République Algérienne Démocratique et Populaire
Université Saad Dahlab Blida 1
Faculté des Sciences
Département d'informatique

Thème :

Mitigation de l'attaque numéro de version contre les réseaux IoT basés sur le protocole de routage RPL

MEMOIRE DE MASTER

Domaine : MI

Filière : Informatique

Option : Sécurité des Systèmes d'informations

Organisme d'accueil : CERIST

Réalisé par :
Ihsane Djabrouhou
Ikram Boursas

Encadreur : Dr. Faiza Medjek
Promoteur : Dr. Zakaria Sahnoune
Président : Pr.Mohamed Ould-Khaoua
Examinateur : Dr.Fella Bey

Année universitaire : 2020/2021

Remerciements

Nous tenons à remercier tout d'abord ALLAH le Tout-Puissant de nous avoir donné la santé, le courage et la volonté pour réaliser ce mémoire.

Nous remercions et témoignons notre reconnaissance à notre encadreur : **Mme. Faiza MEDJEK**, et notre promoteur : **Mr. Zakaria SAHNOUNE**, de leurs orientations, leurs précieux conseils, leurs soutiens constants et leurs aides qui nous ont permis de mener à bien ce travail.

Nous tenons également à exprimer une reconnaissance aux membres du jury pour avoir accepté d'examiner et de porter leur jugement sur notre travail.

Nos vifs remerciements à nos familles pour nous avoir aidées à surmonter tous les obstacles et forger sur dent à travers les difficultés vécues tout au long de cette période de travail.

Nos vifs remerciements à nos proches amis qui nous ont vivement soutenues et encouragées au cours de la réalisation de ce modeste travail, et nos remerciements à toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicaces

Je dédie ce modeste travail :

À mes chers parents, pour tous leurs amours, leurs patiences, leurs grands sacrifices et leurs soutiens tout au long de mes études,

À mes chères sœurs Nesrine, Wiam, Romaissa pour leurs encouragements permanents, et leur soutien moral,

À mon cher frère Ishak, pour son appui et ses encouragements,

À toute ma famille qui m'a donné de l'amour et de la vivacité,

À mes amis Ihsane, Nafissa, Ayoube qui m'ont toujours encouragé, et à qui je souhaite plus de succès,

À mes professeurs de mémoire, pour leurs suivis, leurs soutiens et leurs conseils.

Ikram

C’est grâce à Allah seul qu’on a pu réaliser ce travail. Je le dédie :

À Ma très chère mère, qui a toujours été présente pour moi, et qui sans cesse veille sur moi avec ses prières, pour ses grands sacrifices et tout l’amour qu’elle me porte,

À Mon très cher père, pour tous ses conseils et pour toute la confiance qu’il a mise en moi et pour son dévouement pour mon bonheur,

À Mes chères sœurs Hayet, Nadia, Faiza, Ikram, pour leur patience et leur soutien,

À mes petits neveux et nièces adorés "Iline, Mounaime et Sana" pour m’avoir fait partager leur joie de vivre et m’avoir ainsi soutenu dans mes efforts,

À Tous mes amis(es), plus particulièrement Manel, Yasmine, Ayyoub,

À Ma binôme BOURSAS Ikram et toute sa famille.

Ihsane

Résumé

Le concept de l'Internet des objets implique le déploiement de réseaux à faible puissance et avec perte (LLNs), composé de milliers de nœuds contraints et de liaisons non fiables. Ces réseaux LLNs présentent de nouveaux défis surtout concernant le routage. L'IETF a conçu le protocole de routage pour les réseaux à faible puissance et avec perte (RPL) pour prendre en charge ces réseaux contraints.

RPL organise des réseaux à faible puissance et avec perte sous la forme d'un ou de plusieurs graphes acycliques orientés dirigés vers une destination (DODAGs). Chaque DODAG se voit attribuer un numéro de version. Le but du numéro de version est de garantir qu'il existe des chemins libres de boucles vers le nœud root, que les entrées de table de routage des nœuds dans le DODAG ne sont pas obsolètes et qu'il n'y a aucune incohérence dans le DODAG. Le nœud root d'un DODAG incrémente le numéro de version en cas d'incohérence. Cela nécessite un processus de réparation global et le DODAG est reconstruit. Un nœud malveillant peut publier un faux numéro de version dans son message de contrôle pour forcer une réparation globale. Malheureusement, les services de sécurité actuellement disponibles dans RPL ne le protégeront pas contre un nœud interne compromis qui peut construire et diffuser de faux messages.

Dans ce mémoire, nous proposons une stratégie de détection basée sur la cryptographie avec des algorithmes dédiés capables de détecter l'attaque et identifier les nœuds malveillants à l'origine de l'attaque. Les résultats de la simulation montrent que l'approche proposée défend de manière fiable l'attaque du numéro de version du DODAG.

Mots-clés : Internet des Objets, Sécurité, LLN, RPL, Attaque Numéro de Version.

Abstract

The concept of the Internet of Things involves the deployment of Low-power and Lossy Networks (LLNs), made up of thousands of constrained nodes and unreliable links, presenting new challenges especially regarding routing. The IETF designed the Routing Protocol for Low Power Lossy Networks (RPL) to support these constrained networks.

RPL organizes the low-power and lossy networks in the form of one or more destination-oriented directed acyclic graphs (DODAGs). Each DODAG is assigned a version number. The purpose of the version number is to ensure that there are free loop paths to the root node, also that the node routing table entries in the DODAG are not out of date and there are no inconsistencies in the DODAG. The root node of a DODAG increments the version number in case of inconsistency. This requires a comprehensive repair process and the DODAG is rebuilt. A malicious node can post a false version number in its control message to force a global repair. Unfortunately, the security services currently available in RPL will not protect against a compromised internal node that can construct and broadcast fake messages.

In this thesis, we propose a cryptography-based detection strategy with dedicated algorithms capable of detecting the attack and identifying the malicious nodes at the origin of the attack. The simulation results show that the proposed approach reliably defends the DODAG version number attack.

Keywords: Internet of Things, Security, LLN, RPL, Version Number Attack.

ملخص

ينطوي مفهوم إنترنت الأشياء على نشر شبكات منخفضة الطاقة وعرضة بشكل عام لفقدان الحزم (LLNs) ، المكونة من آلاف العقد المقيدة والروابط غير الموثوقة. تمثل شبكات LLNs هذه تحديات جديدة خاصة فيما يتعلق بالتوجيه. صممت IETF بروتوكول التوجيه للشبكات منخفضة الطاقة وعرضة لفقدان الحزم (RPL) لدعم هذه الشبكات المقيدة.

ينظم الـ RPL الشبكات المنخفضة الطاقة والعرضة لفقدان الحزم في شكل واحد أو أكثر من الرسوم البيانية الحلقية الموجهة نحو الوجهة (DODAGs) . يتم تعيين رقم إصدار لكل DODAG. الغرض من رقم الإصدار هو التأكد من وجود مسارات حرة من الحلقات إلى العقدة الجذرية ، وأن إدخالات جدول توجيه العقدة في DODAG ليست قديمة ، وأنه لا توجد أي تناقضات في DODAG. تزيد العقدة الجذرية لـDODAG من قيمة رقم الإصدار في حالة عدم الاتساق. هذا يتطلب عملية إصلاح شاملة وإعادة بناء DODAG. يمكن لعقدة ضارة نشر رقم إصدار خاطئ في رسالة التحكم الخاصة بها لفرض إصلاح عام. لسوء الحظ، فإن الخدمات الأمنية المتوفرة حاليًا في الـRPL لن تحميها من عقدة داخلية مخترقة يمكنها إنشاء وبث رسائل مزيفة.

في هذه الاطروحة ، نقترح استراتيجية كشف قائمة على التشفير مع خوارزميات مخصصة قادرة على اكتشاف الهجوم وتحديد العقد الخبيثة في أصل الهجوم. تظهر نتائج المحاكاة أن النهج المقترح يدافع بشكل موثوق عن هجوم رقم الاصدار.

الكلمات المفتاحية: إنترنت الأشياء، الأمن، LLN، RPL، هجوم رقم الإصدار.

Table des matières

Liste des figures

Liste des tableaux

Abréviations

6LBR	IPv6 Low-power Border Router
6LOWPAN	IPv6 Low power Wireless Personal Area Networks
AH	Authentication Header
CoAP	Constrained Application Protocol
DAG	Directed Acyclic Graph, : directed acyclic graph
DAO	DODAG Advertisement Object
DAO-Ack	DODAG Advertisement Object Acknowledgment
DDoS	Distributed Denial Of Service
DIO	DODAG Information Objet
DIS	DODAG Information Sollicitation
DODAG	Destination Oriented Direct Acyclic Graph
DoS	Denial of Service
DTLS	Datagram Transport Layer Security
ESP	Encapsulating Security Payload
FO	Objective Function
HTTP	HyperText Tranfert Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	International Engineering Task Force
IKE	Internet Key Exchange
IoT	Internet of Things
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
LBR	LowPAN Border Router
LLN	Low Power and Lossy Networks
M2M	Machine-to-Machine
MAC	Medium Access Control
MITM	Man In The Middle

MP2P	Multi-Point to Point
MRHOF	Minimal Rank With Hystérésis Objective Function
OCP	Objective Code Point
OF0	Objective Function 0
OSI	Open Systems Interconnection
P2MP	Point to Multi-Point
P2P	Point to Point
PHY	Physical layer
QoS	Quality of service
RFID	Radio-Frequency IDentification
RoLL	Routing over Low-power and Lossy Network
RPL	IPv6 Routing Protocol for Low-power and Lossy Network
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VN	Version Number
VNA	Version Number Attack
WSN	Wireless Sensor Networks

Introduction générale

Ces dernières années, les chercheurs se sont de plus en plus intéressés au domaine de recherche de l'Internet des objets (IoT en anglais), en raison de sa large diffusion dans de nombreux domaines de la vie, tels que les réseaux intelligents, les villes intelligentes, les environnements intelligents, l'agriculture intelligente et les soins de santé, qui améliorent la vie quotidienne des personnes.

L'IoT présente un nouveau paradigme du futur Internet qui vise à fournir une communication interactive entre une variété d'appareils et leurs objets associés via des normes et des protocoles de réseau à tout moment et en tout lieu [1]. Ces appareils forment des réseaux avec différentes topologies (par exemple, en étoile, maillé) en utilisant diverses technologies de communication (par exemple, IEEE 802.15.4, Bluetooth, Wifi, LoRa, cellulaire, etc.) et ils peuvent détecter à distance et collecter des données brutes à partir de leur environnement physique pour le traitement des données et la prise de décision [2]. Cependant, le caractère contraint de ces appareils rend l'utilisation des protocoles de l'Internet difficile dans leurs formes natives pour établir une connectivité mondiale. Pour surmonter ce problème, le groupe ROLL du groupe IETF a introduit et normalisé le protocole de routage pour les réseaux à faible puissance et faible liaison de communication (RPL) afin de s'adapter aux contraintes de ressources des appareils intelligents.

