. Le premier concerne l'identification des méthodes de signatures et de leurs comparaisons. Dans notre exemple, nous avons utilisé des mécanismes très simplifiés pour illustrer notre méthodologie. Il serait intéressant d'identifier quelles méthodes de signatures peuvent

être utilisées et surtout avec quelles sont les différentes fonctions envisageables pour calculer leurs distances. Un second axe d'approfondissement concerne les informations qui sont utilisées et leur pondération en fonction de leur importance. Nous avons mis en évidence ce besoin lors de notre dernier scénario d'attaque en Section 5.4.4.3 dans lequel, avec nos méthodes utilisées, le remplacement de l'hôte d'une implémentation *white-box* avait une faible influence sur la signature locale de l'implémentation.

En conclusion, plusieurs axes doivent encore être explorés pour valider ce principe, cependant l'ancrage local des implémentations *white-box* semble leur offrir de nouvelles perspectives de protection.

Troisième partie
Framework et Conclusion

L'objet de cette partie est de rassembler les travaux effectués au cours de cette thèse, de décrire en quoi ces travaux contribuent à l'élaboration d'un framework de sécurisation adaptative des objets de l'IoT et de les positionner vis-à-vis de ce framework.

Chapitre 6

Framework et application

Le rôle de ce chapitre est d'introduire notre framework de sécurisation adaptative des objets de l'IoT, de décrire ses éléments constitutifs et de positionner nos contributions vis-à-vis de ce framework. Nous présentons également son utilisation au travers d'un bref exemple consistant à déployer un correctif de sécurité sur un parc d'objets IoT.

Sommaire

6.1 Présentation générale du framework de sécurisation adaptative 145
6.2 Éléments constitutifs du framework 146
6.3 Positionnement des contributions de la thèse vis-à-vis du framework . 149
6.4 Exemple d'utilisation du framework 150
6.5 Gestion de la mobilité et de la dynamicité des objets IoT 151

Au cours de cette thèse, nous avons contribué sur divers aspects de la sécurisation des objets de l'IoT tout en gardant un focus sur les services de sécurité. Le rassemblement de ces travaux contribue à la définition d'un composant structurel que nous appelons notre framework de sécurisation adaptative des objets de l'IoT.

6.1 Présentation générale du framework de sécurisation adaptative

Ce framework s'articule autour de notre formalisation des problèmes de placement de services de sécurité et de leurs modélisations à l'aide d'une structure sémantique réalisée au Chapitre 4. Le rôle de ce framework est de contribuer aux principes de *context awareness* en fournissant un système capable de récolter de l'information sur un réseau d'objets IoT et d'adapter la sécurité de ce réseau en fonction de son état.

Pour ce faire, notre framework doit permettre d'effectuer les opérations illustrées en Figure 6.1.

FIGURE 6.1 – Étapes du framework de sécurisation adaptative des objets de l'IoT.

Le framework doit être en mesure de récupérer des informations sur le réseau d'objets IoT, de traiter ces informations, de déterminer quels sont les besoins en sécurité du réseau et quels sont les services qui y répondent. Une fois les services identifiés, le framework doit alors déterminer quels sont les emplacements les plus appropriés pour le déploiement de ces services de sécurité et effectuer leur déploiement. Constituer notre framework de sécurisation adaptative des objets de l'IoT revient alors à modéliser une structure qui reprend ces différentes étapes.

Notre framework se construit atour de la structure sémantique que nous avons présentée au Chapitre 4. Cette structure constitue ce que nous avons appelé le bloc sémantique qui correspond à notre base de connaissances. Autour de ce bloc gravitent les diverses entités qui modélisent les étapes décrites en Figure 6.1. Nous avons ainsi une entité en charge de capturer les informations sur le réseau d'objets IoT pour les saisir auprès de notre base de connaissances. Par la suite, une seconde entité s'appuie sur la base de connaissances et des ressources extérieures pour déterminer quels sont les besoins de sécurité du réseau et quels services pourraient répondre à ces besoins. Une troisième entité détermine alors, pour un service de sécurité donné et à l'aide du bloc sémantique, l'emplacement idéal pour le déploiement de ce service de sécurité. Cette entité s'accompagne d'une quatrième entité chargée du déploiement du service de sécurité requis à l'emplacement indiqué. La Figure 6.2 illustre ces différentes entités.

Dans leur méthodologie de conception d'ontologies, Noy et al. mentionnent une règle fondamentale relative à la modélisation [201]. Les auteurs stipulent qu'il n'existe pas d'unique manière de modéliser un domaine. Cela signifie qu'il n'existe également pas d'unique manière de constituer notre framework de sécurisation et que d'autres façons de procéder sont également pertinentes.

La section suivante aborde nos différents choix de représentation au travers de la description des principaux éléments constitutifs du framework.

6.2 Éléments constitutifs du framework

Notre framework de sécurisation adaptative des objets de l'IoT est composé de six entités principales que nous appelons blocs. Ces blocs remplissent chacun des rôles spécifiques afin de modéliser le plus fidèlement les étapes que nous avons décrites en Figure 6.1. Les blocs ainsi décrits peuvent alors, lors de la réalisation de ce framework, être considérés comme indépendants ou être combinés.

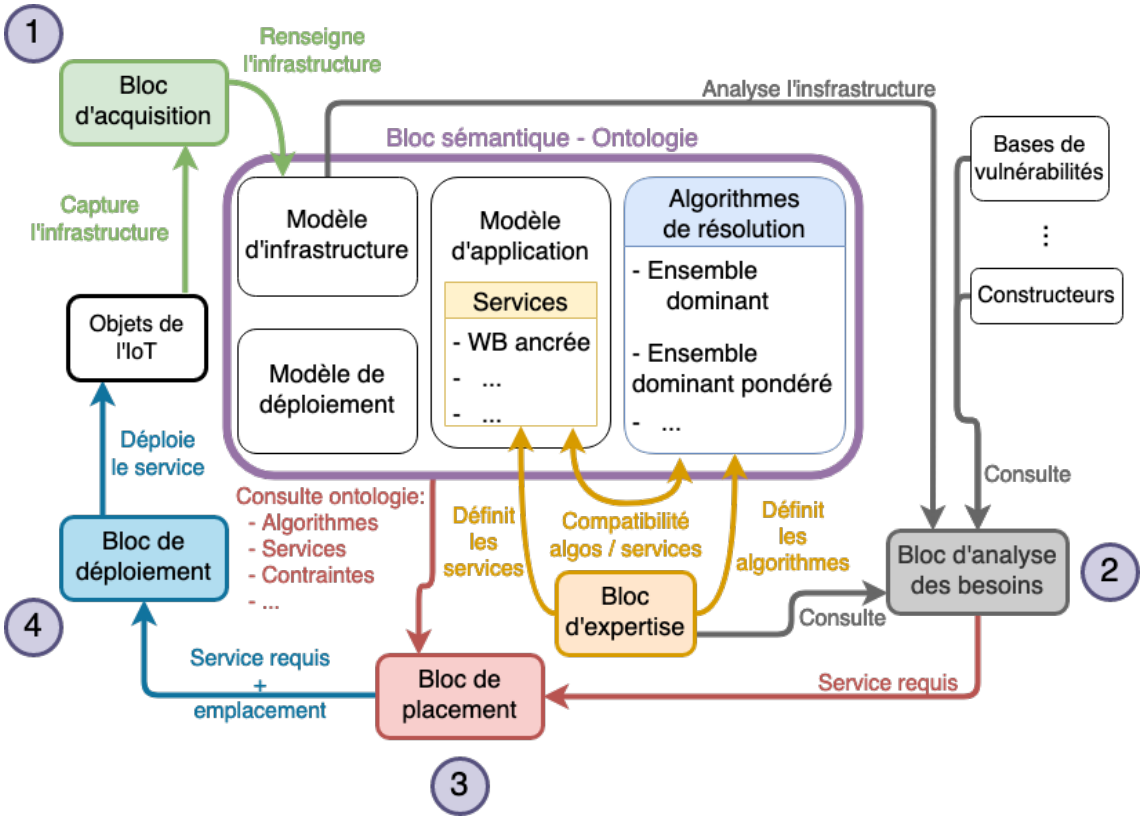


FIGURE 6.2 – Illustration du framework de sécurisation adaptatif des objets de l'IoT.

6.2.1 Bloc sémantique

Le bloc sémantique constitue le cœur de notre framework. Il est composé d'une ontologie permettant la définition des problèmes de placement de services de sécurité dans l'IoT (décrite au Chapitre 4). Cette dernière agit comme une base de connaissances et recense l'ensemble des informations relatives à l'infrastructure à sécuriser, les différents services de sécurité à disposition, leurs contraintes de placement ainsi que les algorithmes de placement de ces services. Grâce à l'utilisation de cette structure, le bloc sémantique est en mesure d'inférer de nouvelles connaissances en s'appuyant sur l'utilisation d'un raisonneur et de règles de raisonnement.

L'ensemble des informations sont alors mises à disposition des autres entités du framework.

6.2.2 Bloc d'acquisition

Le rôle du bloc d'acquisition est de saisir le maximum d'informations possibles sur le réseau d'objets IoT pour ensuite les renseigner dans le bloc sémantique et plus particulièrement auprès de la base de connaissances. Les méthodes d'acquisition peuvent être manuelles ou automatiques. Dans le cas d'une acquisition manuelle, les informations du réseau sont récupérées par un individu, typiquement un administrateur du système. L'acquisition automatique repose quant à elle sur l'utilisation d'outils de

reconnaissance tels que *Nmap* [175] pour la découverte de nouveaux périphériques et de capteurs pour capturer les diverses informations du réseau d'objets et de leur environnement.

6.2.3 Bloc d'expertise

Le bloc d'expertise agit auprès de plusieurs entités. Son rôle est de regrouper l'ensemble des *domain experts*, des individus possédant des connaissances et compétences dans un domaine spécifique.

La mise en place de notre framework requiert diverses compétences spécifiques. Dans un premier temps, un premier groupe d'experts en sécurité est nécessaire afin qu'ils puissent définir les différents services de sécurité auprès de notre base de connaissances. En complément du service, ces experts doivent également indiquer quelles sont les contraintes du service (performances minimum, présence de matériel spécifique ...) ainsi qu'une fonction objective permettant de déterminer quelle sera la solution la plus intéressante si plusieurs sont disponibles. Par la suite, un second groupe est requis pour la définition des algorithmes de résolutions des problèmes de placement de services ainsi que pour assigner à chaque service les algorithmes de résolution compatibles. Finalement, un troisième groupe d'experts intervient dans l'analyse des besoins de sécurité.

6.2.4 Bloc d'analyse des besoins

Le bloc d'analyse des besoins observe l'état du réseau d'objets IoT, détermine les besoins en sécurité et indique le service à déployer pour répondre à ces besoins. Pour ce faire, le bloc met en relation les informations de l'infrastructure IoT avec des sources de données externes telles que les bases de vulnérabilités. En complément de cette mise en relation, un groupe d'experts peut être consulté afin de déterminer si la sécurité de l'infrastructure est compromise et d'identifier quels services de sécurité pourraient résoudre le problème.

L'analyse des besoins ne requiert pas systématiquement la consultation d'experts physiques, les travaux de Gyrard et al. [129] permettent par exemple à des développeurs et concepteurs d'applications, n'étant pas spécialistes en sécurité, de réaliser des applications sécurisées et de prendre en compte les principaux risques de sécurité liés aux différentes technologies.

Une approche similaire pourrait alors être employée auprès du bloc d'analyse des besoins afin de limiter le besoin d'experts et de reposer principalement sur des règles de raisonnement. Nous pourrions ainsi nous rapprocher d'une automatisation de cette procédure.

6.2.5 Bloc de placement

Le rôle du bloc de placement est de résoudre les problèmes de placement de services dans l'IoT. Le bloc reçoit une requête pour placer un service de sécurité spécifique dans l'infrastructure IoT. À l'aide du bloc sémantique et des informations relatives

au service présent dans la base de connaissances (dont notamment, ses contraintes, sa fonction objective et l'infrastructure du réseau), le bloc de placement est en mesure de construire un problème de placement de services. Une fois le problème de placement défini, le bloc de placement consulte alors la structure sémantique afin d'identifier quels sont les algorithmes de résolutions compatibles avec le service à positionner. En cas de multiples solutions au problème de placement, une solution optimale est sélectionnée en se référant à la fonction objective du service. Le bloc de placement réalise ainsi une association entre les services à déployer et leurs emplacements optimaux.

6.2.6 Bloc de déploiement

Le bloc de déploiement consiste à récupérer les services de sécurité qui doivent être déployés ainsi que les emplacements auxquels ils doivent l'être afin d'effectuer le déploiement. Le rôle de cette entité est alors d'adapter les méthodes de déploiement en fonction des différents objets sur lesquels les services de sécurité doivent être déployés. Cela peut aller de la simple mise à jour comme au déploiement d'un nouveau firmware personnalisé avec le service de sécurité requis. Le framework n'étant pas limité à l'aspect logiciel, une méthode de déploiement possible peut consister à requérir l'intervention d'un technicien pour effectuer, par exemple, l'installation d'un nouveau composant matériel.

6.3 Positionnement des contributions de la thèse vis-à-vis du framework

Le framework que nous avons décrit dans les sections précédentes reprend l'ensemble des contributions que nous avons effectuées dans cette thèse.

Lors de notre étude sur le placement des services de sécurité dans l'IoT réalisée au Chapitre 3, nous avons découvert que les travaux sur le positionnement de service dans l'IoT pouvaient offrir des perspectives intéressantes pour effectuer une sécurisation adaptative des réseaux de l'IoT. En effet, selon les services qui sont déployés et l'emplacement de ces déploiements, il est possible d'adapter et même d'accroître la sécurité de l'ensemble d'un réseau d'objet IoT. Nous avons ainsi fait le choix d'orienter notre framework de sécurisation autour de cette notion de placement de services de sécurité afin de bénéficier de ces aspects adaptatifs.

Nous avons par la suite, au Chapitre 4, formalisé les problèmes de placement de services de sécurité dans l'IoT et construit une structure sémantique qui reprend cette formalisation. Nous avons ainsi repris ces travaux afin de constituer notre bloc sémantique qui constitue le cœur de notre framework.

Au cours des travaux décrits dans le Chapitre 3 de ce manuscrit, nous avons défini deux algorithmes de résolutions des problèmes de placement de sécurité : un premier qui identifie un ensemble dominant en privilégiant les sommets d'un graphe dont le poids est important et une variante de cet algorithme dans lequel les poids des sommets dépendent de la capacité d'un objet IoT à recevoir un service de sécurité.

Ces algorithmes peuvent ainsi être décrits dans notre structure sémantique et ainsi être des éléments constitutifs de notre framework tel qu'illustré en Figure 6.3.

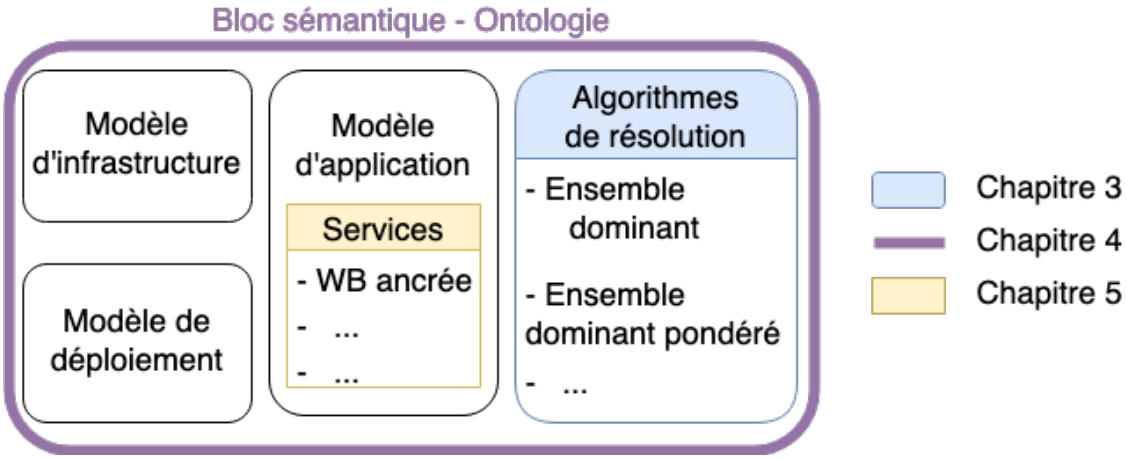


FIGURE 6.3 – Positionnement des contributions sur le bloc sémantique de notre framework de sécurisation adaptatif des objets de l'IoT.