Le protocole de routage RPL (Routing Protocol for Low-power and Lossy Networks) est considéré comme un protocole de routage léger pour les appareils IoT, et il est conçu pour être un protocole de réseau interopérable et simple pour répondre aux exigences des appareils et des applications à ressources limitées qui sont interconnectés via des réseaux maillés à sauts multiples tels que dans les environnements industriels, domestiques et urbains [3] [4]. Il permet également d'utiliser efficacement l'énergie des appareils intelligents et d'établir une topologie et un routage flexibles des données [5].

De plus, le protocole RPL prend en charge plusieurs topologies de routage, appelées DODAG, construites à l'aide de différentes fonctions objectives. La maintenance d'un DODAG optimisé est assurée par deux mécanismes à savoir, la réparation globale et locale. La racine (root en anglais) initie la réparation globale en incrémentant un compteur spécial, nommé numéro de version (VN). Une telle mise à jour est propagée via le message de contrôle RPL DIO (DODAG Information Object) et le DODAG est reconstruit en fin du

processus. Dans RPL, seul le root a le droit de changer le VN et donc démarrer le mécanisme de réparation global [6]. Un nœud malveillant peut publier un faux numéro de version dans son message de contrôle DIO pour forcer une réparation globale. Ce type d'action est connu sous le nom d'attaque par numéro de version (VNA). VNA est une menace très sérieuse pour les réseaux et les applications IoT basés sur RPL, car elle peut entraîner l'épuisement des ressources limitées des nœuds et affecter gravement les performances du réseau. Dans ce contexte, il est nécessaire d'adapter les nœuds exécutants RPL pour répondre d'une manière sécurisée à une sollicitation à la reconstruction globale de la topologie, et ainsi éliminer les risques que des nœuds malveillants endommagent le réseau.

Dans ce mémoire, nous étudierons l'attaque par numéro de version, les techniques existantes pour la détection de cette attaque, puis nous proposerons et implémenterons une approche légère (lightweight en anglais) et efficace pour contrer VNA.

Notre mémoire est structuré en cinq chapitres comme suit :

Chapitre 1 : Est consacré à la présentation de l'Internet des objets et leurs principaux caractéristiques, leurs différents protocoles ainsi que leurs différents domaines d'applications et la sécurité de cette technologie.

Chapitre 2 : Est une présentation du protocole de routage RPL, ses principaux concepts, ses fonctionnements et les mesures de sécurité pour protéger ce protocole.

Chapitre 3 : Est une étude de l'attaque par numéro de version et une présentation de notre approche contre cette attaque.

Chapitre 4 : Sera consacré pour l'implémentation et la simulation de notre proposition sous le simulateur Cooja/Contiki.

Chapitre 5 : Expose les différents résultats de simulation et leurs discussions.

Nous terminons notre mémoire par une conclusion générale, ainsi que quelques perspectives pour des travaux futurs.

Chapitre I : Introduction à L'internet des Objets et Sécurité

Table des matières

I.1 Introduction

L'Internet des objets est la nouvelle ère de la mise en réseau et communication intelligente qui est devenue un domaine attractif pour les chercheurs en raison de sa vaste collection d'applications et de sa facilité de déploiement dans plusieurs domaines de la vie réelle [7]. Entre autres, il rend les objets qui nous entourent intelligents en leur offrant la

faculté de communiquer entre eux pour atteindre des objectifs communs dans de nombreux domaines d'application, mais tout ce qui est connecté à l'Internet peut être exposé à des cyber-attaques.

Dans ce chapitre, nous présentons l'IoT, leurs domaines d'applications, caractéristiques et protocoles. De plus, nous examinerons les problèmes de sécurité de l'IoT et leurs mesures de protection.

I.2 Internet des objets

I.2.1 Historique

En 1989, Mark Weiser, professeur à Berkeley, avait une vision d'un monde où la technologie s'intègre dans les objets de la vie quotidienne. Sa vision a pris forme avec le développement de l'informatique et l'électronique [8]. Elle a été nommée « Internet des Objets » ou « Internet of Things (IoT) » en anglais. Ce terme a été pour la première fois utilisé par K. Ashton en 1999, co-fondateur d'Auto-ID Center au MIT dans l'une de ses présentations [9]. Il y décrit sa vision de la façon dont les objets et les personnes du monde physique peuvent être recensés et gérés informatiquement grâce à la technologie RFID (Radio-Frequency IDentification). Depuis lors, ce concept a beaucoup évolué grâce aux énormes progrès effectués dans divers domaines technologiques, allant de celui des microcontrôleurs, de la nanotechnologie, des capteurs et actionneurs à celui des technologies sans fils. Il fait aujourd'hui référence à la possibilité de connecter tout objet du quotidien à l'Internet traditionnel [10]. L'intérêt pour ces objets a énormément grandi au cours des dernières années. En effet, comme le montre la Figure I.1, malgré la pandémie actuelle du Covid-19, le marché de l'IoT continue de croître. En 2020, pour la première fois, il y avait plus de connexions IoT (par exemple, voitures connectées, appareils domestiques intelligents, équipements industriels connectés) que de connexions non IoT (smartphones, ordinateurs portables et ordinateurs). Sur les 21,7 milliards d'appareils connectés actifs dans le monde, 11,7 milliards (soit 54%) ont été des connexions d'appareils IoT (à la fin de 2020). D'ici 2025, il devrait y avoir plus de 30 milliards de connexions IoT, soit près de 4 appareils IoT par personne en moyenne.

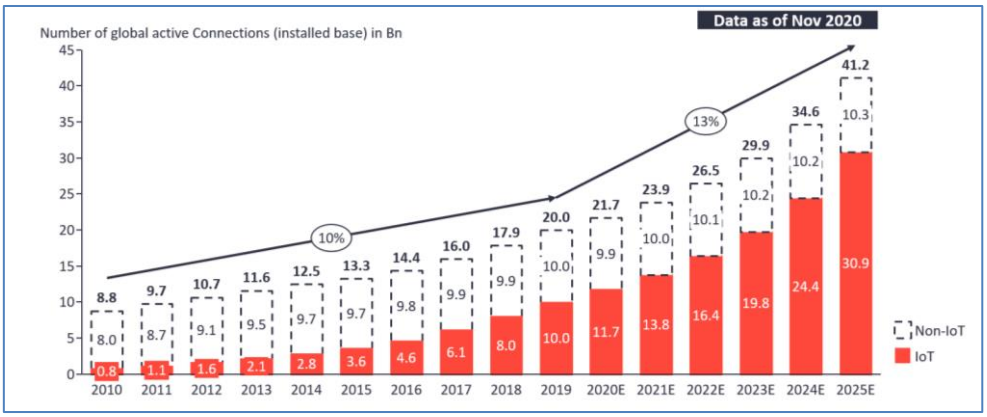


Figure I.1: Nombre total de connexions d'appareils en 2020 [11].

I.2.2 Définition

L'IoT est une technologie qui permet de connecter n'importe quels ensembles d'objets du monde physique entre eux à travers l'Internet et /ou des réseaux locaux comme les réseaux de capteurs sans fil (WSN), pas seulement des dispositifs électroniques, mais consiste à intégrer et embarquer des capteurs et systèmes intelligents dans les divers produits, pour récupérer les informations et les données (sur leur identité, leurs caractéristiques et leur environnement, etc.) [12].

L'IoT attribue à chaque objet du monde réel une identification unique sous forme d'une étiquette lisible par des dispositifs mobiles sans fil.

L'IoT définit un monde dans lequel les objets peuvent communiquer automatiquement entre eux et avec des ordinateurs, prendre des décisions intelligemment et même offrir à un utilisateur distant le contrôle de ses objets afin de fournir des services pour divers secteurs (par exemple domotique, santé, industrie, etc.) [13]. La FigureI.2 est un exemple des nouveaux objets connectés qui sont aujourd'hui associés aux objets traditionnels.

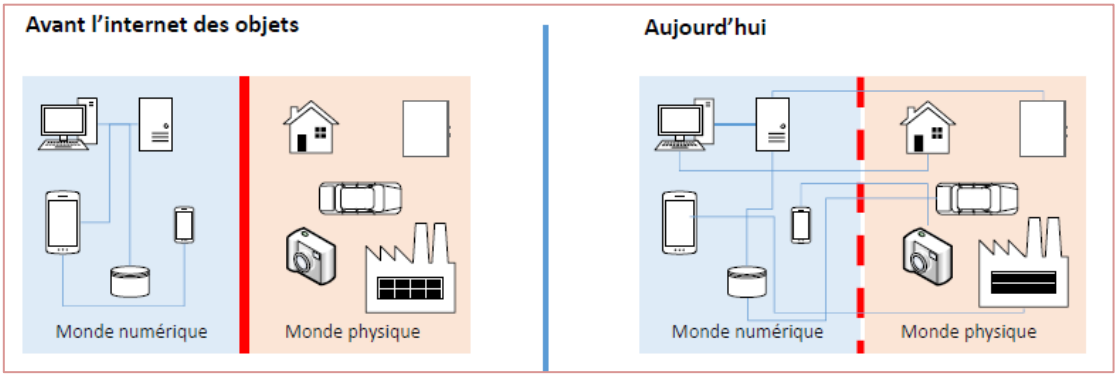


Figure I.2: Fonction entre le monde physique et le monde numérique avant et après L'IoT.

I.3 Pourquoi l'IoT

L'IoT pourrait faciliter tous les aspects de notre vie. Pour faire simple, elle permet aux machines de supprimer une partie de l'effort manuel en exécutant les tâches à un temps donné ou selon une condition précise. Cela réduit les coûts et rend la vie plus confortable. Des machines à café automatiques, une production industrielle qui répond rapidement à la demande, des voitures autonomes et des bracelets qui détectent et signalent immédiatement les maladies, les applications possibles couvrent un large éventail de domaines de la vie. De nombreuses activités peuvent être mieux planifiées sur la base des données collectées par les objets en réseau. En particulier, en combinaison avec des méthodes de l'intelligence artificielle, les objets mis en réseau via l'IoT fonctionnent de manière plus fiable et, surtout, plus rapide que les humains.

I.4 Domaines d'application de l'IoT

Plusieurs domaines d'application sont touchés par l'IoT. Les exemples courants sont présentés dans la FigureI.3. Parmi ces principaux domaines, nous citons les suivants :

- **Villes intelligentes :** circulation routière intelligente, transports intelligents, collecte des déchets, cartographies diverses (bruit, énergie, etc.).
- **Environnements intelligents :** prédiction des séismes, détection d'incendies, qualité de l'air, etc.
- **Sécurité et gestion des urgences :** radiations, attentats, explosions, etc.
- **Logistique :** facilite les décisions de la gestion d'approvisionnement, suivre des éléments spécifiques dans le processus de la livraison, etc.
- **Contrôle industriel :** mesure, pronostic et prédiction des pannes, dépannage à distance.
- **Santé :** suivi des paramètres biologiques à distance.
- **Agriculture intelligente, domotique, applications ludiques,** etc.

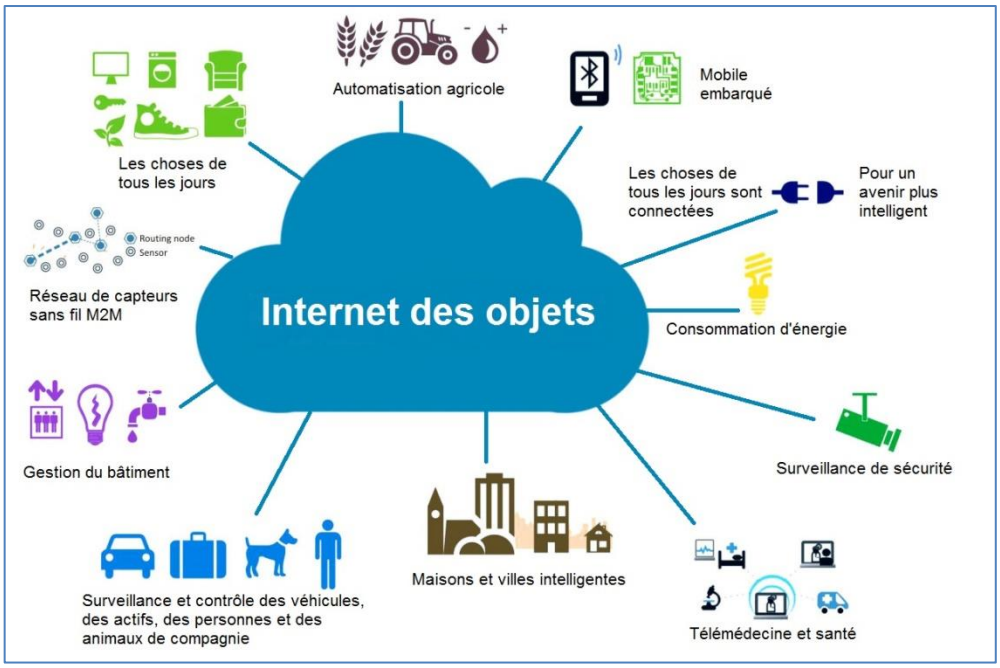


Figure I.3: Domaines d'application de l'IoT.

I.5 Caractéristique de l'IoT

Au cours des dernières années, l'IoT est devenu l'une des technologies les plus importantes du 21e siècle attirant l'attention de nombreux experts, gouvernements et industries. Comme l'importance de l'IoT augmente de jour en jour, différentes caractéristiques doivent être prises en considération. Ces derniers sont illustrés dans la FigureI.4.

- **La mobilité**

La mobilité est un grand défi pour les implémentations IoT, où la plupart des appareils tels que les smartphones ont un degré élevé de mobilité. En effet, la mobilité est une caractéristique très demandée dans le monde de l'IoT, mais elle rend également l'IoT vulnérable à de nombreux risques en termes de sécurité [14].

- **L'évolutivité**

L'évolutivité de l'IoT fait référence à la capacité d'ajouter de nouveaux appareils, services et fonctions pour les clients sans affecter négativement la qualité des services existants. [15] Néanmoins, la propriété d'évolutivité de l'IoT nécessite la mise en place de cadres de sécurité flexibles et innovants qui peuvent réduire le risque de sécurité [14].

- **Hétérogénéité**

Ce qui rend l'IoT spécial, c'est la multiplicité des types d'appareils et des technologies participant dans un même réseau ou sur des réseaux différents [16].

- **Connectivité et ubiquité**

Une caractéristique importante de l'IoT est de permettre la communication et la connexion entre tous les objets. De plus, cette connectivité pourra avoir lieu n'importe quand et n'importe où, et permet également la compatibilité d'accès au réseau et d'échanger les données.

- **Auto-organisation et autoréparation**

Étant donné que le nombre d'objets IoT et leurs états de connexion et d'emplacement changent de manière dynamique, les objets IoT intelligents sont équipés d'une intelligence embarquée leur permettant de réagir de manière autonome et de s'auto-organiser en réseaux ad hoc transitoires selon des situations, états et contexte actuel. Les intrus peuvent exploiter ces caractéristiques pour déclencher plusieurs attaques contre les réseaux IoT [17].

- **Interopérabilité**

L'interopérabilité est très importante dans l'IoT du fait qu'il se compose d'un grand nombre d'appareils hétérogènes appartenant à différentes plates-formes qui communiquent avec d'autres appareils, échangent et analysent également les données entre plusieurs systèmes sur différents réseaux. Ainsi, les systèmes IoT doivent gérer un degré élevé d'interopérabilité [16].

- **Limitation des ressources**

Les objets IoT étant caractérisés par leur hétérogénéité, ils sont différents en termes de capacités énergétiques, de stockage et de calcul. En fait, la majorité des objets IoT sont connus pour leurs contraintes de ressources limitées. Par conséquent, le développement des plates-formes IoT doit optimiser et minimiser autant que possible l'utilisation de l'énergie, du stockage et du calcul des objets [17].

- **Sécurité**

La sécurité est un enjeu important dans l'IoT en raison du nombre croissant d'appareils intelligents qui nous entourent en plus de l'hétérogénéité de ces appareils et leurs ressources limitées. Aussi, la garantie de sécurité doit être prise en compte lors de la conception des protocoles et des applications IoT.

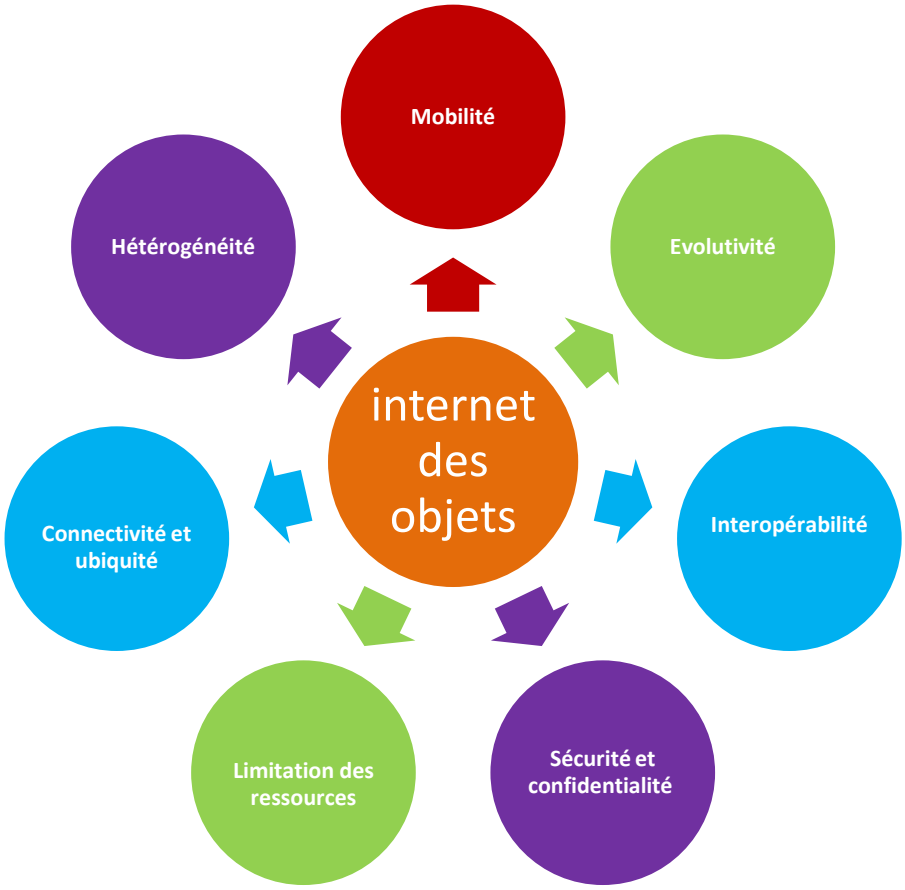


Figure I.4: Caractéristiques de l'IoT.

I.6 Les réseaux à faible consommation et avec perte

Diverses technologies sont impliquées dans la mise en œuvre de l'idée de l'IoT. Nous nous concentrerons sur les réseaux à faible consommation et avec perte, connus en anglais sous le nom « Low Power and Lossy Networks (LLN) », car ils représentent un composant nécessaire pour l'IoT.

Les LLNs sont des réseaux de capteurs dans lesquels les routeurs et les nœuds sont fortement limités en ressources en termes de capacité de traitement, de batterie et de taille de

mémoire, et leurs liens d'interconnexion sont instables avec des taux de perte élevés, de faibles débits de données et de faibles taux de livraison de paquets [8]. Même si plusieurs classes d'appareils peuvent être utilisées dans ces réseaux, les ressources informatiques disponibles sont assez minimes, comparées aux appareils informatiques standards utilisés dans la plupart des applications d'aujourd'hui. Cela signifie que les protocoles pour l'IoT doivent fonctionner dans les limites de ressources impliquées par ces dispositifs [18].

Depuis l'émergence de l'IoT, le routage dans les LLNs est l'un des principaux défis. Les limitations et les contraintes des LLNs doivent être prises en compte pour la conception et la mise en œuvre de tout protocole de routage [8]. Les caractéristiques des nœuds dans les LLNs pourraient constituer un défi ouvert qui consiste à trouver un protocole de routage optimal dans les réseaux de capteurs sans fil. Un protocole de routage appelé RPL a été spécialement conçu par l'IETF pour répondre aux contraintes spécifiques qu'impose ce type de réseaux. La FigureI.5 illustre un exemple de scénario LLN multi-sauts basés sur le protocole RPL.

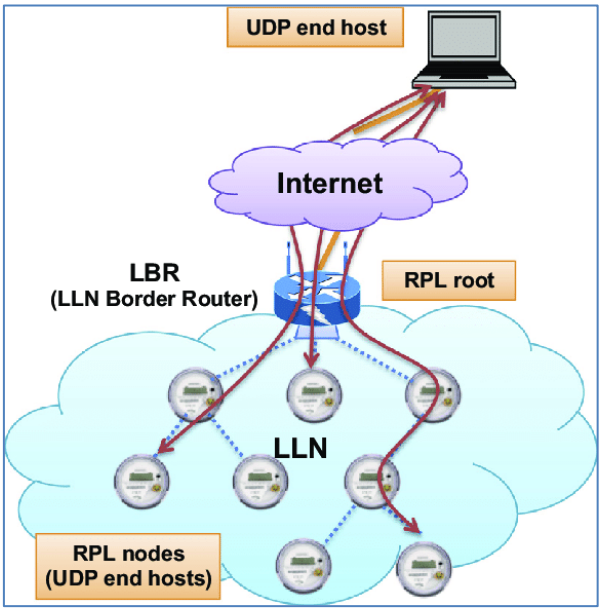


Figure I.5: Illustration d'un scénario LLN multi-sauts IoT [19].