Nous pouvons d'ailleurs observer dans la Figure 6.3 que notre troisième contribution relative à l'ancrage d'implémentations cryptographique *white-box* sur l'environnement local d'un objet s'intègre également dans notre framework de sécurisation des objets de l'IoT. Les travaux de cette contribution, décrite au Chapitre 5, peuvent ainsi être intégrés au framework sous la forme de services de sécurité spécifique et peuvent ainsi être déployés suivant le modèle de déploiement détaillé en Section 5.3.2 grâce au bloc de déploiement.

Le bloc sémantique est l'élément dans lequel nos contributions sont les plus présentes, cependant ces dernières ont également impacté les autres blocs de notre framework. Alors que le Chapitre 4 se focalise sur le bloc sémantique, les Chapitres 3 et 5 jouent le rôle d'experts par la définition de services de sécurité et d'algorithmes de résolution des problèmes de placement de services. Les travaux présentés au Chapitre 3 en particulier contribuent au framework en proposant une méthodologie de résolution des problèmes de placements de services de sécurité qui peut être modélisée au sein du bloc de placement.

6.4 Exemple d'utilisation du framework

Illustrons l'utilisation de notre framework de sécurisation sur l'exemple du déploiement d'un correctif de sécurité.

La première étape consiste à acquérir, par l'intermédiaire de notre bloc d'acquisition, le maximum d'informations possibles sur le parc d'objets IoT à sécuriser. Dans notre exemple, ces informations peuvent contenir les versions de firmware ou d'applications des appareils. Ces données sont ensuite saisies auprès du bloc sémantique.

Les données ainsi présentes dans notre bloc sémantique sont ensuite analysées par le bloc d'analyse des besoins qui va les mettre en relation avec d'éventuelles autres ressources externes comme une base de vulnérabilité. Les versions de firmware des objets de l'infrastructure sont ainsi comparées avec les bases de vulnérabilité afin de déterminer si certaines versions sont présentes dans la base et d'identifier les appareils vulnérables. Un expert en sécurité est ensuite consulté. Dans notre exemple, ce rôle peut être endossé par la base de vulnérabilité qui peut fournir un correctif de sécurité. Ce dernier est alors défini dans le bloc sémantique s'il n'y est pas déjà présent, cela comprend la définition du service, de ses contraintes et diverses conditions relatives à son placement dont les algorithmes de résolution.

Le besoin de déployer un service de correctif est alors exprimé auprès du bloc de placement qui consulte le bloc sémantique pour déterminer ses conditions de placement afin d'optimiser la solution de placement. Pour notre exemple, les conditions de placement sont simplifiées, elles correspondent à tous les appareils dont la version de firmware est vulnérable. Les solutions de placement sont alors identifiées et transmises auprès du bloc de déploiement sous la forme de paires (services / emplacement).

Le bloc de déploiement effectue alors le déploiement aux emplacements spécifiés. Dans notre cas, ce bloc va déployer les correctifs de sécurité sur les objets qui possèdent une version de firmware vulnérable.

6.5 Gestion de la mobilité et de la dynamicité des objets IoT

Afin de traiter plus fidèlement les problèmes de placement dans l'IoT, une gestion de la dynamicité et de la mobilité est nécessaire. En effet, comme illustré à la Figure 3.2, les problèmes de placement de services induisent la gestion de ces deux aspects. Dans l'état actuel de notre framework de sécurisation adaptative, la gestion de la mobilité et de la dynamicité ne sont pour le moment pas prises en compte. Nous proposons quelques pistes à considérer dans une future évolution de ce framework afin de potentiellement couvrir le cas où nos objets seraient mobiles.

6.5.1 Mobilité des objets IoT

Les objets de l'IoT subissent régulièrement des déplacements dans leur environnement. C'est notamment le cas dans le contexte des objets de la ville intelligente où nous pouvons par exemple mentionner le cas d'un véhicule qui se déplace dans une ville. Il est alors ainsi nécessaire de prendre en compte de déplacement et l'impact sur les interactions entre les objets qu'il représente. L'exemple du véhicule permet d'illustrer les deux cas typiques de mobilité, le premier cas pourrait correspondre à une mobilité récurrente alors que le second correspondrait plus à une mobilité ponctuelle. Dans le cas de la mobilité récurrente, les objets se déplacement à de multiples reprises aux mêmes localisations. L'exemple le plus représentatif pour un véhicule correspond au trajet domicile-travail. Dans ce premier cas, notre framework

pourrait considérer que l'objet possède deux localisations et adapter le positionnement des services de sécurité en considérant ces deux emplacements dans le réseau. Le second cas de mobilité est plus difficile à couvrir par notre framework puisqu'il correspond à une mobilité ponctuelle et qu'il est difficile de prévoir quels seront les futurs emplacements de notre objet. Une approche similaire au concept de *slices* 5G pourrait être envisagé afin de traiter cette condition. Pour cela il faudrait déterminer divers *slices* dans notre bloc sémantique et affecter nos objets mobiles à ceux-ci. Les services de sécurité seraient alors déployés au sein de ces *slices* et pourraient ainsi protéger, mais surtout suivre le déplacement de ces objets.

6.5.2 Dynamicité

Notre framework permet de gérer la dynamicité d'un réseau IoT à l'aide de son bloc sémantique et au bloc d'acquisition. En effet, le bloc sémantique recense l'ensemble des modifications subies dans notre réseau alors que le bloc d'acquisition permet de renseigner ces changements au moment opportun. Dans nos travaux nous nous sommes focalisés sur un réseau statique, ainsi le code de notre framework ne prend pas encore en compte la dynamicité du réseau. Il est important de mentionner que même si notre framework couvre cet aspect dynamique, il est également nécessaire que nos algorithmes de positionnement soient compatibles. En effet, dans l'algorithme 2 que nous avons présenté en Section 3.4.2, nous observons que ce dernier n'est pas compatible avec un graphe dynamique et nécessiterait alors d'être relancé pour fournir un résultat actualisé. Une alternative pour cet algorithme pourrait consister à, lors de l'ajout d'un objet dans le réseau, identifier si cet objet est en relation directe avec un nœud de sécurité et si cela n'est pas le cas, déployer un service de sécurité sur ce nouveau nœud ou un de ses voisins. Cette procédure est facilement faisable en raison de l'aspect local des ensembles dominants. Afin de préserver des valeurs de centralité (qui est une mesure globale) correctes, il serait alors pertinent de calculer les nouvelles valeurs de centralités après un certain nombre de modifications dans le graphe.

Chapitre 7

Conclusion & perspectives

Sommaire

7.1 Travaux et contributions 153
7.2 Perspectives de travail 155

Au cours de cette thèse, nous avons étudié la sécurité des objets de l'IoT et plus particulièrement le placement de services de sécurité dans les réseaux IoT. Pour ce faire, nous avons commencé notre étude en décrivant le contexte de la thèse, ses problématiques et nos motivations. Par la suite, nous avons analysé la sécurité générale de l'IoT et leurs architectures. Le travail de la thèse est axé autour de trois contributions majeures : la première vise à résoudre les problèmes de placement de services de sécurité dans l'IoT au travers de l'utilisation d'outils de la théorie des graphes. Ce problème est ensuite formalisé et généralisé pour être défini dans une structure sémantique. Finalement, notre dernière contribution se focalise sur les services de sécurité en proposant un mécanisme de signature locale pour l'ancrage d'implémentation de cryptographie *white-box*. Nos contributions participent à la réalisation d'un composant structurel (décrit au Chapitre 6) qui propose une sécurisation adaptative des objets de l'IoT.

7.1 Travaux et contributions

La première contribution de cette thèse repose sur l'élaboration d'une stratégie de déploiement de services de sécurité dans l'IoT qui minimise leurs coûts de déploiements. Pour ce faire, nous avons modélisé notre réseau IoT sous la forme d'un graphe et traduit nos contraintes de placement sous la forme d'un problème de graphe et plus particulièrement par l'identification d'un ensemble dominant favorisant la sélection de sommets avec un poids élevé. Notre pondération repose sur l'utilisation de mesures de centralités ce qui nous a permis de démontrer que l'utilisation de la betweenness centrality offre de nombreux avantages. En plus de permettre une sélection des

sommets en travers du flux d'information, cette mesure réduit considérablement la taille des ensembles dominants tout en permettant d'influer sur les valeurs de centralité en pondérant les sommets. Nous démontrons par la suite qu'une classification des sommets en fonction de leurs capacités à accueillir des services de sécurité permet d'augmenter significativement la qualité des ensembles dominants identifiés sans pour autant augmenter trop leurs tailles. Cette démonstration souligne l'apport de la pondération des sommets en fonction de leurs capacités d'accueil de services de sécurité pour la minimisation des frais de déploiement de ces services. À l'issue de ces travaux, la solution proposée offre des résultats intéressants avec un impact limité sur le transit de l'information à condition qu'une attention particulière soit apportée dans le choix des valeurs de pondération des sommets.

La seconde contribution formalise et généralise les problèmes de placement de services de sécurité dans l'IoT sous la forme d'ontologies. Pour ce faire, nous formalisons les problèmes de placement de service de sécurité dans l'IoT à l'aide de logique Monadique Existentielle du Second Ordre (MESO). Par la suite nous définissons une structure sémantique, à savoir une ontologie, pour décrire ces problèmes et permettre leur interprétation par des machines. Nous proposons ensuite deux méthodologies qui reposent sur cette structure : une première pour l'identification de solution de placement sur ces problèmes et une seconde pour la comparaison de ces solutions. Pour finir nous regroupons les apports des travaux sur cette contribution au sein d'un outil qui repose sur notre structure ontologique démontrant ainsi l'apport d'une telle structure pour la généralisation des problèmes de placement, la comparaison de leurs solutions mais surtout, de manière plus spécifique, pour la sécurisation des objets de l'IoT. L'intérêt de proposer une structure facilement interprétée par des machines offre de nombreux avantages sur les capacités d'adaptation des systèmes d'informations pouvant ainsi automatiser leurs actions.

Notre troisième contribution repose sur l'ancrage d'implémentation cryptographique *white-box* sur leurs environnements pour se prémunir contre les attaques par extraction de code et subtilisation des périphériques d'accueil de ces services de sécurité. Là où des mécanismes d'ancrage sur le périphérique seul seraient toujours vulnérables au vol de l'objet, notre proposition se prémunit contre ce problème. Nos travaux supposent l'existence d'un compilateur d'implémentations cryptographique *white-box* qui soit en mesure d'effectuer des comparaisons de distance dans le domaine *white-box*. En nous basant sur une telle structure, notre méthodologie d'ancrage de l'implémentation s'effectue par l'intermédiaire d'une signature locale de l'environnement d'accueil des objets. Nous listons l'ensemble des propriétés qu'une telle signature doit respecter pour permettre le fonctionnement de cette méthodologie ainsi que les processus de compilation et distribution de ces implémentations dans le réseau IoT. Nous terminons cette contribution en décrivant plusieurs scénarios d'attaques dans lesquels nous illustrons l'apport d'un tel mécanisme d'ancrage local.

Finalement, au travers de cette thèse, nous discutons des problématiques de sécurité des objets de l'IoT et en particulier des problèmes de placement de services de sécurité et contribuons à la réalisation d'un framework de sécurisation des objets de l'IoT par l'intermédiaire de nos différents travaux. L'ensemble de nos travaux sont ainsi réunis et complétés pour constituer un framework de sécurisation adaptative

des objets de l'IoT.

7.2 Perspectives de travail

Au cours de cette thèse, nous avons dû faire certains choix qui ont orienté nos travaux. Nous avons notamment abordé les problèmes de placement de service de sécurité dans l'IoT de manière statique et centralisée. Une exploration des aspects dynamiques des réseaux IoT et collaboratifs des entités pour leur sécurisation pourrait constituer des perspectives de travail intéressantes pour la poursuite des travaux de cette thèse. L'étude de la confiance dans les réseaux IoT pourrait également constituer un axe de recherche donnant suite aux travaux de cette thèse en complément des notions de sécurité étudiées.

7.2.1 Dynamicité et collaboration entre les entités de sécurisation des réseaux IoT

Tout au long de cette thèse et en particulier dans les contributions des Chapitres 3 et 4, nous avons fait abstraction de l'aspect dynamique des réseaux IoT dans nos modélisations. Cet aspect peut toutefois être pris en considération dans nos modèles notamment grâce au modèle formel que nous avons défini en Section 4.2.1 basé sur la théorie des modèles finis. Plus généralement, il serait intéressant d'étudier les problèmes de placement de service de sécurité dans un contexte différent du nôtre. Comme décrit en Section 3.2.7 et illustré en Figure 3.3, nous n'avons pas pris en considération d'autres aspects tels que la mobilité des objets ou encore la coordination entre les entités de sécurisation. Ce dernier aspect pourrait être abordé en procédant de manière similaire aux travaux de Nezami et al. [199], c'est-à-dire en dupliquant notre framework sur l'ensemble des nœuds fog de notre infrastructure. Il est cependant évident que procéder de la sorte n'est pas une solution optimale en particulier si les informations contenues dans ce framework doivent être maintenues à jour sur l'ensemble de ces nœuds. L'étude de la coordination entre ces entités de sécurisation pourrait alors constituer une suite à cette thèse.

7.2.2 La confiance dans les réseaux IoT

Au travers de la thèse, nous avons principalement abordé les notions de sécurité des objets de l'IoT. Nous avons basé notre approche de la sécurité par l'intermédiaire d'un gestionnaire de service de sécurité qui déploie ses services en fonction de besoins d'un réseau IoT. Lors de nos travaux, nous avons mentionné la possibilité que ces services de sécurité soient déclarés par des entités tierces. Le domaine de l'IoT regroupe un nombre conséquent d'acteurs de tous les domaines, que ce soit des opérateurs de télécommunication, des constructeurs d'objets, de composants ou bien même des développeurs de composants logiciels. La grande diversité de ces acteurs amène à s'interroger sur leur capacité à collaborer au sein de structures similaires à celle présentée dans notre framework. La notion de confiance constitue

une pierre angulaire de cette collaboration. Il est ainsi pertinent de s'interroger sur les mécanismes qui peuvent être mis en œuvre pour effectuer de la gestion de services de sécurité en se basant sur cette notion de confiance. Cet axe de recherche constitue également une piste de travail intéressante pour poursuivre les travaux effectués au cours de cette thèse.

Publications de l'auteur

Journaux

1. **Tanguy Godquin**, Morgan Barbier, Chrystel Gaber, Jean-Luc Grimault, and Jean-Marie Le Bars. Applied graph theory to security : A qualitative placement of security solutions within IoT networks. In Journal of Information Security and Applications.

Conférences internationales

1. **Tanguy Godquin**, Morgan Barbier, Chrystel Gaber, Jean-Luc Grimault, and Jean-Marie Le Bars. Placement optimization of IoT security solutions for edge computing based on graph theory. In 38th IEEE International Performance Computing and Communications Conference (IPCCC 2019).

Conférences nationales

1. **Tanguy Godquin**, Morgan Barbier, Chrystel Gaber, Jean-Luc Grimault, and Jean-Marie Le Bars. IoT : Vers un contrôle des fonctionnalités au vu des menaces liées. In Rendez-Vous de La Recherche et de l'Enseignement de La Sécurité Des Systèmes d'Information.

Brevets

FR3093882 **Titre** Procédé de configuration d'un objet communicant dans un réseau de communication, terminal utilisateur, procédé de connexion d'un objet communicant au réseau, équipement d'accès et programmes d'ordinateur correspondants.

Auteurs Godquin Tanguy, Jean-Luc Grimault

Numéro d'enregistrement FR1902427

Numéro de notice FR3093882

Bibliographie

[1] Chicago smart lighting program.

[2] Classification des tags RFID.

[3] FIPS 197, Advanced Encryption Standard (AES). page 51.

[4] Home page — LoRa Alliance™.

[5] Internet of Things market worldwide 2020 — Statistic.

[6] IoT : Number of connected devices worldwide 2012-2025.

[7] IoT Testing Guides - OWASP.

[8] Linked Open Vocabularies (LOV).

[9] Market Pulse Report, Internet Of Things (IoT).

[10] Object — Definition of object in English by Oxford Dictionaries.

[11] OOPS ! - OntOlogy Pitfall Scanner !

[12] OWASP Internet of Things Project - OWASP.

[13] PURL Administration.

[14] RDF Triple-Checker.

[15] Russian State-Sponsored Cyber Actors Targeting Network Infrastructure Devices — US-CERT.

[16] Semantic Sensor Network Ontology.

[17] Semantic Web Best Practices For Dummies.

[18] Sigfox - The Global Communications Service Provider for the Internet of Things (IoT).

[19] Sigfox Buy.

[20] Smart Agriculture Market by Agriculture Type - 2022 — MarketsandMarkets.

[21] Smart Mining Market by Hardware Component (Sensors, RFID tags, Intelligent Systems), System & Solution (Logistics Software, Data & Operation Management Software, Safety & Security Systems, Connectivity Solutions, Analytics Solutions, Remote Management Solutions, Asset Management Solutions), Service (Support & Maintenance, System Integration, Consulting Services) - Global Opportunity Analysis and Industry Forecast, 2014 - 2022.

[22] Smart Streetlights Program — Sustainability — City of San Diego Official Website.

[23] SmartCardAPI.

[24] SmartSantander.

[25] Social Internet of Things.

[26] What is Host Card Emulation (HCE) ? — Thales eSecurity.

[27] ZigBee & Z-Wave — All the Internet of Things - Episode One — Adafruit Learning System.

[28] Zigbee : Securing the Wireless IoT.

[29] IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks Specific Requirements Part 15.4 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). pages 0_1–670, 2003.

[30] Bluetooth specification version 4.0, 2010.

[31] IEEE Standard for Local and metropolitan area networks–Part 15.4 : Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1 : MAC sublayer. pages 1–225, 2012.

[32] Bluetooth specification version 4.2, 2014.

[33] The trade-offs of HCE on mobile payment security, 2015.

[34] Bluetooth specification version 5.0, 2016.

[35] IEEE Standard for Ethernet. pages 1–4017, 2016.

[36] IoE vs. IoT vs. M2M : What's the Difference and Does It Matter ?, 2016.

[37] Sigfox technical overview, 2017.

[38] 802.11ac : The Fifth Generation of Wi-Fi Technical White Paper. page 20, 2018.

[39] Connecting all the things in the Internet of Things, 2018.

[40] eSIM Whitepaper : The what and how of Remote SIM Provisioning, 2018.

[41] A Guide to the Internet of Things Infographic, 2018.

[42] Zigbee 3.0, 2018.

[43] Rachit Agarwal, David Gomez Fernandez, Tarek Elsaleh, Amelie Gyrard, Jorge Lanza, Luis Sanchez, Nikolaos Georgantas, and Valerie Issarny. Unified IoT ontology to enable interoperability and federation of testbeds. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 70–75. IEEE, 2016.

[44] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things : A survey on enabling technologies, protocols, and applications. 17(4) :2347–2376, 2015.

[45] HUSSEIN AHMAD AL-OFEISHAT. Near Field Communication (NFC). page 8, 2012.

[46] Thamer A. Alghamdi, Aboubaker Lasebae, and Mahdi Aiash. Security analysis of the constrained application protocol in the Internet of Things. In *Second International Conference on Future Generation Communication Technologies (FGCT 2013)*, pages 163–168. IEEE, 2013.

[47] Vincent Alimi and Marc Pasquet. Post-Distribution Provisioning and Personalization of a Payment Application on a UICC-Based Secure Element. pages 701–705. IEEE, 2009.

[48] Ruhul Amin, SK Hafizul Islam, G. P. Biswas, Muhammad Khurram Khan, Lu Leng, and Neeraj Kumar. Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks. 101 :42–62, 2016.

[49] Ioannis Andrea, Chrysostomos Chrysostomou, and George Hadjichristofi. Internet of Things : Security vulnerabilities and challenges. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 180–187. IEEE, 2015.

[50] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *USENIX Security Symposium*, 2017.

[51] Kevin Ashton et al. That 'internet of things' thing. 22(7) :97–114, 2009.

[52] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things : A survey. 54(15) :2787–2805, 2010.

[53] G. Avoine and P. Oechslin. A scalable and provably secure hash-based RFID protocol. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 110–114, 2005.

[54] Sachin Babar, Parikshit Mahalle, Antonietta Stango, Neeli Prasad, and Ramjee Prasad. Proposed security model and threat taxonomy for the Internet of Things (IoT). pages 420–429, 2010.

[55] Garvita Bajaj, Rachit Agarwal, Pushpendra Singh, Nikolaos Georgantas, and Valerie Issarny. A study of existing Ontologies in the IoT-domain, 2017.

[56] Jayant Baliga, Robert Ayre, Kerry Hinton, and Rodney Tucker. Energy consumption in wired and wireless access networks. 49(6) :70–77, 2011.

[57] S. Bandyopadhyay and A. Bhattacharyya. Lightweight Internet protocols for web enablement of sensors using constrained gateway devices. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 334–340. IEEE, 2013.

[58] Partha Sarathi Banerjee and Biswajit Maiti. Optimality criterion for the insertion of multi-interface nodes to improve connectivity in heterogeneous IoT framework. In *2017 Devices for Integrated Circuit (DevIC)*, pages 556–560. IEEE, 2017.

[59] Mario Ballano Barcena and Candid Wueest. Insecurity in the Internet of Things. 2015.

[60] Nathan Beckmann and Miodrag Potkonjak. Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions. In Stefan Katzenbeisser and Ahmad-Reza Sadeghi, editors, *Information Hiding*, volume 5806, pages 206–220. Springer Berlin Heidelberg, 2009.

[61] Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi, and Kerry Taylor. IoT-Lite : A lightweight semantic model for the internet of things and its use with dynamic semantics. 21(3) :475–487, 2017.

[62] Tim Berners-Lee. Linked Data - Design Issues, 2006.

[63] Marc Beunardeau, Aisling Connolly, Remi Geraud, and David Naccache. White-box cryptography : Security in an insecure environment. 14(5) :88–92, 2016.

[64] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Annual International Cryptology Conference*, pages 513–525. Springer, 1997.

[65] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In *International Workshop on Selected Areas in Cryptography*, pages 227–240. Springer, 2004.

[66] Mardiana binti Mohamad Noor and Wan Haslina Hassan. Current research on Internet of Things (IoT) security : A survey. 148 :283–294, 2019.

[67] Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich. Cryptographic schemes based on the ASASA structure : Black-box, white-box, and public-key. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 63–84. Springer, 2014.

[68] Andrey Bogdanov and Takanori Isobe. White-Box Cryptography Revisited : Space-Hard Ciphers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pages 1058–1069. ACM Press, 2015.

[69] Phillip Bonacich. Power and centrality : A family of measures. 92(5) :1170–1182, 1987.

[70] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog Computing : A Platform for Internet of Things and Analytics. In Nik Bessis and Ciprian Dobre, editors, *Big Data and Internet of Things : A Roadmap for Smart Environments*, volume 546, pages 169–186. Springer International Publishing, 2014.

[71] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing - MCC '12*, page 13. ACM Press, 2012.

[72] Maurizio A. Bonuccelli, Francesca Lonetti, and Francesca Martelli. Instant collision resolution for tag identification in RFID networks. 5(8) :1220–1232, 2007.

[73] Carsten Bormann. Guidance for light-weight implementations of the internet protocol suite. 2012.

[74] Stuart A Boyer. *SCADA : Supervisory Control and Data Acquisition*. International Society of Automation, 2009.

[75] Frank Brockners, Shwetha Bhandari, Sashank Dara, Carlos Pignataro, Hannes Gedler, Steve Youell, John Leddy, David Mozes, and Tal Mizrahi. Proof of Transit. 2018.

[76] Antonio Brogi, Stefano Forti, Carlos Guerrero, and Isaac Lera. How to Place Your Apps in the Fog – State of the Art and Open Challenges. 50(5) :719–740, 2020.

[77] Siri Bromander, Audun Jøsang, and Martin Eian. Semantic Cyberthreat Modelling. page 5.

[78] Zhipeng Cai, Zaobo He, Xin Guan, and Yingshu Li. Collective Data-Sanitization for Preventing Sensitive Information Inference Attacks in Social Networks. 15(4) :577–590, 2018.

[79] Benjamin Caudill. *Amazon Key Security : CloudCam Subject to Disruption Attacks*. 2017.

[80] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi. Long-Range Communications in Unlicensed Bands : The Rising Stars in the IoT and Smart City Scenarios. 23, 2016.

[81] Jihoon Cho, Kyu Young Choi, Orr Dunkelman, Nathan Keller, Dukjae Moon, and Aviya Vaidberg. *Hybrid WBC : Secure and Efficient White-Box Encryption Schemes*. 2016.

[82] Stanley Chow, Phil Eisen, Harold Johnson, and Paul C Van Oorschot. A white-box DES implementation for DRM applications. In *ACM Workshop on Digital Rights Management*, pages 1–15. Springer, 2002.

[83] Stanley Chow, Philip Eisen, Harold Johnson, and Paul C Van Oorschot. White-box cryptography and an AES implementation. In *Selected Areas in Cryptography*, volume 2595, pages 250–270. Springer, 2002.

[84] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. 51(4) :661–703, 2009.

[85] Stephen Cobb. RoT : Ransomware of Things. 2017.

[86] Vedat Coskun, Busra Ozdenizci, and Kerem Ok. A Survey on Near Field Communication (NFC) Technology. 71(3) :2259–2294, 2013.

[87] Michael J Covington and Rush Carskadden. Threat Implications of the Internet of Things. page 12.

[88] Jean Creusefond. Caractériser et détecter les communautés dans les réseaux sociaux, 2017.

[89] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries : A survey. 10(4) :2233–2243, 2014.

[90] M. Dehmer and F. Emmert-Streib. *Quantitative Graph Theory : Mathematical Foundations and Applications*. Discrete Mathematics and Its Applications. CRC Press, 2014.

[91] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. White-Box Security Notions for Symmetric Encryption Schemes. In Tanja Lange, Kristin Lauter, and Petr Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013*, volume 8282, pages 247–264. Springer Berlin Heidelberg, 2014.

[92] Jyoti Deogirikar and Amarsinh Vidhate. Security attacks in IoT : A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 32–37. IEEE, 2017.

[93] Parwinder Kaur Dhillon and Sheetal Kalra. A lightweight biometrics based remote user authentication scheme for IoT services. 34 :255–270, 2017.

[94] Whitfield Diffie and Martin Hellman. New directions in cryptography. 22(6) :644–654, 1976.

[95] Danny Dolev and Andrew Yao. On the security of public key protocols. 29(2) :198–208, 1983.

[96] Bruno Donassolo, Ilhem Fajjari, Arnaud Legrand, and Panayotis Mertikopoulos. Fog Based Framework for IoT Service Provisioning. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2019.

[97] Roberto Doriguzzi-Corin, Sandra Scott-Hayward, Domenico Siracusa, Marco Savi, and Elio Salvadori. Dynamic and Application-Aware Provisioning of Chained Virtual Security Network Functions. 17(1) :294–307, 2020.

[98] Bruno Dorsemaine, Jean-Philippe Gaulier, Jean-Philippe Wary, Nizar Kheir, and Pascal Urien. Internet of Things : A definition & taxonomy. In *Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference On*, pages 72–77. IEEE, 2015.

[99] M J Dworkin. Recommendation for block cipher modes of operation : : The CMAC mode for authentication, 2016.

[100] Morris Dworkin. Recommendation for block cipher modes of operation : The CCM mode for authentication and confidentiality. page 27.

[101] Dave Evans. How More Relevant and Valuable Connections Will Change the World. page 9.

[102] Dave Evans. The internet of things : How the next evolution of the internet is changing everything. 1(2011) :1–11, 2011.

[103] Sébastien Faux. La sécurité à l'ère des objets connectés : Comment s'y prendre ? In *Workshop "Sécurité Des Objets Connectés"*, 2017.

[104] Mohamed Amine Ferrag, Leandros A. Maglaras, Helge Janicke, Jianmin Jiang, and Lei Shu. Authentication Protocols for Internet of Things : A Comprehensive Survey. 2017 :1–41, 2017.

[105] Loïc Ferreira. IoT & sécurité : Retour vers le futur ?, 2017.

[106] Roy T Fielding and Richard N Taylor. *Architectural Styles and the Design of Network-Based Software Architectures*, volume 7. University of California, Irvine Doctoral dissertation, 2000.

[107] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 2259, pages 1–24. Springer Berlin Heidelberg, 2001.

[108] B Fouladi and S Ghanoun. Security Evaluation of the Z-Wave Wireless Protocol. page 6, 2013.

[109] Pierre-Alain Fouque, Pierre Karpman, Paul Kirchner, and Brice Minaud. Efficient and Provable White-Box Primitives. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, volume 10031, pages 159–188. Springer Berlin Heidelberg, 2016.

[110] Aurélien Francillon, Boris Danev, and Srdjan Capkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. 2011.

[111] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. 40(1) :35–41, 1977.

[112] Radek Fujdiak, Petr Blazek, Konstantin Mikhaylov, Lukas Malina, Petr Mlynek, Jiri Misurec, and Vojtech Blazek. On Track of Sigfox Confidentiality with End-to-End Encryption. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, pages 19 :1–19 :6. ACM, 2018.

[113] J. Furtak, Z. Zieliński, and J. Chudzikiewicz. Security techniques for the WSN link layer within military IoT. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 233–238, 2016.

[114] Eric Garcetti. IES SALC Conference 2016 - Mayor Garcetti Speech. 2016.

[115] M. R. Garey and David S. Johnson. Computers and Intractability : A Guide to the Theory of NP-Completeness. 1978.

[116] Sanjam Garg, Craig Gentry, Shai Halevi, and Amit Sahai. Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits. page 46, 2013.

[117] Gartner. *Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015.* 2014.

[118] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178. Association for Computing Machinery, 2009.

[119] J. Girao, D. Westhoff, and M. Schneider. CDA : Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 5, pages 3044–3049 Vol. 5, 2005.

[120] Tanguy Godquin, Morgan Barbier, Chrystel Gaber, Jean-Luc Grimault, and Jean-Marie Le Bars. IoT : Vers un contrôle des fonctionnalités au vu des menaces liées. In *Rendez-Vous de La Recherche et de l'Enseignement de La Sécurité Des Systèmes d'Information*, 2018.

[121] Tanguy Godquin, Morgan Barbier, Chrystel Gaber, Jean-Luc Grimault, and Jean-Marie Le Bars. Placement optimization of IoT security solutions for edge computing based on graph theory. In *38th IEEE International Performance Computing and Communications Conference (IPCCC 2019)*, 2019.

[122] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and Evaluation of Bluetooth Low Energy : An Emerging Low-Power Wireless Technology. 12(9) :11734–11753, 2012.

[123] Jérôme Gorin. Objets connectés : Typologie des risques sur la vie privée. In *Workshop "Sécurité Des Objets Connectés"*, 2017.

[124] Louis Goubin, Pascal Paillier, Matthieu Rivain, and Junwei Wang. *How to Reveal the Secrets of an Obscure White-Box Implementation.* 2018.

[125] David Grawrock. Using the tpm with iot. page 20.

[126] Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is an Ontology ? In *Handbook on Ontologies*, pages 1–17. 2009.

[127] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. iFogSim : A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments, 2016.

[128] A. Gyrard, C. Bonnet, K. Boudaoud, and M. Serrano. LOV4IoT : A Second Life for Ontology-Based Domain Knowledge to Build Semantic Web of Things Applications. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 254–261, 2016.

[129] Amelie Gyrard, Christian Bonnet, and Karima Boudaoud. The STAC (security toolbox : Attacks & countermeasures) ontology. In *Proceedings of the 22nd International Conference on World Wide Web - WWW '13 Companion*, pages 165–166. ACM Press, 2013.

[130] Amelie Gyrard, Soumya Kanti Datta, Christian Bonnet, and Karima Boudaoud. Cross-Domain Internet of Things Application Development : M3 Framework and Evaluation. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pages 9–16. IEEE, 2015.

[131] Amelie Gyrard, Martin Serrano, and Ghislain A. Atemezing. Semantic web methodologies, best practices and ontology engineering applied to Internet of Things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 412–417. IEEE, 2015.

[132] V. V. h Ram, H. Vishal, S. Dhanalakshmi, and P. M. Vidya. Regulation of water in agriculture field using Internet Of Things. In *2015 IEEE Technological Innovation in ICT for Agriculture and Rural Development (TIAR)*, pages 112–115, 2015.

[133] Shai Halevi. Graded Encoding, Variations on a Scheme. page 20.

[134] Stephan Haller, Stamatis Karnouskos, and Christoph Schroth. The Internet of Things in an Enterprise Context. In John Domingue, Dieter Fensel, and Paolo Traverso, editors, *Future Internet – FIS 2008*, volume 5468 of *Lecture Notes in Computer Science*, pages 14–28. Springer Berlin Heidelberg, 2009.

[135] Joseph Y Halpern and Riccardo Pucella. MODELING ADVERSARIES IN A LOGIC FOR SECURITY PROTOCOL ANALYSIS. page 26.

[136] Eric Hazane. Sécurité numérique des objets connectés, l'heure des choix, 2018.

[137] Debiao He, Neeraj Kumar, and Jong-Hyouk Lee. Secure pseudonym-based near field communication protocol for the consumer internet of things. 61(1) :56–62, 2015.