I.7 Les protocoles de l'IoT

La communication est le seul moyen pour les objets intelligents de l'IoT d'échanger des données entre eux. Par conséquent, l'IEEE et l'IETF travaillent continuellement à la formation et au développement de protocoles standardisés. La FigureI.6 représente les protocoles utilisés dans l'IoT et Internet suivant le modèle TCP/IP.

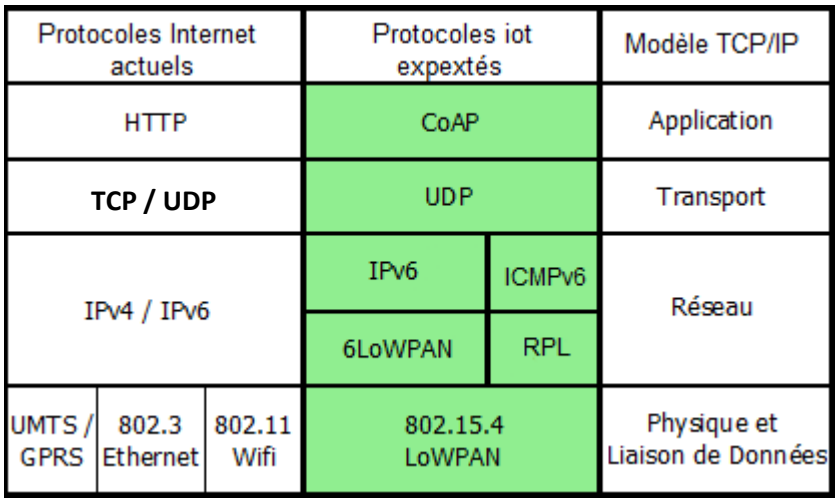


Figure I.6: Suite de protocoles pour l'IoT comparée à la suite TCP/IP classique [17].

Les sections suivantes donnent un aperçu de certains des protocoles standardisés les plus courants dans le domaine de l'IoT. Ces protocoles se caractérisent tous par des trames courtes, avec des entêtes courts, mais aussi des espaces réservés aux données fortement réduites.

I.7.1 L'IEEE 802.15.4

Le standard IEEE 802.15.4 est une norme de communication sans fil pour les LLNs. Il spécifie une sous-couche pour le contrôle d'accès au support (MAC) et une sous-couche physique (PHY) pour ces réseaux. Il prend en charge différentes topologies de réseau et différents modes d'accès aux canaux [20]. Le standard a été défini par l'IEEE dans le groupe de travail 802.15 (PAN). Il vise les faibles bandes passantes (< 250 kbps) et les faibles puissances (< 1mW). Les équipements ont une portée de quelques dizaines de mètres et peuvent opérer sur batterie pendant plusieurs années en fonction des applications. L'IEEE 802.15.4 est utilisé comme couche de transmission (niveau 1 et 2 de l'architecture OSI (Open Systems Interconnection)) par de nombreux standards tels que WirelessHART, ISA 100a, ZigBee, RPL… [21].

I.7.2 6LowPAN

En raison de la taille limitée des paquets, de la faible capacité d'alimentation et d'autres contraintes de l'IoT, la communauté des chercheurs (groupe de travail 6LoWPAN IETF) a développé une version compressée d'IPv6 nommée 6LoWPAN. Il s'agit d'une couche d'adaptation IPv6 (Internet Protocol version 6) fonctionnant au-dessus de la norme IEEE 802.15.4 (couche MAC/PHY). Le principe est de reporter la charge d'adaptation (compression d'en-tête, fragmentation, adressage, etc.) à la frontière du réseau, tout en

conservant un fonctionnement normal de ce dernier [21]. Il se base principalement sur deux mécanismes afin de réduire la taille des datagrammes IPv6 à savoir la fragmentation et la compression des entêtes afin de permettre aux paquets IPv6 d'être envoyés ou reçus via les réseaux LLN. Ce mécanisme est exécuté au niveau du 6LoWPAN Border Router (6LBR) qui représente une passerelle entre les nœuds 6LoWPAN et Internet. Le 6LBR est également responsable de la gestion du trafic entre les interfaces IPv6 et IEEE 802.15.4, c'est une solution pour compresser les grandes trames de paquets pour les insérer dans les tailles de trame IEEE 802.15.4 [22] [23]. La FigureI.7 montre qu'avec l'adoption du protocole IPv6 par les maisons intelligentes actuelles, ces appareils embarqués pourraient être adressables de manière unique et peuvent être connectés directement à Internet via les routeurs frontaliers 6LoWPAN.

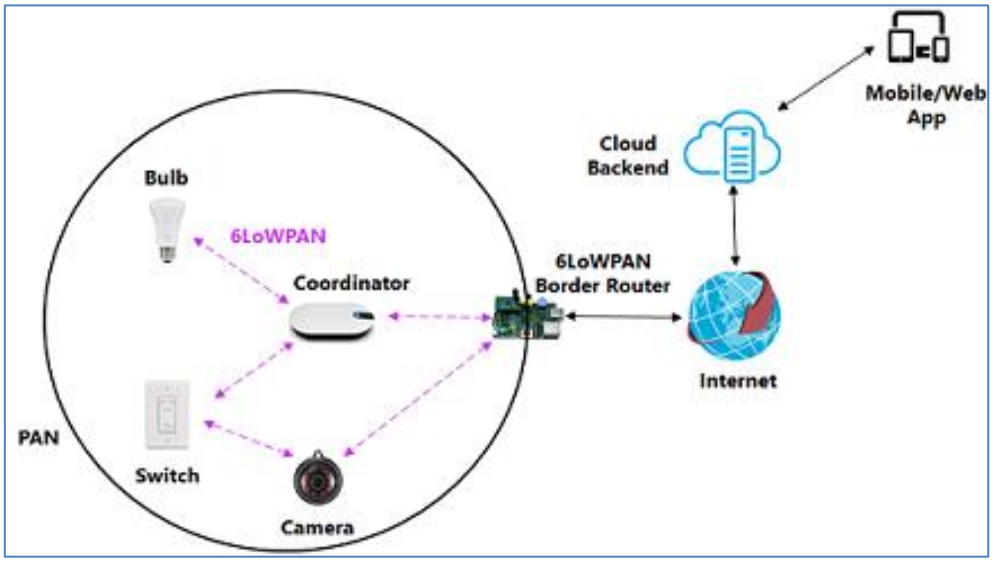


Figure I.7: Architecture de réseau d'une maison intelligente basée sur 6LBR et 6LoWPAN.

I.7.3 Le protocole de routage RPL

Le RPL est le protocole de routage normalisé par l'IETF pour le routage des paquets IPv6 dans les LLNs. Il construit un graphe acyclique dirigé avec destination orientée (DODAG) de nœuds au moyen de messages de contrôle. Parmi l'ensemble des messages de contrôle, l'objet d'information du DODAG (DIO) est celui qui porte les informations essentielles telles que la position relative du nœud dans le DODAG (c.-à-d. Rang), version et identification du DODAG [24]. Comme RPL est au cœur de notre travail, plus de détails seront présentés dans le chapitre suivant.

I.7.4 Le protocole d'application contraint (CoAP)

L'IETF Constrained Application Protocol (CoAP) est un protocole de la couche application adapté aux appareils à ressources limitées et aux applications M2M (Machine-to-Machine). Il permet la communication sur Internet entre les objets IoT qui prennent en charge UDP (User Datagram Protocol) et 6lowPAN, ce qui permet d'obtenir une faible surcharge et de prendre en charge la multidiffusion [25]. CoAP est un protocole RESTful léger qui est basé sur UDP et hérite du même paradigme client/serveur adopté en HTTP (HyperText Tranfert Protocol) [26] comme illustré dans la FigureI.8. En effet, il implémente un ensemble de techniques pour compresser les métadonnées du protocole de la couche application sans compromettre l'interopérabilité des applications.

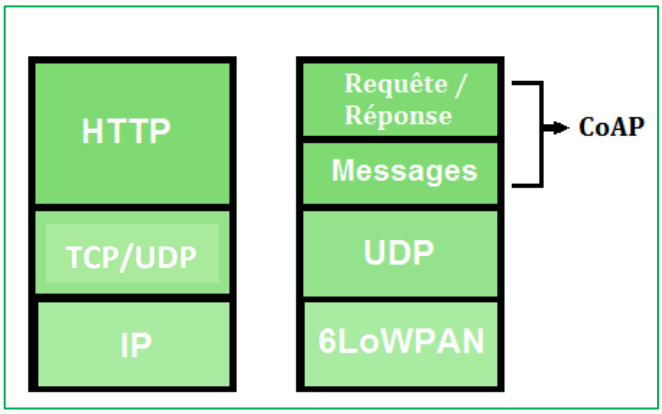


Figure I.8: Piles des protocoles HTTP et CoAP [27].

I.8 La sécurité dans l'IoT

Étant donné que tout le monde peut accéder à certains appareils IoT de n'importe où sans l'autorisation de l'utilisateur, la sécurité des appareils et des communications IoT est devenue une question brûlante. Un large éventail de systèmes de sécurité doit être mis en œuvre pour protéger ces derniers. Cependant, la structure physique des appareils IoT limite sa fonctionnalité de calcul, ce qui limite la mise en œuvre d'un protocole de sécurité complexe [16].

Dans la littérature, il existe différentes architectures pour l'IoT, comme celles composées de trois, quatre ou cinq couches. Dans ce qui suit, nous présenterons les problèmes de sécurité suivant l'architecture trois couches pour raison de simplicité.

I.8.1 Couche perception

Le rôle de la couche perception est de rassembler les informations du monde réel (l'environnement) et de les traiter si nécessaire. Les informations traitées sont ensuite transmises à la couche réseau pour un traitement ultérieur. En raison de la nature restreinte de ces appareils et du fait que les objets ne sont pas inviolables, et en raison du manque de sécurité sur ces appareils, un intrus peut facilement y accéder afin d'endommager ou de reprogrammer des actions illégales sur eux. [17]. Différentes attaques peuvent être déclenchées comme, les suivantes :

- **Attaques par canal :** qui sont une famille d'attaques consistant à extraire les informations relevant de la cryptographie comme les clés de chiffrement, obtenues lors de la mise en œuvre du système. Les attaques les plus célèbres sont les attaques de synchronisation, attaques d'analyse de puissance, attaques d'analyse électromagnétique, attaques par induction de pannes, attaques de canal latéral optique, attaque d'analyse de trafic, attaques acoustiques et attaques d'imagerie thermique [28].
- **Attaque par force brute :** Une attaque par force brute, également connue sous le nom de recherche exhaustive, est une méthode de craquage populaire qui consiste à deviner les combinaisons possibles d'un mot de passe pour obtenir un accès non autorisé à un système et décoder les données sensibles.
- **Attaques de l'homme du milieu (MITM) :** L'attaque de l'homme du milieu (HDM) ou man-in-the-middle attack (MITM), parfois appelée attaque de l'intercepteur, est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elle a été compromis. Afin d'acquérir ou de modifier des données importantes circulant entre eux.
- **Autres :** La falsification des nœuds et les attaques physiques, les attaques par collision, attaques abusives, attaques de mascarade, attaques d'écoute clandestine, attaques de destruction de nœuds, les attaques par brouillage, les attaques par épuisement de la batterie, les attaques par rejet, les attaques par reniflage du trafic et attaques d'authentification RFID, etc.

I.8.2 Couche réseau

La couche réseau agit comme un canal de communication pour transférer les données, collectées dans la couche de détection, vers d'autres appareils connectés. Dans les appareils IoT, la couche réseau est implémentée en utilisant les réseaux de communication filaire et sans fil existants tels que Wifi, Bluetooth, Zigbee, LoRa, Z-Wave, LTE et GPRS. De plus, la couche de communication est considérée comme l'épine dorsale des systèmes IoT. Et le canal principal entre la couche application et les différentes activités d'exploitation du système IoT.

I.8.2.1 Attaques sur la couche d'adaptation

Dans [29] les chercheurs ont présenté une enquête sur les attaques liées à la couche d'adaptation 6LoWPAN. Certaines des attaques sont les suivantes :

- **Attaques par fragmentation :** Dans ces attaques, l'attaquant peut modifier ou reconstruire les champs de fragmentation des paquets en mettre ses propres fragments dans la chaîne de fragmentation. En raison du manque de mécanisme d'authentification côté récepteur pour détecter les fragments falsifié ou double de la chaîne de fragmentation. Ainsi, un attaquant peut prétendre être un nœud légitime et exploiter cette faiblesse pour se lancer dans d'autres attaques telles qu'une attaque par déni de service (DoS) [29], [30]. Ces attaques peuvent causer des dommages critiques à un nœud, par exemple, un débordement de tampon de réassemblage, l'épuisement des ressources.

- **Attaques d'authentification :** Malheureusement, il n'existe aucun mécanisme d'authentification pour vérifier les nœuds avant qu'ils ne rejoignent le réseau, ce qui permet aux nœuds malveillants de rejoindre facilement le réseau et de générer d'autres attaques. Ces attaques entraînent des dommages importants aux systèmes IoT.

- **Attaques de confidentialité :** La confidentialité, consistant à assurer que seuls les nœuds légaux et autorisés peuvent accéder, surveiller et contrôler les données du réseau. Assurer la confidentialité dans 6LoWPAN atténue diverses attaques telles que l'écoute clandestine, l'homme au milieu, les attaques d'usurpation d'identité, etc. la confidentialité peut être fournie par IPsec (Internet Protocol Security).

- **Attaques Internet :** Les réseaux 6LoWPAN sont devenus vulnérables à plusieurs attaques d'Internet. Parce que les appareils de ces réseaux sont caractérisés par une mémoire limitée, une puissance de calcul et une sécurité très limitée. En plus de sa connexion directe à Internet non sécurisé.

Concernant les attaques contre le protocole de routage RPL, nous les aborderons dans le prochain chapitre.

I.8.3 Couche d'application

Cette couche fournit des services en fonction des besoins de l'utilisateur. En outre, elle implémente et présente les résultats de la couche de traitement des données pour réaliser des applications disparates des appareils IoT, tels que le transport intelligent, la maison intelligente, les soins personnels, les soins de santé, etc.

La couche d'application sur Internet est généralement basée sur le protocole HTTP. Cependant, HTTP ne convient pas dans un environnement à ressources limitées, car il est extrêmement lourd et entraîne donc une surcharge d'analyse importante. Ainsi, de nombreux protocoles alternatifs ont été développés pour les environnements IoT, tels que le Protocol (CoAP). Pour la sécurité, le protocole CoAP utilise Datagram Transport Layer Security (DTLS) sur UDP, qui fournit une couche de protection supplémentaire. Bien que des mécanismes de sécurité aient été mis en place dans CoAP, il souffre d'attaques courantes, notamment attaques d'analyse, attaques de mise en cache, les attaques par amplification et attaques d'usurpation d'identité [31], les attaques par injection de fausses données, attaque DoS basé sur le chemin, attaque de reprogrammation et l'attaque de capteur accablant [32].

I.9 Contre-mesure

Les problèmes de sécurité telle que ceux présentés dans la section précédente peuvent menacer l'existence de l'IoT si des mesures de protection ne sont pas prises. Dans ce contexte, plusieurs travaux se concentrent sur la fourniture de mécanismes pour atténuer ces risques. Par exemple, pour les attaques par liaison, la perte de confidentialité et la manipulation de données, des scientifiques ont proposé l'utilisation de petites trames comme ligne de défense contre l'injustice [33]. Concernant les attaques MITM, les efforts de recherche peuvent se concentrer sur la mise en œuvre de protocoles de gestion de clés appropriés pour les communications IoT. Une autre solution peut être l'utilisation de pare-feu et de systèmes de détection d'intrusion (IDS) [32].

Pour éviter les attaques de la couche application et les attaques Internet, un pare-feu peut être installé sur le 6LBR pour contrôler les paquets malveillants provenant d'Internet.

Pour atténuer l'attaque de duplication de fragments et l'attaque de réservation de tampon au niveau de la conception contre le mécanisme de fragmentation 6LoWPAN. Les auteurs de [30] ont proposé un schéma de chaînage de contenu et l'approche de tampon divisé avec une stratégie de rejet de paquets sur mesure. De plus, dans [34], un nouveau protocole de sécurité appelé « 6LowPAN Sec » a été proposé, fonctionnant au niveau de la couche d'adaptation.

D'un autre côté, le chiffrement de la couche de liaison empêcherait les attaques externes telles que l'écoute clandestine [34], et certaines solutions sont fournies dans [35], [36], [37], [38], [39]. Comme mentionné dans les travaux [40] et [41], les protocoles de sécurité standard sont utilisés pour fournir un mécanisme pour protéger le réseau contre les nœuds malveillants et de nombreux types d'attaques (comme les attaques DoS, les attaques Sybil et CloneID, etc.). En plus de cela, il s'avère que les clés secrètes (clé symétrique et clé asymétrique) sont utilisées par les protocoles choisis pour offrir plus de sécurité et de robustesse.

Pour éviter les attaques de confidentialité, IPsec définit un ensemble de protocoles pour sécuriser la communication sécurisée de bout en bout. Il permet d'authentifier et crypter également les paquets de données échangés, en utilisant les protocoles de sécurité Authentication Header (AH) [42] et Encapsulating Security Payload (ESP) [43], les algorithmes d'authentification et de chiffrement.

De plus, ils existent des travaux sur la gestion sécurisée des clés pour l'IoTtels que ceux dans [44] et [45]. Cependant, le cryptage n'est pas suffisant pour résoudre les exigences de sécurité où il est considéré comme la première ligne de défense pour résoudre la confidentialité, l'intégrité et l'authentification. Par conséquent, il est très important de fournir des solutions de sécurité supplémentaires pour coopérer avec les mécanismes de cryptages, tels que les systèmes de détection d'intrusion (IDS) requis comme deuxième ligne de défense, qui peuvent surveiller et détecter les nœuds malveillants dès la phase initiale.

I.9.1 IDS pour les réseaux IoT

Un IDS est utilisé pour détecter les attaques internes et externes. En effet, il est principalement utilisé pour contrer les attaques contre le réseau [17]. Il surveille et analyse les informations, le trafic et le comportement des nœuds du réseau et tente de détecter les intrus tentant de perturber le réseau. Il existe principalement trois composants de l'IDS : surveillance, analyse et détection, alarme [46]. Les IDS peuvent être classés en fonction de leur emplacement **(**Centralisé, Distribué, Hybride**)** ou de la méthode utilisée pour la détection (Basé sur la signature, Basé sur les anomalies, Basé sur les spécifications, Basé sur l'hybride, Basé sur les événements).

I.10 Conclusion

Récemment, l'Internet des objets est devenu l'une des technologies les plus importantes du 21e siècle, où il s'est développé rapidement dans le contexte des réseaux et des services.

Dans ce chapitre, nous avons présenté les différents concepts généraux liés à l'IoT, à savoir la définition et l'historique de l'IoT, ainsi que ses domaines d'applications, les caractéristiques d'IoT et les différents protocoles tels que 6LOWPAN, CoAP, RPL, etc. De plus, nous avons présenté les différentes attaques selon chaque couche de la pile IoT, et nous avons présenté quelques mécanismes de sécurité existants pour protéger l'IoT. Dans le chapitre suivant, nous décrivons le protocole de routage RPL.

Chapitre II : Le Protocole de Routage RPL

Table des matières

II.1 Introduction

Le protocole de routage pour les réseaux à faible consommation avec perte (RPL) a été conçu par le groupe de travail IETF RoLL (Routing Protocol for Low Power and Lossy Networks), et est considéré comme le protocole de routage de l'Internet des objets. Il est développé sur la base d'une sélection de fonctions objectives (FO) qui sont utilisées pour évaluer les performances pour répondre aux diverses exigences prédéfinies de la domotique, du contrôle industriel et de l'environnement urbain [47].

Dans ce chapitre, nous allons présenter le protocole RPL et fournir quelques informations de base à son sujet. De plus, nous discuterons le fonctionnement de ce protocole en abordant le processus de construction du DODAG et les différents messages de contrôle qui contribuent à sa construction. Enfin, nous présenterons les différentes attaques qui ciblent le protocole RPL, et les mesures de sécurité nécessaires pour le protéger.

II.2 Le protocole de routage RPL

Le protocole de routage pour réseaux à faible consommation et à perte RPL (Routing Protocol for Low power and Lossy networks-LLNS) est le protocole de routage standardisé pour l'IoT basé sur IP, qui a été développé par le groupe de travail IETF ROLL pour répondre aux limites des réseaux LLN. Il est conçu pour prend re en charge le fonctionnement de plusieurs protocoles de couche liaison, y compris la couche physique IEEE 802.15.4 et la couche MAC [48].

Une caractéristique clé du RPL est qu'il représente une solution de routage spécifique pour les réseaux à faible puissance et avec perte [49] [50], qui sont des réseaux avec des ressources très limitées en termes d'énergie, de calcul et de bande passante. Le RPL a été conçu et capable d'être extrêmement adaptable aux conditions du réseau et de prendre en charge des routes alternatives lorsque les routes par défaut ne sont accessibles à aucun moment [7]. Il construit une topologie logique appelée DODAG (Destination Oriented

Directed Acyclique Graph). En construisant ce DODAG, RPL permet de transporter des paquets IPv6 à leur destination à travers une ou plusieurs routes.