[138] Grant Hernandez, Orlando Arias, Daniel Buentello, and Yier Jin. Smart Nest Thermostat : A Smart Spy in Your Home. page 8.

[139] Ben Herzberg, Dima Bekerman, and Igal Zeifman. *Breaking down Mirai : An IoT DDoS Botnet Analysis*. 2016.

[140] Elike Hodo, Xavier Bellekens, Andrew Hamilton, Pierre-Louis Dubouilh, Ephraim Iorkyase, Christos Tachtatzis, and Robert Atkinson. Threat analysis of IoT networks using artificial neural network intrusion detection system. In *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE, 2016.

[141] Matthew Horridge. A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3. page 108, 2011.

[142] Md Mahmud Hossain, Maziar Fotouhi, and Ragib Hasan. Towards an analysis of security issues, challenges, and open problems in the internet of things. In *Services (SERVICES), 2015 IEEE World Congress On*, pages 21–28. IEEE, 2015.

[143] HP. *HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack*. 2014.

[144] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. Cyber-Physical Systems Security – A Survey, 2017.

[145] Michaela Iorga, Larry Feldman, Robert Barton, Michael J Martin, Ned Goren, and Charif Mahmoudi. Fog computing conceptual model, 2018.

[146] Matthias Jacob, Dan Boneh, and Edward Felten. Attacking an obfuscated cipher by injecting faults. In *ACM Workshop on Digital Rights Management*, pages 16–31. Springer, 2002.

[147] F. Ben Jemaa, G. Pujolle, and M. Pariente. QoS-Aware VNF Placement Optimization in Edge-Central Carrier Cloud Architecture. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, 2016.

[148] Salim Jouili, Ines Mili, and Salvatore Tabbone. Attributed Graph Matching Using Local Descriptions. In Jacques Blanc-Talon, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, volume 5807 of *Lecture Notes in Computer Science*, pages 89–99. Springer Berlin Heidelberg, 2009.

[149] Marc Joye. On white-box cryptography. pages 7–12, 2008.

[150] Jung-Hyun Cho, Jikon Kim, Jae-Whan Kim, Kyungil Lee, Kwang-Duk Ahn, and Shiho Kim. An NFC transceiver with RF-powered RFID transponder mode. pages 172–175. IEEE, 2007.

[151] Mohamed Karroumi. Protecting white-box AES with dual ciphers. In *International Conference on Information Security and Cryptology*, pages 278–291. Springer, 2010.

[152] Sindhu Karthikeyan and Mikhail Nesterenko. RFID security without extensive cryptography. page 63. ACM Press, 2005.

[153] Masanobu Katagi and Shiho Moriai. Lightweight Cryptography for the Internet of Things. page 4.

[154] Olaide O. Kazeem, Olubiyi O. Akintade, and Lawrence O. Kehinde. Comparative Study of Communication Interfaces for Sensors and Actuators in the Cloud of Internet of Things. 6(1) :9–13, /26/2017.

[155] Todd Kennedy and Ray Hunt. A Review of WPAN Security : Attacks and Prevention. In *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, Mobility '08, pages 56 :1–56 :8. ACM, 2008.

[156] Neal Koblitz. Elliptic Curve Cryptosystems. page 7.

[157] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. 1(1) :5–27, 2011.

[158] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, volume 1109, pages 104–113. Springer Berlin Heidelberg, 1996.

[159] Brian Krebs. *Target Hackers Broke in via HVAC Company*. 2014.

[160] David Kushner. The real story of stuxnet. 3(50) :48–53, 2013.

[161] Markku Laine. RESTful Web Services for the Internet of Things. page 3.

[162] Jean-Baptiste Lamy. Owlready : Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. 80 :11–28, 2017.

[163] R. Langner. Stuxnet : Dissecting a Cyberwarfare Weapon. 9(3) :49–51, 2011.

[164] Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and B. Samadi. A survey on wireless security protocols (WEP, WPA and WPA2/802.11i). In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 48–52, 2009.

[165] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A Comparative Study of Wireless Protocols : Bluetooth, UWB, ZigBee, and Wi-Fi. pages 46–51. IEEE, 2007.

[166] Guillaume Lehembre. Wi-Fi security – WEP, WPA and WPA2. page 14.

[167] Marco Leo, Federica Battisti, Marco Carli, and Alessandro Neri. A federated architecture approach for Internet of Things security. In *2014 Euro Med Telco Conference (EMTC)*, pages 1–5, 2014.

[168] C. Lesjak, D. Hein, and J. Winter. Hardware-security technologies for industrial IoT : TrustZone and security controller. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pages 002589–002595, 2015.

[169] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things : Architecture, enabling technologies, security and privacy, and applications. 4(5) :1125–1142, 2017.

[170] Jie Ling, Ying Wang, and Wei-Feng Chen. An Improved Privacy Protection Security Protocol Based on NFC. pages 39–46, 2016.

[171] An Liu and Peng Ning. TinyECC : A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *2008 International Conference on Information Processing in Sensor Networks (Ipsn 2008)*, pages 245–256. IEEE, 2008.

[172] Z. Liu, Z. Cao, and D. S. Wong. White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Any Monotone Access Structures. 8(1) :76–88, 2013.

[173] K. Lofstrom, W.R. Daasch, and D. Taylor. IC identification circuit using device mismatch. In *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.00CH37056)*, pages 372–373. IEEE, 2000.

[174] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. Visualizing ontologies with VOWL. 7(4) :399–419, 2016.

[175] Gordon Lyon. Nmap, 2020.

[176] Rwan Mahmoud, Tasneem Yousuf, Fadi Aloul, and Imran Zualkernan. Internet of things (IoT) security : Current status, challenges and prospective measures. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 336–341. IEEE, 2015.

[177] Redowan Mahmud and Rajkumar Buyya. Modelling and simulation of fog and edge computing environments using iFogSim toolkit. pages 1–35, 2019.

[178] Koji Makino, Daichi Kishi, and Jun Bian. Building of GSMA3. 1-compliant eSIM Commercial System for IoT/M2M through Partnership between Operators. 2018.

[179] Pratyusa Manadhata and Jeannette M Wing. An Attack Surface Metric. page 23, 2005.

[180] Pratyusa K. Manadhata and Jeannette M. Wing. An Attack Surface Metric. 37(3) :371–386, 2011.

[181] Stephen M Matyas. Generating strong one-way functions with cryptographic algorithm. 27 :5658–5659, 1985.

[182] Julian McAuley and Jure Leskovec. Learning to Discover Social Circles in Ego Networks. page 9.

[183] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of LPWAN technologies for large-scale IoT deployment. 2018.

[184] Robert M Metcalfe and David R Boggs. Ethernet : Distributed Packet Switching for Local Computer Networks. 19(7) :10, 1976.

[185] Wil Michiels and Paul Gorissen. Mechanism for software tamper resistance : An application of white-box cryptography. In *Proceedings of the 2007 ACM Workshop on Digital Rights Management*, pages 82–89. ACM, 2007.

[186] Daniel Miessler. Securing the Internet of Things : Mapping Attack Surface Areas Using the OWASP IoT Top 10. page 54, 2015.

[187] Victor S. Miller. Use of Elliptic Curves in Cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, volume 218, pages 417–426. Springer Berlin Heidelberg, 1986.

[188] Roberto Minerva, Abyi Biru, and Domenico Rotondi. Towards a definition of the internet of things (IoT). 1(1) :1–86, 2015.

[189] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things : Vision, applications and research challenges. 10(7) :1497–1516, 2012.

[190] Javier Miranda, Niko Makitalo, Jose Garcia-Alonso, Javier Berrocal, Tommi Mikkonen, Carlos Canal, and Juan M. Murillo. From the Internet of Things to the Internet of People. 19(2) :40–47, 2015.

[191] Mahdi H. Miraz, Maaruf Ali, Peter S. Excell, and Rich Picking. A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT). pages 219–224. IEEE, 2015.

[192] Deepti Mittal, Damandeep Kaur, and Ashish Aggarwal. Secure Data Mining in Cloud Using Homomorphic Encryption. In *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–7, 2014.

[193] Amir-Hamed Mohsenian-Rad, Vincent W. S. Wong, Juri Jatskevich, Robert Schober, and Alberto Leon-Garcia. Autonomous Demand-Side Management Based on Game-Theoretic Energy Consumption Scheduling for the Future Smart Grid. 1(3) :320–331, 2010.

[194] Kenneth Moore. Towards optimal IoT workload placement in a fog computing environment. 2019.

[195] A. Mosenia and N. K. Jha. A Comprehensive Study of Security of Internet-of-Things. 5(4) :586–602, 2017.

[196] Nicky Mouha. The Design Space of Lightweight Cryptography. page 19.

[197] Mark A Musen. The protégé project : A look back and a look forward. 1(4) :4–12, 2015.

[198] Jihoon Myung, Wonjun Lee, and J. Srivastava. Adaptive binary splitting for efficient RFID tag anti-collision. 10(3) :144–146, 2006.

[199] Zeinab Nezami, Kamran Zamanifar, Damiano Arena, and Dimitris Kiritsis. Ontology-Based Resource Allocation for Internet of Things. In *Advances in Production Management Systems. Production Management for the Factory of the Future*, pages 323–330. Springer, Cham, 2019.

[200] Jianting Ning, Zhenfu Cao, Xiaolei Dong, Lifei Wei, and Xiaodong Lin. Large Universe Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability. In Mirosław Kutyłowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014*, volume 8713, pages 55–72. Springer International Publishing, 2014.

[201] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101 : A guide to creating your first ontology. 2001.

[202] Mehedi Hassan Onik, Nasr Al-Zaben, Hung Phan Hoo, and Chul-Soo Kim. A Novel Approach for Network Attack Classification Based on Sequential Questions. 2(2) :14, 2018.

[203] Kyosuke Osaka, Tsuyoshi Takagi, Kenichi Yamazaki, and Osamu Takahashi. An Efficient and Secure RFID Security Method with Ownership Transfer. In *RFID Security*, pages 147–176. Springer, Boston, MA, 2008.

[204] Colin O'Flynn. A Lightbulb Worm ? Details of the Philips Hue Smart Lighting Design. page 48.

[205] N. Petrovic and M. Tosic. SMADA-Fog : Semantic model driven approach to deployment and adaptivity in Fog Computing. page 102033, 2019.

[206] Charles P Pfleeger, Shari Lawrence Pfleeger, and Jonathan Margulies. *Security in Computing*. Prentice Hall Professional Technical Reference, 2002.

[207] Philippe Pittoli, Pierre David, and Thomas Noël. Améliorations de dtls : connexion rapide et augmentation de la charge utile pour les réseaux contraints. page 5.

[208] Pawani Porambage, Jude Okwuibe, Madhusanka Liyanage, Mika Ylianttila, and Tarik Taleb. Survey on Multi-Access Edge Computing for Internet of Things Realization. 20(4) :2961–2991, 2018.

[209] Mason A. Porter. Small-world network. 7(2) :1739, 2012.

[210] Jon Postel. User datagram protocol, 1980.

[211] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS ! (OntOlogy Pitfall Scanner !) : An On-line Tool for Ontology Evaluation, 2014.

[212] Krishna P. N. Puttaswamy, Ranjita Bhagwan, and Venkata N. Padmanabhan. Anonygator : Privacy and Integrity Preserving Data Aggregation. In Indranil Gupta and Cecilia Mascolo, editors, *Middleware 2010*, volume 6452 of *Lecture Notes in Computer Science*, pages 85–106. Springer Berlin Heidelberg, 2010.

[213] R. A. Rahman and B. Shah. Security analysis of IoT protocols : A focus in CoAP. In *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pages 1–7, 2016.

[214] Sandra Rasoamiaramanana. Conception de schémas de chiffrement boîte blanche pour la sécurité des applications mobiles., 2020.

[215] Sandra Rasoamiaramanana, Gilles Macario-Rat, and Marine Minier. White-Box Traitor-Tracing from Tardos Probabilistic Codes. In Emil Simion and Rémi Géraud-Stewart, editors, *Innovative Security Solutions for Information Technology and Communications*, Lecture Notes in Computer Science, pages 125–141. Springer International Publishing, 2020.

[216] Partha Pratim Ray. Internet of things for smart agriculture : Technologies, practices and future direction. 9(4) :395–420, 2017.

[217] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low Power Wide Area Networks : An Overview, 2016.

[218] Molka Rekik, Khouloud Boukadi, and Hanêne Ben-abdallah. Cloud Description Ontology for Service Discovery and Selection :. In *Proceedings of the 10th International Conference on Software Engineering and Applications*, pages 26–36. SCITEPRESS - Science and and Technology Publications, 2015.

[219] Matthieu Rivain. White-Box Cryptography. page 98.

[220] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. 21(2) :120–126, 1978.

[221] Marco Rocchetto and Nils Ole Tippenhauer. CPDY : Extending the Dolev-Yao Attacker with Physical-Layer Interactions, 2016.

[222] Rodrigo Roman, Pablo Najera, and Javier Lopez. Securing the Internet of Things. page 10, 2011.

[223] Eyal Ronen and Adi Shamir. Extended functionality attacks on iot devices : The case of smart lights. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium On*, pages 3–12. IEEE, 2016.

[224] Mehdi Roopaei, Paul Rad, and Kim-Kwang Raymond Choo. Cloud of things in smart agriculture : Intelligent irrigation monitoring by thermal imaging. 4(1) :10–15, 2017.

[225] Tomáš Rosa. Bypassing Passkey Authentication in Bluetooth Low Energy. page 3.

[226] Britta Ruhnau. Eigenvector-centrality—a node-centrality ? 22(4) :357–365, 2000.

[227] Gert Sabidussi. The centrality index of a graph. 31(4) :581–603, 1966.

[228] Syed Muhammad Sajjad and Muhammad Yousaf. Security analysis of IEEE 802.15.4 MAC in the context of Internet of Things (IoT). pages 9–14. IEEE, 2014.

[229] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre. An overview of service placement problem in Fog and Edge Computing, 2019.

[230] Eloi Sanfelix, Cristofaro Mune, and Job de Haas. Unboxing the White-Box. 2015.

[231] David E. Sanger. Obama Ordered Wave of Cyberattacks Against Iran. 2012.

[232] saraclay. Trusted Platform Module (TPM) - Windows IoT.

[233] Subhadeep Sarkar and Sudip Misra. Theoretical modelling of fog computing : A green computing paradigm to support IoT applications. 5(2) :23–29, 2016.

[234] Alex Schiffer. How a fish tank helped hack a casino, 2017.

[235] Bruce Schneier. *Lessons From the Dyn DDoS Attack*. 2016.

[236] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez, and Thierry Monteil. IoT-O, a Core-Domain IoT Ontology to Represent Connected Devices Networks. In Eva Blomqvist, Paolo Ciancarini, Francesco Poggi, and Fabio Vitali, editors, *Knowledge Engineering and Knowledge Management*, volume 10024 of *Lecture Notes in Computer Science*, pages 561–576. Springer International Publishing, 2016.

[237] Z Shelby. CoRE Link Format, draft-ietf-core-link-format-11.

[238] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP), 2014.

[239] Cogan M Shimizu. Rendering OWL in LaTeX for Improved Readability : Extensions to the OWLAPI. page 107, 2017.

[240] Sooyeon Shin and Taekyoung Kwon. Two-Factor Authenticated Key Agreement Supporting Unlinkability in 5G-Integrated Wireless Sensor Networks. 6 :11229–11241, 2018.

[241] John Sian. Securing the Internet of Things - where's the risk ?, 2014.

[242] Matti Siekkinen, Markus Hiienkari, Jukka K. Nurminen, and Johanna Nieminen. How low energy is bluetooth low energy ? Comparative measurements with ZigBee/802.15.4. pages 232–237. IEEE, 2012.

[243] Philipp Singer. Not every long tail is power law !, 2018.

[244] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. Optimized IoT service placement in the fog. 11(4) :427–443, 2017.

[245] Olena Skarlat, Matteo Nardelli, Stefan Schulte, and Schahram Dustdar. Towards QoS-Aware Fog Service Placement. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 89–96. IEEE, 2017.

[246] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez. A Survey of IoT-enabled Cyberattacks : Assessing Attack Paths to Critical Infrastructures and Services. pages 1–1, 2018.

[247] T. Stock and G. Seliger. Opportunities of Sustainable Manufacturing in Industry 4.0. 40 :536–541, 2016.

[248] Hui Suo, Jiafu Wan, Caifeng Zou, and Jianqi Liu. Security in the internet of things : A review. In *2012 International Conference on Computer Science and Electronics Engineering*, volume 3, pages 648–651. IEEE, 2012.

[249] York Sure, Steffen Staab, and Rudi Studer. On-To-Knowledge Methodology (OTKM). In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 117–132. Springer, 2004.