Nous détaillons ci-dessous le fonctionnement et les principales caractéristiques du protocole RPL.

II.3 Identifiant RPL

Un réseau RPL est identifié par trois champs : RPLInstanceID, DODAGID, et DODAGVersionNumber.

- **RPLInstanceID :** Un RPLInstanceID est un identifiant unique dans un réseau. Les DODAG avec le même RPLInstanceID partagent la même Fonction Objectif (FO) [50].
- **DODAGID :** DODAGID est une adresse IPv6 de 128 bits définis par un root DODAG [51]. Une instance RPL peut avoir plusieurs DODAG, chacun ayant un DODAGID unique.
- **DODAGVersionNumber :** Un DODAG est parfois reconstitué à partir du root DODAG, en incrémentant le DODAGVersionNumber. La combinaison de RPLInstanceID, DODAGID et DODAGVersionNumber identifie de manière unique une version DODAG [50].
- **Rank :** Pour éviter les boucles dans le routage, RPL introduit un terme appelé rangs (rank en anglais). Le rang d'un nœud est une représentation scalaire de l'emplacement de ce nœud dans une version DODAG. Le rang est utilisé pour éviter et détecter les boucles et, en tant que tel, doit démontrer certaines propriétés. Le calcul exact du rang est laissé à la FO [50]. En particulier, une valeur de rang doit diminuer de façon monotone à mesure que les gradients s'écoulent vers le root DODAG. Le rang est également utilisé pour établir la position relative du nœud par rapport aux autres nœuds.

II.4 DAG et DODAG

Un DAG (graphe orienté acyclique) est un graphe orienté sans cycles. Il décrit les liens orientés entre les nœuds, se terminant à un ou plusieurs nœuds root [50]. Chaque DAG possède au moins un nœud root. Ce graphe est composé d'un ou de plusieurs DAG orientés vers la destination (DODAG) avec un root par DODAG. La Figure II.1 montre la différence entre DAG et DODAG.

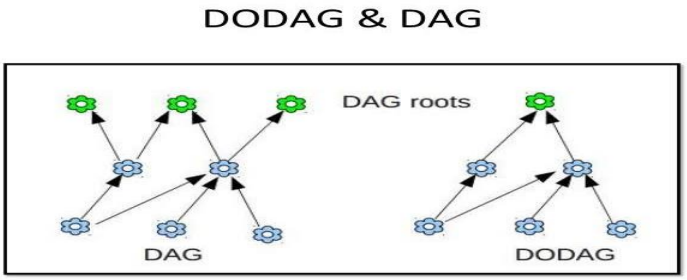


Figure II.1: Les graphes DAG et DODAG [52].

II.5 Types de nœuds dans RPL

RPL défini trois types de nœuds :

II.5.1 Low Power and Lossy Border Routers (LBR)

Il s'agit du root d'un DODAG qui représente un point de collecte dans le réseau et a la capacité de construire un DAG [51]. Le LBR agit comme un dispositif de routage et peut éventuellement héberger d'autres fonctions telles que le collecteur de données ou l'agrégateur et agit comme une passerelle (ou routeur de périphérie) entre Internet et le LLN [49].

II.5.2 Routeur

Il s'agit d'un appareil qui peut transférer et générer du trafic. Un tel routeur n'a pas la possibilité de créer un nouveau DAG, mais de s'associer à un existant [51].

II.5.3 Hôte

Il s'agit d'un périphérique final capable de générer du trafic de données, mais qui n'est pas en mesure de transférer le trafic [51].

II.6 Messages de contrôle RPL

Pour mettre en place et maintenir la topologie de routage, quatre messages ICMPv6 ont été définis pour le protocole RPL, comme illustré dans la Figure II.2 :

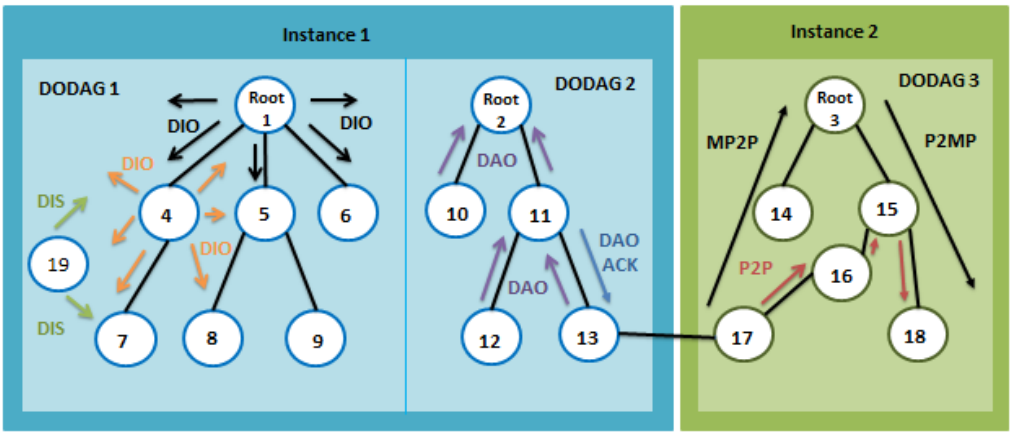


Figure II.2: Exemple de réseau RPL composé de deux instances et trois DODAG [18]

II.6.1 DIO (DODAG Information Object)

Ce message est utilisé pour la création des routes ascendantes dans lequel d'autres nœuds (nœuds non root/récepteur) peuvent découvrir le nœud root (instance RPL) [7], obtenir leurs paramètres de configuration et le rejoindre en tant que nœud parent. En particulier, un message DIO est composé de cinq champs principaux correspondant à l'ID d'instance RPL qui identifie de façon unique l'instance RPL dans le réseau, l'ID DODAG qui caractérise le root (très souvent l'une de ses adresses IPv6), le numéro de version du DODAG permettant de savoir si les informations reçues sont « fraîches » ou obsolètes, la valeur de rang d'un nœud émetteur dans le DODAG qui renseigne sur sa position dans l'arbre construit sur le réseau. Le champ mode de fonctionnement (MOP) permet le maintien des routes descendantes et les communications multicast. La dissémination des DIO se fait par défaut en multidiffusion, mais en cas de sollicitation spécifique d'un nœud, la monodiffusion est utilisée [53].

II.6.2 DIS (DODAG Information Sollicitation)

Ce message est utilisé par un nœud qui souhaite rejoindre la topologie (envoi en multidiffusion) ou réclame des informations de configuration plus récentes (envoi en monodiffusion). Dans tous les cas, un nœud recevant un DIS répond à l'initiateur par un paquet DIO [53].

II.6.3 DAO (DODAG Advertisement Object)

Ce message est utilisé pour construire des routes descendantes le long du graphe DODAG, où il est envoyé par chaque nœud, autre que le root DODAG. Il permet de diffuser des informations de destination le long de DODAG et permet à un nœud de se joindre en tant qu'enfant au root DODAG ou au parent DAO [7].

II.6.4 DAO-Ack (DAO-Acknowledgment)

C'est un message d'accusé de réception unicast envoyé par le destinataire DAO en tant que réponse au message DAO reçu pour acquitter ce dernier. En cas de non-réception du DAO-Ack par la source, celle-ci peut réémettre le DAO initial [53]. Il contient des informations sur RPLInstanceID, DAOSequence et Status.

Tous ces différents messages de contrôle jouent un rôle important dans la construction et la maintenance des DODAG.

II.7 Construction du DODAG

Comme illustré dans la Figure II.3, pour construire un DODAG, le root commence à envoyer des paquets DIO contenant des informations sur le DODAG. Lorsqu'un nœud reçoit un message DIO d'un root, il traite ce message, puis prend une décision (joindre la structure ou pas) basée sur certaines règles (selon la OF, les caractéristiques du DAG et le coût du chemin annoncé).

Pour que le nœud rejoigne le DODAG, il ajoute l'expéditeur du message DIO à sa liste de voisins candidats (contenant des nœuds de rang inférieur) et forme un ensemble de parents. Ensuite, il choisit un parent préféré optimisant l'OF (parent ayant le rang le plus bas), qui devient la passerelle par défaut à utiliser lorsque des données doivent être envoyées vers le root DODAG. Le nœud choisit aussi parmi son ensemble de voisins des parents de substitution (appelé Feasible Successors). Dans le cas, où le parent n'est plus joignable, un des parents de substitution le remplacera dans le DODAG [54]. De plus, le nœud détermine sa propre valeur de rang dans le graphe. Enfin, le nœud peut alors commencer à envoyer ses propres messages DIO en ajoutant sa propre métrique à celle annoncée par le parent à ses voisins. Tous les nœuds voisins répètent le processus, jusqu' à ce que le DODAG soit construit.

Ainsi, le DODAG se construit étape par étape du root jusqu'aux feuilles. Il existe une route entre chaque nœud du DODAG et le root. Un nœud peut ainsi envoyer des paquets de données au root en envoyant le paquet à son parent. Les paquets sont transmis par conséquent saut par saut jusqu'au root du DODAG [54]. Une fois la construction terminée, les messages DIO sont envoyés périodiquement en respectant la minuterie Trickle [55] qui optimise la fréquence de transmission des messages de contrôle.

Pour identifier de manière unique un nœud dans un DODAG, RPL utilise le triplet [ID d'instance RPL, ID DODAG, Rang], où le rang est déterminé en utilisant une fonction objective. Le rang décrit la position relative d'un nœud par rapport au nœud récepteur [56]. Il existe des routes ascendantes dirigées des nœuds DODAG vers le root DODAG et des routes descendantes dirigées du root DODAG vers les nœuds DODAG [57]. Le root envoie périodiquement dans son voisinage des messages DIO. Les nœuds qui reçoivent ces informations peuvent rejoindre le réseau en choisissant leur prochain saut (parent RPL) vers le root. Dès qu'il fait partie intégrante de la topologie RPL, le nœud commence par envoyer ses propres DIO [53]. Dans le RPL, le numéro de version (VN) est utilisé comme indicateur de

réparation globale. Selon la spécification RPL, seul le root DODAG devrait le changer et ainsi lancer une réparation globale [24]. Chaque fois que le mécanisme de réparation global est appelé, le numéro de version du DODAG est incrémenté. Si un nœud détient une ancienne copie du numéro de version, cela signifie que ses entrées de table de routage sont obsolètes. L'état actuel d'un DODAG est identifié de manière unique par le triplet [ID d'instance RPL, ID DODAG, Numéro de version DODAG]. Par conséquent, le numéro de version est un paramètre important pour maintenir une topologie stable et sans boucle des LLN dans l'environnement IoT [56].

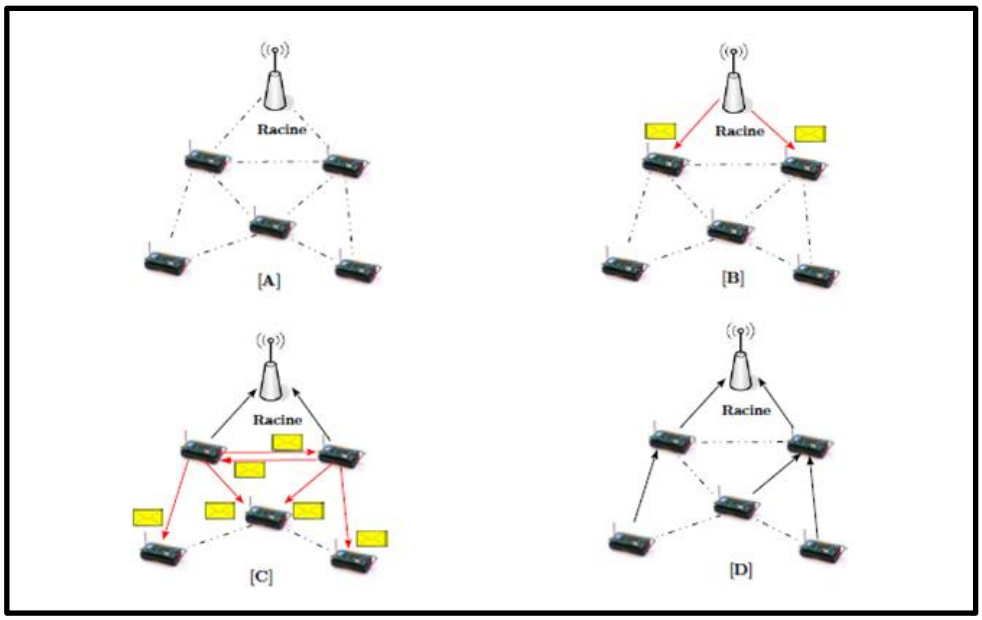


Figure II.3: Construction du DODAG [53].

II.8 Mécanismes de réparation du DODAG

Les mécanismes de réparation sont très importants pour un protocole de routage pour mettre à jour les routes de manière dynamique et adapter la topologie du réseau aux pannes potentielles. RPL prend en charge deux mécanismes de réparation intégraux ; en particulier réparation locale et réparation globale [51]. Lorsqu'un nœud détecte une défaillance du réseau (par exemple, une liaison entre deux nœuds échoue), il déclenche une réparation locale. Cette réparation consiste à rechercher en urgence un chemin de sauvegarde sans tenter de réparer tout le DODAG. Cependant, ce chemin de récupération alternatif peut ne pas être le meilleur chemin par rapport à l'OF définie. Le deuxième mécanisme est la réparation globale, où le root DODAG peut déclencher cette réparation si les réparations locales ne sont pas efficaces pour la récupération du réseau en raison de plusieurs incohérences. Il incite à la reconstruction fondamentale de toute la topologie du réseau, mais au prix d'un trafic de contrôle

supplémentaire dans le réseau [58]. Les nœuds de la nouvelle version DODAG peuvent choisir une nouvelle position dont le rang ne dépend pas de la précédente. La réparation globale est considérée comme une réoptimisation complète des routes.

II.9 Le Trickle Timer

RPL utilise le Trickle Timer [55] ; un algorithme qui gouverne l'envoi de messages DIO pour réduire la surcharge des messages de contrôle et préserver l'énergie des nœuds en ne transmettant les mises à jour que lorsque des incohérences sont détectées dans le réseau.

Initialement, la fréquence de transmission des messages DIO est réglée sur un intervalle minimal. Ce délai double à chaque expiration du minuteur de l'expéditeur, jusqu'à une valeur maximale. Quand la topologie est stable (i.e. consistant) lorsque tous les nœuds ont obtenu les mêmes paramètres de configuration et que ceux-ci sont cohérents (i.e. non contradictoires), l'intervalle de la minuterie de maintien augmente et le débit de transmission sera ralenti (c'est-à-dire moins de messages de contrôle). Sinon, s'il y a des incohérences (par exemple des messages DIO altérés, etc.) ou qu'une nouvelle information est disponible, qui implique des changements dans la topologie, la minuterie Trickle sera réinitialisée à une valeur inférieure et le taux de transmission sera accéléré [59].

II.10 Fonction objective

La fonction objective (OF) définit comment les nœuds RPL sélectionnent et optimisent les routes au sein d'une instance RPL en fonction d'un ensemble de métriques et/ou contraintes. L'OF est identifiée par un Objectif Code Point (OCP) dans l'option de configuration DIO [50]. Une OF définit comment les nœuds traduisent les métriques de routage et contraintes en une valeur appelée rank. Elle est également responsable de déterminer comment les nœuds sélectionnent les parents. L'IETF a défini deux fonctions objectives dans le RPL.

II.10.1 OF0 (Objective Function 0)

OF0 est la fonction objective par défaut définie dans RPL. Avec l'OF0, un nœud calcule son rang en ajoutant une valeur au rang de son parent. La valeur ajoutée au rang de son parent représente la métrique de routage du parent ou du lien vers le parent [56]. De plus, elle implémente le nombre de sauts comme métrique.

II.10.2 MRHOF (Minimal Rank with Hysteresis Objective Function)

La métrique utilisée par MRHOF est déterminée dans le conteneur de métrique du DIO. Le plus souvent, c'est le nombre de retransmissions attendues (ETX) à utiliser avec une hystérésis pour éviter les différences minimes de rang. Cette métrique permet à RPL de trouver les chemins stables à partir des nœuds vers le root. En l'absence d'une métrique dans le conteneur métrique DIO, MRHOF utilise par défaut ETX [60].

II.11 Les modes d'opération du protocole RPL

Le protocole RPL offre deux modes de fonctionnement, le fonctionnement en «mode avec stockage» et celui en «mode sans stockage».

II.11.1 Mode avec stockage

Dans ce mode, les nœuds intermédiaires sont en mesure de garder dans des tables de routages des informations de routage puis de rediriger les données reçues dans les messages DAO vers la bonne destination en consultant les informations du routage. En outre, lorsqu'un nœud génère un nouveau DAO, il devrait le diffuser à chacun de ses parents DAO. Il ne doit pas diffuser le message DAO à des nœuds qui ne sont pas des parents DAO et lorsqu'un nœud supprime un nœud de son ensemble parent DAO, il devrait envoyer un message DAO sans chemin à ce parent DAO supprimé pour invalider la route existante [50].

II.11.2 Mode sans stockage

En mode sans stockage, seul le root est en mesure de stocker des informations de routage. Les autres nœuds du réseau ne conservent que les adresses parentes directes. Puisque les nœuds (excepté le root) n'enregistrent pas les routes descendantes, un paquet envoyé remonte tout le chemin jusqu' au root. Ensuite, le root construit le chemin jusqu'à la destination et envoie le paquet à la destination sur le chemin construit [56].

Dans ce mode, les messages DAO sont monodiffusés vers le root DODAG. En cas de besoin de router des données vers une destination quelconque, les nœuds transmettent des données dans des messages DAO jusqu'à ce qu'ils atteignent le root, où chaque nœud qui reçoit un message DAO ajoute son propre DAO et le transmet. Lorsqu'il atteint le root, le root utilise les données reçues dans les messages DAO pour reconstruire le chemin.

II.12 Les paradigmes de communication

RPL prend en charge trois paradigmes de communication (MP2P), (P2MP) et (P2P) comme le montre le troisième DODAG dans la Figure II.2.

II.12.1 Opération multipoint à point

RPL prend en charge le trafic multipoint à point qui concerne le trafic de collecte de données transmis dans le sens de la route ascendante à partir de plusieurs nœuds vers le root DODAG. Le trafic de collecte de données est appelé trafic unicast entrant [66]. Le multipoint à point (MP2P) est un flux de trafic dominant dans de nombreuses applications LLN [50].

RPL prend en charge le trafic MP2P de sorte que les destinations peuvent être atteintes via les roots DODAG. Une fois que le DODAG est construit, il est utilisé pour construire les routes ascendantes des routeurs vers le root. Les routes par défaut des nœuds aux root sont établies via la chaîne des parents préférés [51].

II.12.2 Opération point à multipoint

RPL définit également l'opération point à multipoint, qui représente le trafic transmis dans la direction descendante du root vers plusieurs nœuds [51]. Cette configuration est communément appelée trafic unicast sortant, et il est essentiel pour certaines applications LLN comme Home Automation [32] et Industrial Automation [17].

RPL prend en charge le trafic P2MP en utilisant un mécanisme d'annonce de destination qui fournit des routes descendantes vers des destinations (préfixes, adresses ou groupes de multidiffusion) et loin des roots. Les annonces de destination peuvent mettre à jour les tables de routage à mesure que la topologie DODAG sous-jacente change. [51]

II.12.3 Opération point à point

RPL fournit des mécanismes de routage point à point (P2P) entre deux nœuds quelconques du DODAG. Pour qu'un réseau RPL prenne en charge le trafic P2P, un root doit être capable d'acheminer des paquets vers une destination. Les nœuds du réseau peuvent également avoir des tables de routage vers des destinations. Un paquet circule vers un root jusqu'à ce qu'il atteigne un ancêtre qui a une route connue vers la destination. Cet ancêtre commun peut être le root DODAG. Dans d'autres cas, il peut s'agir d'un nœud plus proche à la fois de la source et de la destination [50].

II.13 Attaques sur le protocole RPL

Le protocole RPL est exposé à une grande variété d'attaques de sécurité. Ces attaques sont difficiles à détecter et à atténuer en raison de la nature vulnérable des nœuds et du réseau sans fil, de la nature facilement falsifiable des nœuds, de la mobilité des nœuds et des contraintes de ressources. Dans la littérature, il existe différentes classifications des attaques contre RPL. Les auteurs dans [34], [61], et [62] ont classé les attaques RPL en trois catégories: les attaques sur les ressources, les attaques sur la topologie et les attaques sur le trafic. Medjek et al. [61] ont classé les attaques RPL en deux classes principales : les nouvelles attaques basées sur la spécification RPL, et les attaques de routage existantes. Dans ce qui suit, nous présenterons les attaques selon le dernier classement.

II.13.1 Nouvelles attaques basées sur la spécification RPL

Attaques de rang : L'attaque de rang augmenté consiste à augmenter volontairement la valeur du rang d'un nœud RPL afin de générer des boucles dans le réseau. Un nœud attaquant effectue l'attaque de rank en changeant illégitimement sa valeur de rang, attirant ainsi les nœuds voisins pour le sélectionner comme parent, en supposant que le nœud mène au nœud root dans le coût du chemin le plus court [63].

Attaque de voisin ou de répétition : Dans l'attaque de voisin, un nœud attaquant duplique et multicast tous les messages DIO valides provenant d'autres nœuds dans le réseau. Dans le cas des réseaux dynamiques, cette attaque est assez dommageable, car la topologie et les chemins de routage sont souvent modifiés. Les attaques par rejeu forcent les nœuds à mettre à jour leurs tables de routage avec des données obsolètes [61]. Cela conduit à créer de fausses routes, à perturber le réseau ou à consommer plus de ressources. Une autre variante de cette attaque est proposée dans [61] et appelé attaque par rejeu DIO.