[250] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The NeOn Methodology for Ontology Engineering. In *Ontology Engineering in a Networked World*, 2012.

[251] Latanya Sweeney. K-ANONYMITY : A MODEL FOR PROTECTING PRIVACY. 10(05) :557–570, 2002.

[252] Zhiqing Tang, Xiaojie Zhou, Fuming Zhang, Weijia Jia, and Wei Zhao. Migration Modeling and Learning Algorithms for Containers in Fog Computing. 12(5) :712–725, 2019.

[253] Ming Tao, Jinglong Zuo, Zhusong Liu, Aniello Castiglione, and Francesco Palmieri. Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes. 78 :1040–1051, 2018.

[254] M. U.Farooq, Muhammad Waseem, Anjum Khairi, and Sadia Mazhar. A Critical Analysis on the Security Concerns of Internet of Things (IoT). 111(7) :1–6, 2015.

[255] P. Urien. Cloud of secure elements : An infrastructure for the trust of mobile NFC services. In *2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 213–218, 2014.

[256] P. Urien. Innovative DTLS/TLS security modules embedded in SIM cards for IoT trusted and secure services. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 276–277, 2016.

[257] Mike Uschold and Michael Gruninger. Ontologies : Principles, methods and applications. 11(2) :93–136, 1996.

[258] Tzi-cker Chiueh Chitra Venkatramani. Supporting Real-Time Tra c on Ethernet. page 21.

[259] Malisa Vucinic, Bernard Tourancheau, Thomas Watteyne, Franck Rousseau, Andrzej Duda, Roberto Guizzetti, and Laurent Damon. DTLS Performance in Duty-Cycled Networks, 2015.

[260] Mališa Vučinić, Bernard Tourancheau, Franck Rousseau, Andrzej Duda, Laurent Damon, and Roberto Guizzetti. OSCAR : Object security architecture for the Internet of Things. 32 :3–16, 2015.

[261] Jessie Walker et al. Unsafe at any key size ; an analysis of the WEP encapsulation. 802(00) :362, 2000.

[262] Zhao Wang, Zhigang Xu, Wei Xin, and Zhong Chen. Implementation and Analysis of a Practical NFC Relay Attack Example. pages 143–146. IEEE, 2012.

[263] J. Wei, Q. Cheng, R. V. Penty, I. H. White, and D. G. Cunningham. 400 Gigabit Ethernet using advanced modulation formats : Performance, complexity, and power dissipation. 53(2) :182–189, 2015.

[264] Martin Woolley. Bluetooth 5 : Go Faster. Go Further, 2018.

[265] Brecht Wyseur. White-box cryptography, 2011.

[266] Brecht Wyseur. White-box cryptography : Hiding keys in software. 2012.

[267] Ye Xia, Xavier Etchevers, Loïc Letondeur, Thierry Coupaye, and Frédéric Desprez. Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed IoT applications in the fog. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing - SAC '18*, pages 751–760. ACM Press, 2018.

[268] Junfeng Xu, Tao Zhang, Dong Lin, Ye Mao, Xiaonan Liu, Shiwu Chen, Shuai Shao, Bin Tian, and Shengwei Yi. Pairing and Authentication Security Technologies in Low-Power Bluetooth. pages 1081–1085. IEEE, 2013.

[269] Lei Xu, Chunxiao Jiang, Jian Wang, Jian Yuan, and Yong Ren. Information Security in Big Data : Privacy and Data Mining. 2 :1149–1176, 2014.

[270] Teng Xu, James B. Wendt, and Miodrag Potkonjak. Security of IoT systems : Design challenges and opportunities. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 417–423. IEEE, 2014.

[271] Xinyu Yang, Xuebin Ren, Jie Lin, and Wei Yu. On Binary Decomposition Based Privacy-Preserving Aggregation Schemes in Real-Time Monitoring Systems. 27(10) :2967–2983, 2016.

[272] Yuchen Yang, Longfei Wu, Guisheng Yin, Lijie Li, and Hongbin Zhao. A Survey on Security and Privacy Issues in Internet-of-Things. 4(5) :1250–1258, 2017.

[273] Huaqing Zhang, Yong Xiao, Shengrong Bu, Dusit Niyato, Richard Yu, and Zhu Han. Computing Resource Allocation in Three-Tier IoT Fog Networks : A Joint Optimization Approach Combining Stackelberg Game and Matching, 2017.

[274] Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong-Kuan Chen, and Shiuhpyng Shieh. IoT security : Ongoing challenges and research opportunities. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pages 230–234. IEEE, 2014.

[275] Kai Zhao and Lina Ge. A Survey on the Internet of Things Security. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 663–667, 2013.

[276] Xu Zheng, Zhipeng Cai, Jianzhong Li, and Hong Gao. Location-privacy-aware review publication mechanism for local business service systems. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[277] Joe Zimmerman. How to Obfuscate Programs Directly. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9057, pages 439–467. Springer Berlin Heidelberg, 2015.

[278] Hubert Zimmermann. OSI reference model–The ISO model of architecture for open systems interconnection. 28(4) :425–432, 1980.

[279] Alf Zugenmaier, Michael Kreutzer, and Günter Müller. The Freiburg Privacy Diamond : An Attacker Model for a Mobile Computing Environment. page 12.

Annexe A

Les standards et protocoles de communication dans l'IoT

L'Internet des Objets repose entièrement sur la capacité des objets à inter communiquer. Pour permettre ces interactions, une multitude de standards et de protocoles existent.

Les standards de communication des objets de l'Internet des Objets sont variés ; certains utilisent les fréquences radio, d'autres le Wi-fi ou encore les communications satellites. Cette grande diversité de standards de communication amène à s'interroger sur la capacité des objets à interagir entre eux et sur la sécurité de leurs communications. La figure A.1 présente différents protocoles et standards utilisés dans l'Internet des Objets ainsi que leur position par rapport au modèle OSI. Les sous-paragraphes suivants aborderont les principaux protocoles de l'IoT (représentés en bleu sur la figure A.1) accompagnés d'une brève description.

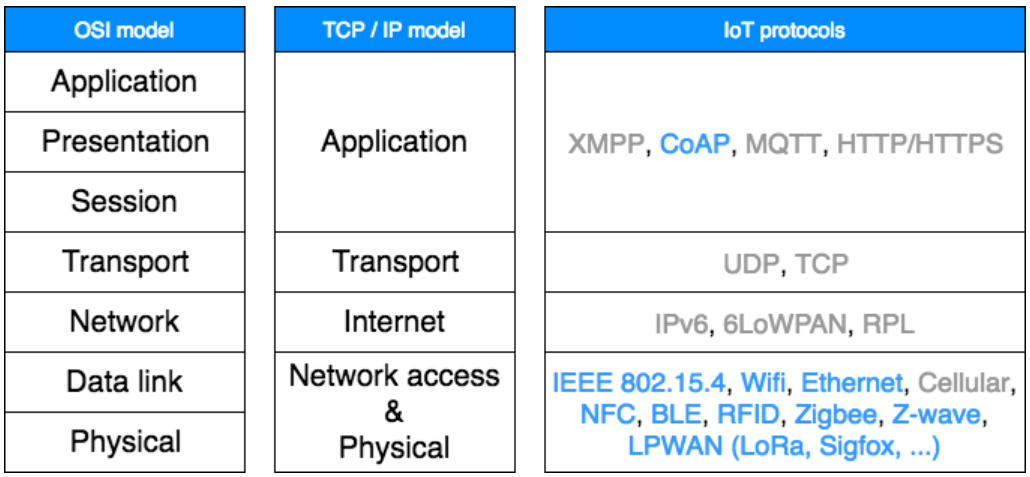


FIGURE A.1 – Position des principaux protocoles IoT sur les modèles OSI et TCP/IP (inspiré par [39])

A.1 Internet conventionnel

Les technologies de l'internet conventionnel, à savoir le Wifi et Ethernet, sont également présentes dans le domaine de l'Internet des Objets. Ces technologies sont rarement adoptées dans l'élaboration de réseaux de capteurs sans fil (*Wireless Sensor Networks*) en raison de leur forte consommation énergétique et du besoin de connectique physique dans le cas d'Ethernet. Malgré cela, ces technologies restent intéressantes lors de besoins de transferts d'informations volumineuses puisqu'elles permettent d'atteindre des débits théoriques de l'ordre de 3.47 Gbits/s [38] pour le Wifi et 400Gbit/s [263] pour Ethernet.

Wifi

Le Wi-Fi (ou Wifi) est une technologie de communication sans fil qui repose sur les standards IEEE 802.11 et l'ensemble des protocoles qu'ils régissent. Cette technologie de communication permet d'atteindre des débits très élevés jusqu'à 3.47 Gbits/s théoriques avec le standard IEEE 802.11ac [38] mais occasionne une consommation énergétique plus importante que d'autres protocoles de communication sans-fil [154].

Il existe plusieurs types de réseaux possibles. Le premier, dit "infrastructure", comporte un point d'accès central auquel tous les périphériques souhaitant communiquer se connectent. Il correspond au type de Wifi mis à disposition par défaut par les fournisseurs d'accès internet dans les foyers via leurs routeurs. Le second, appelé "bridge", permet de relier plusieurs points d'accès entre eux. Le troisième type de réseau Wifi, nommé "repeater" se contente d'augmenter l'intensité du signal Wifi et permet ainsi d'étendre la zone de couverture. La dernière catégorie de réseaux Wifi est dite "ad-hoc" et permet à deux périphériques de se connecter directement l'un à l'autre sans passer par un point d'accès.

La sécurisation des communications Wifi repose sur la robustesse des algorithmes cryptographiques utilisés. RC4 est le premier algorithme cryptographique utilisé dans la norme IEEE 802.11 de 1999 qui est compris dans le protocole WEP. Le protocole WEP a été vite abandonné suite aux attaques découvertes par Walker et al. [261] en 2000 et Fluhrer et al. [107] en 2001 qui ont montré que les paramètres de l'algorithme RC4 utilisé dans le protocole WEP altéraient sa robustesse.

En réponse à cela, l'IEEE a publié la norme IEEE 802.11i dont le protocole WPA2 est issu (WPA n'étant qu'une solution temporaire). Le protocole WPA2 repose sur l'utilisation de l'algorithme cryptographique AES reconnu comme robuste [166] et approuvé par le NIST [3].

Il existe deux possibilités d'utiliser le protocole WPA2 [164] : la première, en mode "entreprise" (WPA-Enterprise) se repose sur l'utilisation de mécanismes d'EAP (Extensible Authentication Protocol) tel que RADIUS et requiert donc un serveur d'authentification, la seconde, en mode "personnel" (WPA-Personal) utilise un secret pré partagé entre les deux entités qui souhaitent communiquer.

Dans les deux modes (WPA-Personal et WPA-Enterprise), une concertation sur un secret commun, appelé *Pairwise Master Key (PMK)* est effectuée entre les deux

entités qui souhaitent communiquer. Avec WPA-Personal, la PMK correspond au secret pré partagé (PSK).

Par la suite, un échange de clés, appelé *4 way handshake*, va être effectué entre le client (appelé *Supplicant*) et le point d'accès (appelé *Authenticator*) et ce pour les deux modes présentés précédemment. Lors de cet échange, on suppose que les deux entités possèdent la clé PMK ainsi que les adresses MAC du client et du point d'accès (puisqu'ils ont déjà communiqué ensemble). Le point d'accès génère alors un nombre aléatoire à usage unique appelé *A nonce* et le communique au client. Le client génère également un nombre aléatoire à usage unique appelé *S nonce*. À cette étape, le client est en mesure de calculer PTK (*Pairwise Transient Key*), clé cryptographique dérivée de le PMK, des adresses MAC du client et du point d'accès ainsi que du *A nonce* et du *S nonce*. Le client génère un code de message d'intégrité (*MIC, Message Integrity Code*) en utilisant cette PTK et le communique avec le *S nonce* au point d'accès. Le point d'accès est ainsi en mesure de calculer la PTK et de vérifier son intégrité à l'aide du MIC reçu. Les deux entités peuvent maintenant installer la PTK et communiquer en unicast à l'aide de cette nouvelle clé. Concernant les communications multicast, le point d'accès génère une GTK (*Group Temporal Key*) ainsi qu'un MIC et communique ces deux éléments au client. Ce dernier vérifie l'intégrité de la GTK à l'aide du MIC et l'installe si son intégrité est confirmée. Le client termine l'échange en indiquant au point d'accès qu'il a bien installé les deux clés. La figure A.2 représente cet échange de clés.

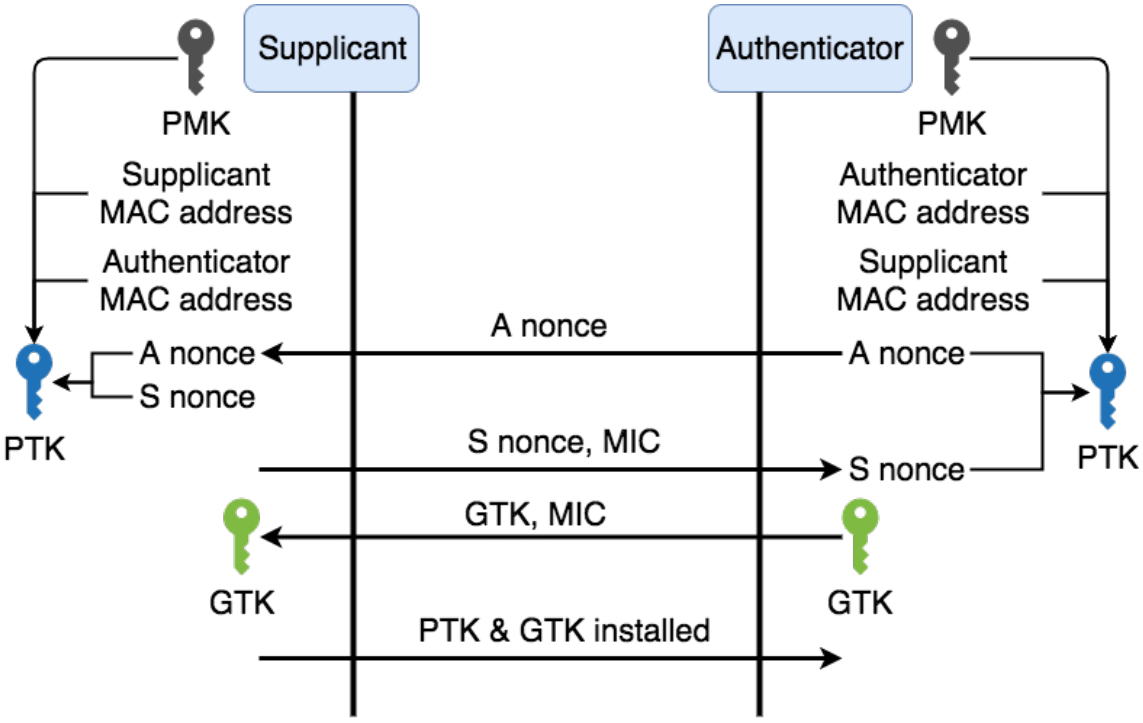


FIGURE A.2 – Représentation du *4 way handshake* utilisé dans WPA

L'utilisation du WPA-Personnal semble la plus adaptée dans une utilisation liée à l'Internet des Objets puisque ce mode ne sollicite pas de tierce partie dans

l'établissement de ses communications. Cependant, l'utilisation d'un serveur d'authentification présente dans WPA-Enterprise représente un atout important en termes de gestion de flottes d'objets connectés et de sécurité (pas de partage de secret). En revanche, cette dernière méthode est plus complexe et nécessite la présence d'un serveur d'authentification.

Ethernet

Le protocole Ethernet est l'un des plus vieux protocoles de communications régis par les normes IEEE 802.3 [35], dont les premières publications remontent à 1976 [184]. Ce protocole agit principalement sur les couches physique (*Physical*) et liaison (*Data link*) du modèle OSI et comprend de nombreux aspects matériels tels que l'utilisation des différents câblages (coaxial ou encore fibre optique). Malgré la forte présence des communications sans-fil entre les objets connectés, il ne faut pas oublier que les réseaux filaires sont toujours fortement utilisés pour des raisons de rapidité, de fiabilité, mais également, et c'est le plus important, dans le cadre des objets connectés, pour des raisons d'économie d'énergie [56]. Ses qualités en font un protocole particulièrement intéressant à utiliser comme fondation tel que l'ont fait remarquer Venkatramani et al. avec RETHER [258].

A.2 Sans-contact

Avec l'Internet des Objets, les technologies de communication sans-contact se développement de manière accrue. Ces technologies permettent à des appareils de communiquer sur des distances souvent très courtes pouvant aller de quelques dizaines de centimètres à plusieurs mètres. Leur atout majeur réside dans leurs structures appelées étiquette (*tag*) et leur capacité à utiliser l'énergie perçue par un lecteur afin de s'alimenter en énergie (étiquettes passives) sans nécessiter d'autres sources d'énergie. La structure des réseaux utilisant ces technologies est principalement de type point à point entre le lecteur et l'étiquette. Elles ne sont à priori pas adaptées à une utilisation sur une autre topologie de réseau.

RFID

Les communications dites RFID se basent sur des objets appelés *tags RFID* ou étiquettes RFID et utilisent les champs électromagnétiques pour communiquer. Il existe de nombreuses catégories d'étiquettes RFID [2]. Elles sont principalement composées d'une antenne et d'un circuit intégré. Il existe des étiquettes qui sont uniquement constituées d'un transpondeur qui font office d'identifiant radio. D'autres étiquettes, de type passif, se composent uniquement d'une diode capacitive indiquant sa présence lors d'une interrogation. Elles sont parfois appelées étiquettes *1 bit* et sont fortement utilisées dans les dispositifs antivol.

Les étiquettes dotées d'un circuit intégré, les plus présentes sur le marché, peuvent bénéficier d'un émetteur radio. Elles sont alors appelées étiquettes actives et bénéficient communément d'une source d'énergie (piles, batteries).

À l'inverse, les étiquettes qui ne bénéficient pas d'un tel émetteur sont appelées passives. Elles utilisent les ondes reçues afin d'alimenter leur circuit électronique et perturbent l'onde afin de transmettre de l'information. Il existe des étiquettes passives qui bénéficient toutefois d'une source d'énergie, elles sont alors dites *Battery Assisted Passive*. Le circuit intégré est alors alimenté par cette batterie permettant d'effectuer des opérations plus coûteuses en énergie ce qui permet le fonctionnement en l'absence de sollicitation extérieure.

Les étiquettes RFID communiquent selon deux protocoles distincts. Le premier, TTF (pour *Tag Talk First*), consiste à initialiser une communication par l'étiquette RFID. Il est important de mentionner que le signal envoyé par l'interrogateur (la borne RFID) n'est pas considéré comme l'initialisation d'une communication. Ainsi, des étiquettes dites passives peuvent communiquer en suivant le protocole TTF. Le principal avantage de ce protocole est sa rapidité à détecter la présence dans le champ d'action de l'interrogateur. Il est cependant sensible aux possibles collisions lors de la présence de multiples étiquettes dans la zone de rayonnement. Le second protocole, ITF (pour *Interrogator Talk First*), repose sur l'initialisation de la communication par l'interrogateur, à la suite de laquelle, l'étiquette répond. Ce protocole est intéressant puisqu'il permet de récupérer l'ensemble des réponses des étiquettes se trouvant dans la zone de rayonnement de l'interrogateur et de gérer les potentiels problèmes de collisions par la suite.

Il existe de nombreuses méthodes pour gérer les collisions lors de communications RFID comme les travaux de Myung et al. [198] ou encore ceux de Bonuccelli et al. [72].

La sécurité des communications RFID est souvent considérée comme faible de par l'impossibilité d'effectuer de la cryptographie asymétrique cependant, certains travaux (tels que ceux d'Avoine et Oechslin [53], d'Osaka et al. [203] ou encore de Karthikeyan et Nesterenko [152]) permettent d'étendre la sécurité de ces systèmes.

NFC

La technologie NFC (*Near Field Communication*) est une adaptation du RFID qui permet à deux objets de communiquer sur de très faibles distances de l'ordre de 5 à 10 cm [262, 155]. Chaque étiquette NFC bénéficie d'un identifiant unique pouvant être utilisé à son identification.

Les communications NFC ne bénéficient pas de mécanismes de diffusion, seuls deux acteurs sont présents [155] : l'initiateur (*Initiator*) et le destinataire (*Target*). L'initiateur débute et gère l'échange des données entre les deux entités alors que le destinataire se contente de lui répondre.

Ces deux rôles sont fréquemment associés aux trois types d'appareils utilisés dans les communications NFC [86] : le lecteur (*reader*) majoritairement associé à l'initiateur, l'étiquette (ou *tag*) qui correspond au destinataire, et le mobile. Le mobile est un appareil qui possède à la fois un lecteur et émule la présence d'une étiquette.

Les appareils NFC peuvent être actifs ou passifs [150, 45]. En mode actif, les deux entités génèrent leurs propres champs de radiofréquences. Cette configuration induit une consommation d'énergie important de la part des deux interlocuteurs. Afin de

remédier à cela, dans le mode passif, le destinataire ne produit plus son champ de radiofréquences, il se repose sur celui de l'initiateur.

Le protocole NFC définit 3 modes opératoires tels que décrits par [262, 45, 86, 47]. Un premier mode, en lecture/écriture permettant la modification et la lecture des données stockées sur une étiquette, qu'elle soit active ou passive. Le second appelé pair-à-pair consiste à échanger des données entre les deux entités telles que des photos. Le dernier consiste à simuler une étiquette de telle sorte qu'un lecteur ne la distingue pas d'une étiquette physique (carte d'accès par exemple). Cette technologie d'émulation se nomme HCE (*Host Card Emulation*) et sera abordée dans la sous-section 2.3.1.4.

Les problèmes de vie privée impliqués par l'absence d'anonymisation des l'identifiant unique [170, 137, 155], la restriction des communications à deux entités et la faible portée de ces dernières, font du NFC un protocole non adapté aux réseaux de capteurs connectés, mais intéressants dans les applications IoT grâce en particulier à sa technologie d'HCE.

A.3 LR-WAN

Le standard IEEE 802.15.4 est un standard de communication établi en 2003 [29] entrant dans la famille des des réseaux sans-fil LR-WAN (Low-Rate Wireless Personal Area Networks). Ce standard se focalise sur les couches basses du modèle OSI (*Physical* et *Media Access Control*) et procure ainsi une base pour un ensemble de protocoles de communication tels que Zigbee ou Thread.

Le protocole IEEE 802.15.4 offre la possibilité de faire communiquer des objets avec une faible consommation énergétique en utilisant notamment le principe de *Time Synchronized Channel Hopping* (TSCH) introduit dans la version IEEE 802.15.4e en 2012 [31]. Les nœuds du réseau vont s'accorder sur un ensemble de fenêtres de temps pendant lesquelles chaque nœud aura un état distinct et où les droits des correspondants en transmission et émission seront stipulés [228]. Lorsqu'un nœud reçoit un message à transmettre provenant de ses couches hautes (applications par exemple), ce dernier sera ajouté dans une file d'attente. Un nœud en état de transmission va consulter cette file, et si elle contient un paquet, le transmettre tout en attendant l'avis de réception du destinataire (*acknowledgement*). S'il ne reçoit pas confirmation de la réception, le nœud re communiquera le paquet lors de la prochaine fenêtre de transmission. Lors de la consultation de la file d'attente, si cette dernière est vide, le nœud retournera en état sommeil sans allumer son périphérique radio économisant ainsi de l'énergie. Lors de l'état sommeil, le périphérique n'utilise pas son périphérique radio, ce dernier est donc éteint. Dans un état de réception, le nœud active sa radio au moment où il est supposé recevoir un message et confirme la réception des paquets à son expéditeur via un *acknowledgement*.

A.3.1 Réseaux maillés

Les réseaux maillés sont particulièrement intéressants pour les réseaux de capteurs sans fil, ils permettent d'avoir une zone de couverture plus large que ce que pourrait fournir un nœud unique et introduit également la possibilité d'utiliser des trajets alternatifs lors de la chute de certains nœuds. Les principaux protocoles qui offrent la possibilité d'utiliser des réseaux maillés sont Z-Wave (propriétaire), ZigBee et BLE (depuis la version 5 de Bluetooth [34]).

ZigBee

Zigbee est un protocole radio basse consommation basé sur IEEE 802.15.4 qui permet de connecter plus de 65000 appareils sur un seul réseau avec une portée allant jusqu'à 60m pour une consommation énergétique environ 5 fois plus faible que le Wifi [165, 154]. Zigbee ne permet pas directement d'accéder à internet, dans un tel cas, une passerelle de conversion doit être mise en place.

Deux topologies de réseau sont offertes par Zigbee comme présenté en figure A.3 : la première, en maille (*Mesh network*), particulièrement intéressante dans le cadre de l'Internet des Objets et la seconde, dite en étoile ou centralisée, qui propose davantage de contrôle et de sécurité sur le réseau au détriment du confort d'utilisation.

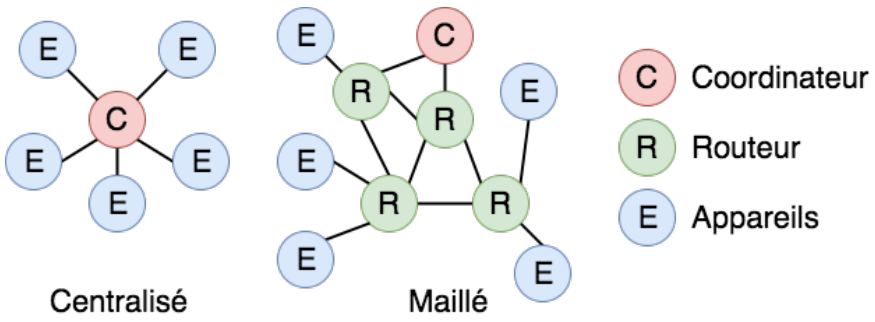


FIGURE A.3 – Les structures de réseau Zigbee

Dans une topologie de réseau décentralisée, chaque nœud peut avoir deux rôles : routeur et/ou appareil (aussi appelé *End device*). Lorsqu'un nouveau nœud souhaite rejoindre le réseau, un routeur déjà présent va lui communiquer une clé de façon sécurisée lui permettant de communiquer avec l'ensemble du réseau. La clé de chiffrement utilisée pour communiquer entre chaque entité du réseau (appelée *Network Key*) est la même pour tous les nœuds. Ainsi, chaque routeur du réseau est en mesure de la communiquer lors de l'arrivée d'un nouveau nœud. Cette procédure, bien que facile d'utilisation pose des problèmes en termes de sécurité puisque toute personne qui possède la clé peut déchiffrer les communications de tout le réseau. La topologie décentralisée introduit également des problèmes dépenses énergétiques puisque les routeurs peuvent être régulièrement sollicités pour rediriger du trafic [27, 28].

Zigbee propose également une topologie de réseau centralisé et plus sécurisé [28]. Dans cette dernière, un tiers de confiance est introduit appelé TC (*Trust Center*). Le TC est le seul à pouvoir fournir les clés cryptographiques et associe, pour chaque

appareil, une clé unique appelée *TC Link Key*. Lorsqu'un appareil souhaite rejoindre un réseau sécurisé, le TC demande à ce que soit fourni un *Install Code*. Ce dernier est un identifiant unique propre à chaque appareil qui est composé d'un nombre aléatoire de 128 bits protégé par 16 bits de CRC (*Cyclic Redundance Check*). Il est important que l'*Install Code* ne soit pas fourni via des communications sans-fil, mais par un autre médium tel que la saisie manuelle ou un QR-code.

L'appareil souhaitant rejoindre le réseau et le TC dérivent l'*Install Code* en utilisant une fonction de hachage basée sur la construction de Matyas-Meyer-Oseas [181] (MMO) pour obtenir une *TC Link Key* unique de 128 bits. Elle sera utilisée par le TC pour communiquer les clés de chiffrement à l'appareil. Un mécanisme de changement de clés de chiffrement est mis en place afin de prévenir tout attaquant qui arriverait à s'emparer d'une des clés. Le TC va régulièrement renouveler sa clé de chiffrement avec l'appareil en la chiffrant avec le *TC Link Key*. La figure A.4 résume l'introduction d'un nouvel appareil dans un réseau et le principe de renouvellement des clés de chiffrement.

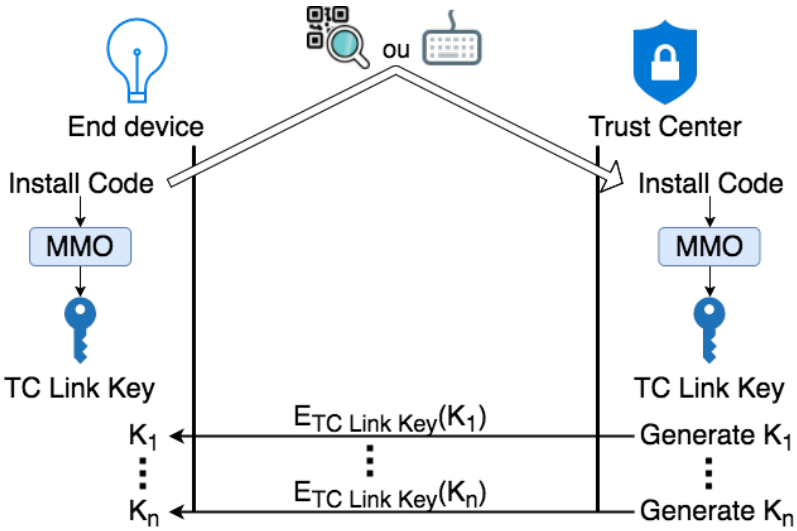


FIGURE A.4 – Procédure d'ajout d'un appareil dans un réseau Zigbee centralisé et renouvellement des clés

Le protocole Zigbee permet l'utilisation d'un *secure boot* pour vérifier l'intégrité d'une mise à jour du micrologiciel d'un objet [28]. Lors d'une mise à jour du système, l'image est chiffrée avec une clé et signée avec un autre, les deux clés étant uniques. À la réception d'une mise à jour du micrologiciel, le *secure boot* déchiffre l'image, vérifie son intégrité et effectue la mise à jour si son intégrité est confirmée. Dans le cas contraire, l'appareil va redémarrer avec sa dernière configuration valide. Il est important de préciser qu'il n'est pas fait mention de *secure element* pour protéger les clés liées au *secure boot*. Zigbee se protège toutefois en offrant la possibilité aux développeurs d'enregistrer l'image dans la puce interne tout en désactivant les options de debug pour restreindre l'accès en lecture.

Z-Wave

Z-Wave est un protocole maillé de la même sorte que Zigbee qui nécessite la présence d'une passerelle pour accéder à internet. Il permet à 232 appareils de communiquer au sein d'un même réseau. Lorsqu'ils ne sont pas alimentés par batteries (pas de restriction énergétique), les appareils du réseau peuvent prendre le rôle de routeur afin de proposer une couverture étendue. Le protocole Z-wave est propriétaire, fourni avec une puce spécifique à l'utilisation du protocole [27].

Il permet aux appareils du réseau d'avoir connaissance de la topologie du réseau auquel ils appartiennent par une découverte dynamique. Ils sont ainsi en mesure de choisir la route la plus adaptée à leurs communications et peuvent réagir en cas de modification de topologie.

Chaque périphérique Z-Wave possède 3 variables par défaut codées en dur dans leur micrologiciel [108] : une clé temporaire par défaut (*temporary default key*) composée de 16 octets de zéro et deux autres clés nommées $Passwd_c$ et $Passwd_m$ ayant toutes deux une taille de 16 octets. Lorsqu'un dispositif souhaite rejoindre le réseau, un nombre aléatoire est généré par le PRNG (Pseudo Random Number Generator) de la puce Z-Wave de l'appareil de commande pour constituer la clé réseau K_n. Ce nombre est ensuite chiffré en utilisant la clé temporaire par défaut inscrite dans le micrologiciel du composant et communiqué à l'appareil de commande.

Une fois K_n partagée, les deux parties peuvent dériver les clés *frame encryption key* (K_c) et *data origin authentication key* (K_m) respectivement obtenues en chiffrant $Passwd_c$ et $Passwd_m$ avec la clé K_n via l'algorithme AES. La figure A.5 représente le processus de génération de ces clés. Lors des futures communications K_c sera utilisée pour chiffrer les commandes alors que K_m sera utilisé pour le contrôle d'authentification et d'intégrité à l'aide d'un MAC (*Message Authentication Code*) [108].

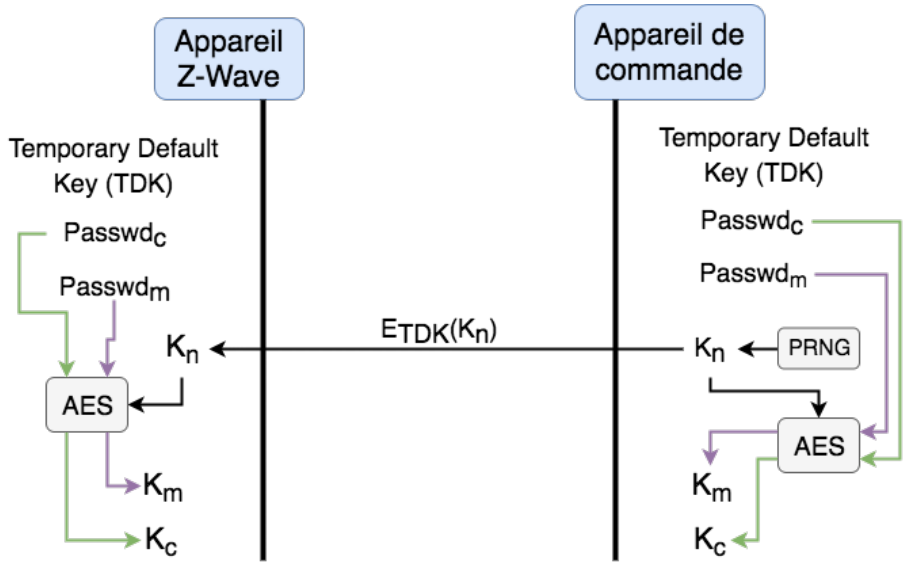


FIGURE A.5 – Échange de clés dans le protocole Z-Wave

BLE

La technologie Bluetooth possède deux formes [34] : *Basic Range* (BR) qui correspond à la technologie standard et *Low Energy* (LE ce qui donne BLE) qui est une version avec une consommation énergétique réduite introduite dans la version 4 du Bluetooth [30].

Le Bluetooth supporte de nombreux profils personnalisés à une multitude d'utilisations (transfert de fichiers, découverte et établissement de communications, périphériques sans-fil...) et permet à plusieurs objets de communiquer ensemble avec une portée fluctuante entre 10m et 100m selon des conditions normales [154, 165]. Le BLE possède une portée plus réduite de quelques dizaines de mètres (pouvant aller à jusqu'à 500m [264] depuis la version 5) avec un débit de données de l'ordre de 2Mb/s (2 Msym/s 1symbole=1bit) [34, 264] et permettant une économie d'énergie (rapport énergie consommée par bit transmit) plus importante que Zigbee [242].

Un réseau BLE est composé d'un nœud maitre (*Master*) et d'un maximum de 7 nœuds esclaves (*Slaves*) [154]. Une telle topologie est appelée *Piconet*. Il est possible de joindre plusieurs Piconets pour former un *Scatternet* comportant plusieurs nœuds maitres. Dans une telle topologie, il est possible qu'un nœud soit à la fois maitre, esclave, et esclave auprès de deux nœuds maitres [154, 165]. La version 5 de Bluetooth [34] propose également une topologie maillée (Mesh) similaire à celle de Zigbee.

L'élaboration d'un canal de communication sécurisé requiert l'appariement des appareils BLE. Pour ce faire, les versions antérieures à la version 4.2 [32] utilisaient une clé cryptographique temporaire (*Temporary Key*). Cependant, la robustesse de ce processus a été remise en cause [268, 225]. Bluetooth SIG a proposé, dans la version 4.2 de Bluetooth, l'utilisation d'un échange de clé Diffie Hellman [94].

Le chiffrement et l'authentification des données sont effectués par des algorithmes cryptographiques utilisant AES en version 128 bits et CCM (*Counter with Cipher Block Chaining-Message Authentication Code* [100]). Lors du chiffrement des données, un code de vérification d'intégrité (MIC, *Message Integrity Check*) est ajouté à la donnée à transmettre et la concaténation des deux éléments est ensuite chiffrée et transmise [122]. BLE permet également d'effectuer de l'authentification sur des données non chiffrées. Pour ce faire, une signature de 12 octets est calculée grâce à un algorithme AES 128 bits puis ajoutée à la suite du message à authentifier. Le destinataire assumera l'identité correcte s'il parvient à vérifier la signature [122].

Afin de se prévenir contre le pistage et de possibles atteintes à la vie privée, BLE propose un mécanisme de *privacy feature* consistant à utiliser une adresse "chiffrée" et à la changer régulièrement. L'appareil BLE génère une clé de chiffrement, la communique à un tiers de confiance et chiffre son adresse publique avec cette dernière. Le tiers de confiance est ainsi en mesure de faire de la résolution d'adresse [122, 34].

A.3.2 LPWAN

La technologie LPWAN (*Low Power Wide Area Networks*) regroupe un ensemble de protocoles permettant la communication sur de longues distances (entre 1 et 5 Km en ville et jusqu'à 40km hors agglomérations) [183, 80] avec une faible consommation énergétique (durée de vie des batteries de l'ordre de la dizaine d'années) [183].

Les deux principaux protocoles LPWAN sont LoRaWAN (protocole utilisant la technologie LoRa), promu par l'association LoRa Alliance [4] et Sigfox, protocole propriétaire régi par l'entreprise du même nom [18].

Ces deux protocoles de communication possèdent une topologie réseau similaire : ils ont tous deux une topologie en étoile. Chaque appareil va devoir passer par une antenne relai afin de pouvoir communiquer avec le reste du réseau. Ces protocoles utilisent également les bandes ISM sans licence sur des fréquences basses à 868MHz (en Europe).

Sigfox

Le protocole Sigfox limite le nombre de communications des objets du réseau. Chaque objet est limité à 140 messages émis maximum [183, 19] avec une charge utile de 12 octets par messages [183] et à 4 messages reçus par jour. Sigfox affirme être ainsi en mesure de supporter un million d'objets par passerelle [80]. Afin de prévenir les risques de pertes de messages, Sigfox transmet plusieurs fois le même message sur des fréquences différentes.

Les communications ne sont pas chiffrées par défaut. Au besoin, il est possible de dériver une clé de chiffrement depuis l'identifiant unique de l'appareil [183, 37]).

LoRaWAN

Le protocole LoRaWAN propose le chiffrement avec un algorithme AES 128 bits en version CCM [100] et CMAC [99] en utilisant une clé symétrique pré partagée [105].

LoRaWAN propose une charge utile dans ses messages de 243 octets et n'impose pas de limites sur le nombre de messages transmis par jour.

LoRaWAN propose une méthode différente de Sigfox pour prévenir la perte de messages. Chaque message est transmis une seule fois, mais à l'ensemble des antennes se trouvant à portée, augmentant ainsi la probabilité que le message soit reçu correctement. Cette méthode permet également d'utiliser les différences de temps (TDOA, *Time Difference Of Arrival*)) de réceptions du message auprès des antennes pour estimer la position géographique de l'objet émetteur.

LoRaWAN définit plusieurs classes d'appareils couvrant un large besoin des objets de l'IoT [183] :

— Classe A : Communications bidirectionnelles
Chaque émission d'un objet est suivie par l'ouverture de deux courtes fenêtres de réception.

— Classe B : Communication bidirectionnelle avec réceptions programmées
Même comportement que la classe A avec ouverture de fenêtres d'écoutes à intervalles réguliers synchronisées avec la station.

— Classe C : Communication bidirectionnelle avec fenêtres de réception maximale
Appareil en écoute constante sauf lors de l'émission de messages.

LoRaWAN suscite l'intérêt des industries grâce à son côté non propriétaire et sa possibilité à créer des réseaux privés.

A.3.3 CoAP

Le standard CoAP est un protocole applicatif adapté aux objets contraints avec de faibles capacités tel que les objets des classes 1 et 2 définies par le LWIG (*Light-Weight Implementation Guidance*) [73] dont les performances sont reprises dans la figure A.6.

Nom	Mémoire (RAM)	Espace
Class 1	10 Ko	100 Ko
Class 2	50 Ko	250 Ko

FIGURE A.6 – Tableau récapitulatif des classes d'objets définies par LWIG [73]

Afin de limiter le nombre de communications, CoAP ne maintient pas de connexions entre les objets. L'utilisation de UDP [210] permet de communiquer avec une entité sans avoir échangé avec cette dernière au préalable, permettant ainsi d'économiser de l'énergie.

Le protocole CoAP se situe sur la septième couche du modèle OSI [278] (Application) et s'appuie sur les protocoles IEEE 802.15.4 ou Wifi pour la communication, 6LoWPAN pour le passage de de fréquences radio à IPv6 et UDP pour le transport. La figure A.7 positionne la couche de protocoles CoAP par rapport au modèle OSI dans le cas d'une utilisation de IEEE 802.15.4.

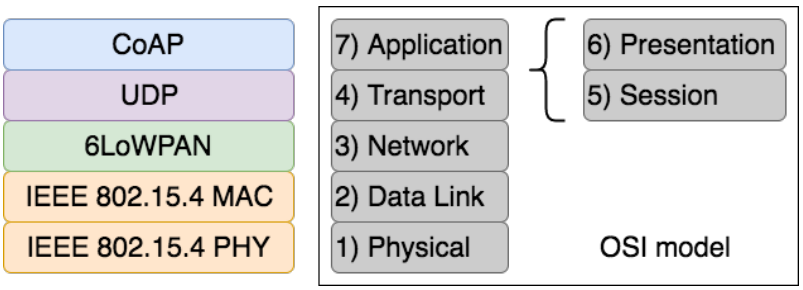


FIGURE A.7 – Position de CoAP par rapport au modèle OSI [161, 57]

CoAP est basé sur une architecture REST [106] comportant quatre méthodes qui permettent une compatibilité réciproque avec les technologies Web. Ces méthodes comportent en paramètre un URI(*Uniform Resource Identifier*) correspondant à l'adresse de la ressource sur laquelle appliquer la méthode (ex : `coap://localhost:5683/testCoAP`).

GET Lecture d'une ressource

POST Écriture d'une ressource

PUT Modification de la ressource

DELETE Suppression de la ressource

L'utilisation du protocole UDP ne garantissant pas l'arrivée des messages à leurs destinataires, CoAP propose de préciser si une réponse du destinataire est attendue. Ainsi, CoAP propose quatre catégories de messages [46, 213] :

Confirmable Requiert la réception d'un message de confirmation de réception (Acknowledgment)

Non-Confirmable Ne requiert pas de confirmation

Acknowledgment Message de confirmation

Reset Le message est réceptionné, mais ne peut être traité

À l'inverse des clients internet classiques qui effectuent du *pooling* (consultation régulière pour rester à jour en cas de changement de la ressource) auprès du serveur, CoAP propose une structure semblable au *design pattern* Observateur. Un client souhaitant consulter une donnée va s'y abonner auprès du serveur et lors d'un changement de cette dernière, le serveur le communiquera au client. Ce procédé permet d'éviter les interrogations régulières par le client.

La découverte des services est également une des fonctionnalités du réseau CoAP. Chaque serveur possède une ressource appelée `.well-known` qui recense les ressources mises à disposition par le serveur. Lorsqu'un nœud souhaite découvrir les ressources disponibles dans son réseau, il peut interroger l'ensemble du réseau en multicast sur cette ressource présente dans chaque serveur. La découverte est également possible en unicast en indiquant directement l'adresse du serveur que l'on souhaite interroger [237, **?**].

Le protocole CoAP ne propose initialement pas de sécurité [238], cependant le protocole DTLS a été ajouté afin de remédier à ce problème. CoAP propose ainsi quatre modes de sécurité décrits par [213] :

NoSec La transmission du message n'est pas sécurisée au niveau de CoAP.

PreSharedKey Les objets possèdent déjà une clé symétrique partagée entre eux.

RawPublicKey Les objets possèdent déjà une liste de clés cryptographiques partagées entre eux afin d'entamer une session DTLS sans certificats.

Certificates Suppose la présence d'une infrastructure de sécurité et permet l'utilisation de cryptographie asymétrique à l'aide de certificats.

Les communications effectuées à l'aide du premier mode (sans sécurité) s'effectuent avec une URI commençant par `coap://` alors que dans les trois autres, la sécurité s'effectue via DTLS (distinguable par l'ajout d'un "s" dans l'URI : `coaps://`). L'utilisation de DTLS au sein de CoAP multiplie par 4 la quantité d'échanges effectués entre un client et un serveur [213, 46] lors de l'établissement d'une communication sécurisée. La figure A.8 représente un accord de sécurisation DTLS (DTLS *handshake*) suivi d'une communication. Si le mode *NoSec* de CoAP est utilisé, seul le dernier échange (*Request*/*Response*) persiste.

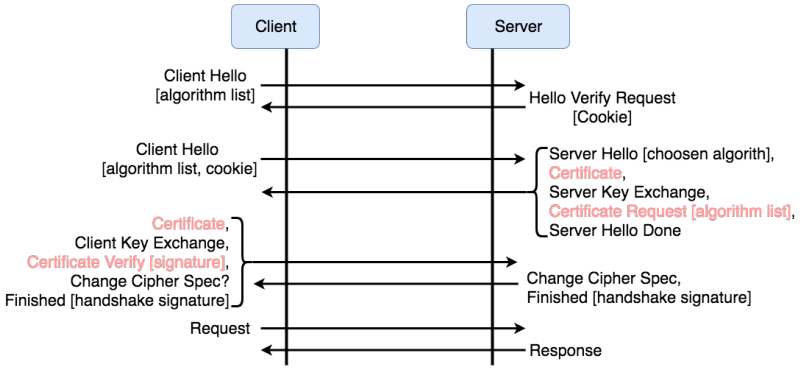


FIGURE A.8 – DTLS handshake avec requête. Les données en rouge ne sont pas transmises dans le mode *PreSharedKey* (source :[259, 207])

La réalisation d'un accord de sécurisation DTLS peut nécessiter plus de 50 secondes dans certains cas [259]. Pittoli et al. [207] proposent une amélioration du processus d'*handshake* DTLS améliorant de 19% le temps de connexion. La sécurisation de CoAP via DTLS reste alors acceptable pour des configurations avec un nombre restreint de DTLS *handshake*. DTLS procure une sécurisation au niveau *Transport* cependant, Vučinić et al. recommandent [259] de s'intéresser à une sécurisation au niveau *Network* ou *Application* pour les réseaux avec un grand nombre de clients.

A.4 Comparatif et sélection

L'Internet des Objets comporte une multitude de protocoles ayant chacun des utilisations particulières. Les réseaux locaux personnels, ou PAN (*Personal Area Networks*) regroupent une grande diversité d'objets dont la sécurité peut être insuffisante. Le tableau A.1 compare 3 des principaux protocoles PAN (à savoir WIfi, BLE et Zigbee) en fonction de leurs fonctionnalités.

	Wifi	**BLE**	**Zigbee**
IEEE standard	IEEE 802.11	IEEE 802.15.1	IEEE 802.15.4
Portée	100m [165, 154]	350-500m [264]	300m [42]
Consommation (Éveillé)	245 mA [154]	35 mA [154]	50 mA [154]
Transfert de données	3.47 Gbits/s [38]	2.1 Mbit/s [34]	250Kbits/s [42]
Topologie du réseau	Point à Point, Étoile	Étoile, Maillé	Maillé

TABLE A.1 – Comparaison des principaux protocoles PAN

La présence d'une topologie de réseau maillé pour les protocoles BLE et Zigbee offre de nombreuses capacités dans le domaine de l'Internet des Objets, notamment concernant la surface de couverture du réseau. Le tableau A.1 présente BLE comme

étant plus intéressant que Zigbee sur les notions de portée, de taux de transfert de données et sur une consommation énergétique plus faible. Il est cependant important de mentionner que l'adoption de Zigbee auprès des réseaux maillés est plus importante. Un compromis devra donc être trouvé entre les performances théoriques offertes par ces protocoles, les performances sub optimales et le taux d'adoption auprès des périphériques de l'Internet des Objets.

Annexe B

Représentations logique des classes et propriétés de l'ontologie FSPlacementOntology

Les tables suivantes représentent respectivement les classes et propriétés de l'ontotologie FSPlacementOntology réalisée au Chapitre 4. Ces représentations ont été générées par l'outil OWL Syntax Converter [1] depuis notre fichier d'ontologie. OWL Syntax Converter est un outil en ligne mis à disposition par l'initiative de recherche Linked Data Finland, mené par le groupe de recherche en informatique sémantique de l'université d'Aalto en collaboration avec l'université d'Helsinki. Cet outil propose de convertir diverses syntaxes depuis le format OWL vers d'autres formats : Turtle, RDF/XML, OWL Functional Syntax, Manchester OWL Syntax, WOL/XML, LaTeX et KRSS2. Les deux tables ci-dessous sont ainsi issues de cette conversion au format LaTeX suite à quoi nous avons effectuer quelques modifications pour faciliter la lecture.

1. `https://www.ldf.fi/service/owl-converter/`

Nom	Représentation logique
CPU	CPU $\sqsubseteq$ ProcessingCapability
CommunicationInterface	
Constraint	
Device	Device $\sqsubseteq \exists$hasCharacteristics.Storage
	Device $\sqsubseteq \exists$hasCharacteristics.CommunicationInterface
	Device $\sqsubseteq \exists$hasCharacteristics.CPU
	Device $\sqsubseteq \exists$hasCharacteristics.Memory
	Device $\sqsubseteq$ System
DeviceCharacteristic	DeviceCharacteristic $\sqsubseteq$ Information
Information	
Memory	Memory $\sqsubseteq$ ProcessingCapability
NamedDevice	NamedDevice $\sqsubseteq$ Device
ObjectiveFunction	
Pairing	Pairing $\sqsubseteq =1$ hasPairedDevice.Device
	Pairing $\sqsubseteq =1$ hasPairedService.Service
PlacementSolution	PlacementSolution $\sqsubseteq \exists$containPairs.Pairing
ProcessingCapability	ProcessingCapability $\sqsubseteq$ DeviceCharacteristic
ResolutionAlgorithm	ResolutionAlgorithm $\sqsubseteq \forall$findSolution.PlacementSolution
SecureElements	SecureElements $\sqsubseteq$ SpecificHardwareComponent
SecurityService	SecurityService $\sqsubseteq$ Service
Service	Service $\sqsubseteq \exists$hasObjectiveFunction.ObjectiveFunction
	Service $\sqsubseteq \exists$hasConstraint.Constraint
SpecificHardwareComponent	SpecificHardwareComponent $\sqsubseteq$ DeviceCharacteristic
Storage	Storage $\sqsubseteq$ ProcessingCapability
System	

TABLE B.1 – Liste des classes constituant l'ontologie FSPlacementOntology

Nom	Représentation logique
containPairs	$\exists containPairs.Thing \sqsubseteq PlacementSolution$
	$\top \sqsubseteq \forall containPairs.Pairing$
dependsOn	$\exists dependsOn.Thing \sqsubseteq Service$
	$\top \sqsubseteq \forall dependsOn.Service$
exposedBy	$exposedBy \equiv exposes^{-}$
	$\exists exposedBy.Thing \sqsubseteq Device$
	$\top \sqsubseteq \forall exposedBy.Service$
exposes	$exposedBy \equiv exposes^{-}$
	$\exists exposes.Thing \sqsubseteq Service$
	$\top \sqsubseteq \forall exposes.Device$
findSolution	$\exists findSolution.Thing \sqsubseteq ResolutionAlgorithm$
	$\top \sqsubseteq \forall findSolution.PlacementSolution$
hasCharacteristics	$\exists hasCharacteristics.Thing \sqsubseteq Device$
	$\top \sqsubseteq \forall hasCharacteristics.DeviceCharacteristic$
hasCommunicationInterface	$\exists hasCommunicationInterface.Thing \sqsubseteq Device$
	$\top \sqsubseteq \forall hasCommunicationInterface.CommunicationInterface$
hasConstraint	$\exists hasConstraint.Thing \sqsubseteq Service$
	$\top \sqsubseteq \forall hasConstraint.Constraint$
hasObjectiveFunction	$\exists hasObjectiveFunction.Thing \sqsubseteq Service$
	$\top \sqsubseteq \forall hasObjectiveFunction.ObjectiveFunction$
hasPairedDevice	$\exists hasPairedDevice.Thing \sqsubseteq Pairing$
	$\top \sqsubseteq \forall hasPairedDevice.Device$
hasPairedService	$\exists hasPairedService.Thing \sqsubseteq Pairing$
	$\top \sqsubseteq \forall hasPairedService.Service$
interactsWith	$\exists interactsWith.Thing \sqsubseteq Device$
	$\top \sqsubseteq \forall interactsWith.Device$
involves	$\exists involves.Thing \sqsubseteq Constraint$
	$\top \sqsubseteq \forall involves.Information$

TABLE B.2 – Liste des propriétés de l'ontologie FSPlacementOntology

Annexe C

Cardinal d'une différence symétrique : une distance - démonstration

En Section 5.4.3 nous avons énuméré les conditions nécessaires et suffisantes pour qu'une fonction constitue une distance. Nous avons alors utilisé le cardinal de la différence symétrique pour effectuer un calcul de distance entre deux signatures. Dans cette annexe nous effectuons un rappel de ces conditions et démontrons que le cardinal de la différence symétrique entre deux ensembles correspond effectivement à une distance.

La différence symétrique est représentée par le symbole $\triangle$. Soit A et B deux ensembles, la différence symétrique entre ces deux ensembles correspond à l'équation suivante :

$$A \triangle B = \{c \in A \wedge x \notin B\} \cup \{x \notin A \wedge x \in B\}$$

Définissons maintenant une fonction D telle que $D(A, B) = |A \triangle B|$. Nous pouvons affirmer que D est une fonction de distance uniquement si les trois conditions suivantes sont valides :

1. $D(A, B) = D(B, A)$
2. $D(A, B) = 0 \Leftrightarrow A = B$
3. $D(A, C) \leq D(A, B) + D(B, C)$.

Démontrons désormais que la fonction D satisfait les conditions énumérés ci-dessus.

$D(A, B) = D(B, A)$

$$\begin{array}{lcl} A\triangle B & = & \{x \in A \wedge x \notin B\} \cup \{x \notin A \wedge x \in B\} \\ & = & \{x \notin A \wedge x \in B\} \cup \{x \in A \wedge x \notin B\} \\ A\triangle B & = & B\triangle A \\ |A\triangle B| & = & |B\triangle A| \\ D(A, B) & = & D(B, A) \end{array}$$

$D(A, B) = 0 \Leftrightarrow A = B$

$$\begin{array}{lcl} A = B & \Rightarrow & A\triangle B = \emptyset \\ & \Rightarrow & |A\triangle B| = 0 \\ A\triangle B & = & \{x \in A \wedge x \notin B\} \cup \{x \notin A \wedge x \in B\} \\ A\triangle B = 0 & \Rightarrow & \nexists x : (x \in A \wedge x \notin B) \cup (x \notin A \wedge x \in B) \end{array}$$

$D(A, C) \leq D(A, B) + D(B, C)$

Soit $x \in A\triangle C$, nous observons deux cas :

1. $x \in A$ et $x \notin C$
   - Si $x \in B$ alors $x \notin C$ et donc $x \in B\triangle C$
   - Si $x \notin B$ alors $x \in A$ et donc $x \in A\triangle B$
2. $x \notin A$ et $x \in C$
   - Si $x \in B$ alors $x \notin A$ et donc $x \in A\triangle B$
   - Si $x \notin B$ alors $x \in C$ et donc $x \in B\triangle C$

Nous distinguons que dans tous les cas, $x \in A\triangle B \cup B\triangle C$. Nous pouvons alors en déduire que :

$$\begin{array}{lcl} A\triangle B & \subseteq & A\triangle B \cup B\triangle C \\ |A\triangle B| & \leq & |A\triangle B| \cup |B\triangle C| \\ D(A, C) & \leq & D(A, B) + D(B, C) \end{array}$$

Nous constatons que le cardinal de la différence symétrique entre deux ensembles satisfait les trois conditions requises pour être considéré comme une distance mathématique. Nous pouvons ainsi confirmer que notre fonction D est une distance.

Table des figures

1.1 Le rôle d'interface de l'IoT 6
1.2 Principe de l'actionneur 7
1.3 Courbe des technologies émergentes, 2011 (source : Gartner 2011) 11
1.4 Diversité des domaines d'application de l'Internet des Objets en 2017 (source : Beecham Research) 12
1.5 Relations simplifiées entre M2M, IoT et IoE 13
1.6 Nouvelle surface d'attaque suite à l'ajout d'un objet connecté au sein d'une infrastructure sécurisée. 15
1.7 Nombre d'objets ICS indexés par SHODAN (à la date du 1er avril 2020) : 55 164 17

2.1 Mise en correspondance des architectures IoT, Fog et SoA. 22
2.2 Architecture de fog computing (source : [145]) 24
2.3 Mise en relation des étapes de traitement des données avec les architectures fog et SoA. 25
2.4 Classification des protocoles d'authentification pour l'IoT d'après leurs cryptosystèmes (source : [104]) 29
2.5 Union des menaces de Roman et al. [222] et Hodo et al. [140] 35
2.6 Classification d'attaques sur l'IoT proposée par Andrea et al. (source : [49]) 36
2.7 Moyens et objectifs d'un attaquant 40

3.1 Aperçu des modèles pour la description du problème de placement de service (SPP) (inspiré de [229]) 48
3.2 Taxonomie du placement de Service (inspiré de [229]) 50
3.3 Nos conditions de placement selon la taxonomie du placement de Service de Salaht et al. [229]) 52
3.4 Exemples de chemins d'attaque par un attaquant de type *outsider*. . . . 54
3.5 Exemple de services de sécurité (nœuds verts) sur le modèle considéré. . 55
3.6 Ensemble dominant (nœuds rouges) localisé en dehors du flux d'informations. 57
3.7 Différences entre valeurs de centralité sur un graphe donné. 61
3.8 Composante préservée (en vert) de la matrice d'adjacence OOR. 64
3.9 Illustration des chemins sécurisés. 68

3.10 Ensemble dominant (rouge) identifié avec notre méthodologie sur le graphe de la Figure 3.6. 68
3.11 L'impact du ratio de poids entre les noeuds à forte et basse priorité sur les critères d'évaluation de l'ensemble dominant. 70
3.12 Qualité des ensembles dominants centraux sur notre dataset pondérée (ratio de poids : 1.2). 71
3.13 Distribution des degrés du graphe IoT 75
3.14 Distribution des degrés du graphe Ego-Facebook [182]. 76

4.1 Illustration des concepts à définir dans notre ontologie. 82
4.2 Principe et enchainement des éléments de l'ontologie pour l'identification et la comparaison des solutions de placement de services. 82
4.3 Processus de constitution de notre base de connaissance. 96
4.4 Résolution d'un problème de placement utilisant la base de connaissance. 99
4.5 Processus de comparaison des solutions de placement. 100
4.6 Visualisation de FSPlacementOntology d'après l'outil de visualisation WebVOWL 102
4.7 Visualisation de la base de connaissance en VOWL 105
4.8 Aperçu de l'interface de CSPS. 109
4.9 Panneau de configuration de CSPS. 109

5.1 Exemple d'encodage via des bijections (F et G) aléatoires. 116
5.2 Catégories d'implémentations cryptographiques boîte blanche 116
5.3 Exemple d'utilisation de white box sur l'architecture OSCAR 122
5.4 Implémentation *white-box* protégée par un mot de passe 123
5.5 Présentation d'une implémentation white-box avec calcul de distance intégré. 124
5.6 Ancrage local d'une implémentation white-box. 128
5.7 Principe de génération d'une implémentation white-box ancrée. 129
5.8 Présentation du fonctionnement de l'ancrage d'implémentation white-box avec partage du même programme. 130
5.9 Présentation du fonctionnement de l'ancrage d'implémentation white-box avec partage de la même clé de chiffrement. 131
5.10 Topologie du réseau initial 132
5.11 Environnement local (vert) du noeud 1. 133
5.12 Représentation visuelle de la différence symétrique (bleu) entre deux signatures locales pour un nœud n_i. 134
5.13 Scénario d'ancrage avec faible variation de signature locale. 136
5.14 Scénario d'ancrage avec implémentation subtilisé. 137
5.15 Scénario d'ancrage avec hôte subtilisé. 138
5.16 Scénario d'ancrage avec remplacement de l'hôte. 139

6.1 Étapes du framework de sécurisation adaptative des objets de l'IoT. . . . 146
6.2 Illustration du framework de sécurisation adaptatif des objets de l'IoT. . 147
6.3 Positionnement des contributions sur le bloc sémantique de notre framework de sécurisation adaptatif des objets de l'IoT. 150

A.1 Position des principaux protocoles IoT sur les modèles OSI et TCP/IP (inspiré par [39]) 179
A.2 Représentation du *4 way handshake* utilisé dans WPA 181
A.3 Les structures de réseau Zigbee 185
A.4 Procédure d'ajout d'un appareil dans un réseau Zigbee centralisé et renouvellement des clés 186
A.5 Échange de clés dans le protocole Z-Wave 187
A.6 Tableau récapitulatif des classes d'objets définies par LWIG [73] 190
A.7 Position de CoAP par rapport au modèle OSI [161, 57] 190
A.8 DTLS handshake avec requête. Les données en rouge ne sont pas transmises dans le mode *PreSharedKey* (source :[259, 207]) 192

Liste des tableaux

2.1 Mécanismes de sécurisation de l'architecture IoT d'après [272, 66, 254] . . 28
2.2 Aperçu des attaques relatives à chaque couche de l'architecture IoT (d'après [169]). 35
2.3 Caractéristiques de comparaison d'attaques sur l'IoT d'après [92] 37
2.4 Qualification des moyens et objectifs à partir des profils d'attaquants définit par Onik et al. [202] 41

3.1 Distribution des catégories d'objets de notre graphe. 64
3.2 Distribution des poids des arêtes dans notre graphe 66
3.3 Comparaison des résultats d'ensemble dominant centraux en fonction de l'ordre de la liste des sommets (ordonnée par centralités et aléatoire). . . 70
3.4 Comparaison des propriétés de graphes entre notre graphe IoT et Ego-Facebook [182]. 75
3.5 Comparaison des ensembles dominants centraux entre notre graphe IoT et Ego-Facebook [182]. 77

4.1 Quelques métriques de la base de connaissance (obtenues avec Protégé [197]) 107
4.2 Exemple d'une comparaison entre plusieurs algorithmes de résolution et scores de fonctions objectives. 110

A.1 Comparaison des principaux protocoles PAN 192

B.1 Liste des classes constituant l'ontologie FSPlacementOntology 196
B.2 Liste des propriétés de l'ontologie FSPlacementOntology 197

Depuis maintenant plusieurs années, nous assistons à l'essor de l'Internet des Objets (IdO ou IoT en anglais). Suite à de récentes attaques sur ces systèmes, les études ont démontré que la sécurité de ces appareils était majoritairement insuffisante. Afin de remédier à ce problème, nous devrions idéalement mettre en place des mécanismes de sécurité sur l'ensemble des périphériques IoT, cependant cette solution n'est pas toujours envisageable.

Une approche alternative, pour sécuriser ces systèmes, consiste à déployer des services de sécurité en bordure du réseau afin de rapprocher les mécanismes de sécurité au plus près des périphériques non sécurisés.

Le but de cette thèse est de constituer un framework de sécurisation adaptative des objets de l'IoT qui repose sur le positionnement de services de sécurité. Ce travail se décompose en trois contributions qui touchent chacune des aspects différents de notre approche.

La première contribution élabore une stratégie de déploiement de services de sécurité qui minimise leurs coûts de déploiement. Cette approche traduit nos contraintes de positionnement sous la forme d'un problème de graphes que nous proposons de résoudre à l'aide d'outils de théorie des graphes.

La seconde contribution formalise les problèmes de placement de services pour les modéliser sous la forme d'une ontologie. Cette dernière est alors utilisée pour résoudre ces problèmes et permettre de comparer leurs différentes solutions.

La troisième contribution se focalise sur les services de sécurité qui implémentent de la cryptographie *whitebox*. Dans cette contribution, nous proposons un mécanisme d'ancrage de ces implémentations sur un réseau IoT afin de prévenir les attaques par extraction de code ainsi que le vol du périphérique.

Finalement, nous proposons un framework de sécurisation adaptative des objets de l'IoT dans lequel nous positionnons l'ensemble des contributions réalisées pendant cette thèse.

Adaptive security of IoT devices using software (White box) and combined (hardware and software) methods.

The Internet of Things (IoT) has been on the rise for several years now. Following recent attacks on these systems, studies have shown that most of these devices were not sufficiently secured. The ideal solution to this problem would be to provide security mechanisms on all IoT devices however, this solution is not always achievable.

An alternative strategy to secure these systems would be to deploy security services at the edge of the network to bring the security mechanisms as close as possible to unsecured devices.

The purpose of this thesis is to design an adaptive security framework for IoT devices relying on security services positioning. This work is divided into three contributions, each of which affects different aspects of our approach.

The first contribution provides a strategy for deploying security services that minimizes the cost of deployment. This method expresses our positioning constraints into a graph problem which we suggest solving using graph theory.

The second contribution formalizes the service placement problems and models them into an ontology. The latter is then used to solve those problems and to compare their different solutions.

The third contribution focuses on security services that implement *whitebox* cryptography. In this contribution, we present a mechanism for anchoring these implementations on an IoT network to prevent code lifting attacks and device theft.

Finally, we present an adaptive security framework for IoT objects in which we position all the contributions made during this thesis.

Spécialité Informatique. Mots-clés : Sécurité IoT, Placement de services, Théorie des graphes, Ontologies, Cryptographie whitebox, Ancrage réseau

---

ENSICAEN - UNICAEN - CNRS - GREYC UMR 6072, F-14050 Caen, France