Attaque par intrusion de choix de routage : Zhang et al. [63] ont proposé une nouvelle attaque de routage interne connue sous le nom d'intrusion de choix de routage, où l'attaquant apprend les règles de routage de RPL (choix d'acheminer les paquets), capture les messages de contrôle et diffuse de faux. Cette attaque est difficile à détecter, car l'intrus n'a qu'à ignorer la détection interne RPL légitime par lui-même et à fonctionner normalement. [17]. Cette attaque peut générer des routes non optimisées, créer des boucles et épuiser des ressources.

Attaques DAO : Les attaquants peuvent exploiter le mode de stockage du RPL pour déclencher des attaques liées au DAO. Ces types d'attaques incluent l'incohérence DAO et la falsification des tables de routage. Dans l'attaque d'incohérence DAO, les attaquants peuvent utiliser le drapeau Forwarding-Error F pour rendre un nœud disponible sur les routes descendantes [64]. Cette attaque vise à rendre la topologie du DODAG non optimisée et à isoler des nœuds. De plus, elle entraîne une augmentation du délai de bout en bout. Dans l'attaque de falsification de table de routage, il est possible de falsifier ou de modifier des informations de routage pour annoncer des routes falsifiées vers d'autres nœuds. Ce qui entraîne des délais, des pertes de paquets et/ou une congestion du réseau.

Attaques DIS : Les messages DIS sont envoyés par de nouveaux nœuds souhaitant rejoindre un DODAG. Ces messages permettent aux nœuds de solliciter des messages DIO des nœuds voisins. Il existe deux types d'attaques DIS. Dans le premier, un nœud malveillant uni diffuse périodiquement de faux messages DIS à un nœud qui répondra par un message DIO. Dans le second, le nœud malveillant envoie un multicast périodiquement des messages DIS à ses voisins qui doivent réinitialiser leurs minuteries, et donc envoyer plus de messages DIO [17]. Les attaques DIS peuvent être qualifiées d'attaques par inondation, car elles impliquent l'inondation de messages DIS dans le réseau [61]. Cela conduit à une augmentation du paquet de contrôle, épuisement de l'énergie des nœuds et interruption du routage.

Attaques par numéro de version : Le numéro de version est un champ important. Il est propagé sans changement dans le graphe DODAG et est incrémenté par le root uniquement, chaque fois qu'une reconstruction du DODAG est nécessaire [61]. Un attaquant peut changer le numéro de version en augmentant illégitimement ce champ dans les messages DIO lorsqu'il les envoie à ses voisins. Cette attaque conduit à augmenter la surcharge des messages de contrôle, et provoque une reconstruction inutile de l'ensemble du graphe DODAG et donc à épuiser les ressources. Nous expliquerons plus en détail cette attaque dans le chapitre suivant.

Attaque de réparation locale : Un attaquant peut déclencher à plusieurs reprises une réparation locale en modifiant le champ DODAG ID ou en diffusant un rang infini, ce qui conduit à mettre à jour la topologie du réseau, et donc à consommer plus de ressources [17], ainsi que la perturbation du processus de routage. Cette dernière attaque est définie dans [65] comme DAG Inconsistency Attack.

Attaques d'épuisement des ressources : Le et al. [66] [67] ont défini les attaques d'épuisement des ressources comme étant celles déclenchées lorsqu'un adversaire lance des activités gourmandes visant à épuiser les ressources des nœuds en faisant des traitements inutiles. Ce qui affecte la disponibilité du réseau en congestionnant les liaisons disponibles et donc à raccourcir la durée de vie du réseau.

II.13.2 Attaques existantes adaptées au contexte de RPL

Attaques HELLO Flood : Dans un réseau RPL, un attaquant peut se présenter comme un voisin des nœuds du réseau en diffusant des messages DIO - en tant que message HELLO - avec une puissance de signal élevée et une métrique de routage favorable [68]. Cette attaque conduit à la congestion du réseau et également à la saturation des nœuds RPL.

Attaque Sinkhole : Dans cette attaque, le nœud malveillant attire beaucoup de trafic en tant que parent préféré par ses voisins, en annonçant des informations falsifiées (par exemple, des liens ascendants et descendants de qualité supérieure). D'après les chercheurs dans [17], cette attaque est similaire à l'attaque de rang.

Attaques par trou noir : Dans cette attaque, un intrus malveillant supprime tous les paquets qui y transitent. Cette attaque peut être très dommageable lorsqu'elle est combinée à l'attaque sinkhole ou de rank provoquant la perte d'une grande partie du trafic [17][61].

Attaques par transfert sélectif : dans ces attaques, un nœud qui se comporte mal peut soit filtrer de manière agressive les messages de contrôle RPL afin de détruire les chemins de routage du réseau, soit supprimer des paquets de données et transférer uniquement le trafic des messages de contrôle, qui conduit à une attaque DoS car aucune donnée ne sera transmise aux nœuds de destination [17]. Ces attaques sont également connues sous le nom d'attaques de trou gris [68]. Elles visent à détruire la disponibilité des informations et des services.

Attaque par trou de ver : Dans les attaques par trou de ver, les nœuds malveillants créent un tunnel de communication direct qui est utilisé pour transmettre les données de trafic réseau. Ce lien de communication est nommé trou de ver (Wormhole en anglais). Dans les réseaux RPL si un attaquant tunnelier les informations de routage vers une autre partie du réseau, les nœuds qui sont réellement distants se voient comme s'ils étaient voisins, ce qui conduit à créer des routes non optimisées [61]. L'attaque Wormhole est très nocive, lorsqu'elle est combinée à d'autres attaques du réseau, telle que l'attaque Sybil.

Attaques Sybil et CloneID : les attaques Sybil et CloneID sont similaires et connues sous le nom d'attaques sur l'identité [24]. Dans une attaque CloneID (Spoofing attack), un attaquant copie la même identité logique sur plusieurs nœuds physiques [21]. Dans l'attaque Sybil, les nœuds malveillants forgent ou créent plusieurs identités afin d'induire en erreur les autres nœuds et prendre le contrôle du réseau.

Attaques par déni de service : les attaques DoS et DDoS visent à rendre les nœuds et/ou le réseau indisponible. DoS est généralement expliqué comme tout type de situation qui consomme des ressources et diminue la capacité d'un réseau, empêchant ainsi le réseau d'exécuter ses fonctionnalités correctement ou en temps opportun [34].

Attaques indirectes : Les attaques indirectes correspondent aux attaques où le nœud malveillant fait que d'autres nœuds génèrent une surcharge pour le réseau. Ils comprennent les attaques d'inondation, le brouillage et l'écrasement qui affectent indirectement les opérations de routage RPL et effectuent des attaques DoS contre le réseau. En effet, ces attaques dégradent le fonctionnement des nœuds en consommant des ressources, voire détruisent le trafic réseau [17].

II.14 Sécurité du RPL

Le protocole RPL est exposé à plusieurs attaques de sécurité internes et externes. Les caractéristiques des réseaux LLN telles que les contraintes de ressources, le manque d'infrastructure, la sécurité physique limitée, la topologie dynamique et les liens peu fiables les rendent particulièrement vulnérables et difficiles à protéger contre les attaques.

De nombreux auteurs ont proposé divers mécanismes de sécurité spécifiques au RPL. En termes de solutions standardisées basées sur la cryptographie, IPSec [69] et IEEE 802.15.4 PHY et Link Layer Security [62] peuvent être considérés respectivement pour la sécurité de bout en bout et hop-by-hop [51]. Cependant, la plupart des implémentations de RPL ne prennent pas en compte les mesures de sécurité en raison d'une spécification incomplète des mécanismes et des frais généraux d'implémentation [70]. Ces mécanismes de sécurité sont efficaces pour se défendre contre les attaques extérieures. Mais, ils échouent en cas d'attaques internes où un attaquant interne peut contourner les mécanismes de sécurité RPL appliqués et perturber les fonctionnalités du réseau.

En raison des milliards de dispositifs interconnectés au réseau, leur sécurisation et leur protection contre diverses attaques et menaces posent un défi critique. Par conséquent, il est très important de fournir des solutions légères de surveillance des performances et de la sécurité pour protéger les réseaux basés sur RPL.

Plusieurs travaux basés sur le calcul de confiance existent pour sécuriser RPL. Par exemple, les auteurs dans [59] [71] ont introduit une nouvelle métrique basée sur la confiance à utiliser lors de la construction de la topologie RPL [17]. Dans [66], les auteurs ont proposé une autre approche basée sur les spécifications axée sur la détection des attaques RPL. Ils ont spécifié le comportement RPL dans une machine à états finis, qui est utilisée pour surveiller le réseau et détecter les actions malveillantes [72].

II.15 Conclusion

Dans ce chapitre, nous avons fourni une description complète du protocole RPL. Tous d'abord, nous avons fourni les identifiants de ce protocole et son modèle de réseau. Puis nous avons montré comment le DODAG est construit et maintenu grâce à l'utilisation de différents messages de contrôle (DIO, DAO et DIS). Nous avons décrit également les mécanismes de réparation DODAG qui empêchent la perte de connectivité et assurent une topologie optimale, qui sont la réparation locale et la réparation globale, et définir la fonction objective et le fonctionnement de l'algorithme Trickle Timer. De plus, nous avons exposé les deux modes de fonctionnement (avec et sans stockage) et les différents paradigmes de communication. Enfin, nous avons montré que le protocole RPL est vulnérable à de nombreuses attaques de sécurité internes et externes, en raison des caractéristiques de ce protocole telles que le manque de ressources (c.-à-d. traitement, mémoire, stockage et réseau). Nous avons ensuite introduit certaines solutions de sécurité existantes que nous avons trouvées dans la littérature pour atténuer ces risques.

Le chapitre suivant sera consacré à l'étude de l'attaque numéro de version, et nous introduirons également notre approche pour la contrer.

Chapitre III : Etude de L’attaque Numéro de Version

Table des matières

III.1 Introduction

L’attaque par numéro de version exploite le mécanisme de réparation globale fourni par le protocole RPL, car il n'y a pas de mécanisme prévu pour protéger le champ du numéro de version. Un nœud malveillant peut augmenter illégitimement le numéro de version du DOGAG, ce qui conduit à une augmentation de la surcharge des paquets de contrôle et la consommation d'énergie. Par conséquent, de nombreux chercheurs ont tenté de trouver une solution pour se défendre contre cette attaque.

À travers ce chapitre, nous allons expliquer l'attaque par numéro de version et son impact sur le réseau. Ensuite, nous présenterons les mécanismes proposés par la littérature et

effectuerons une comparaison entre eux. De plus, nous présenterons notre stratégie de détection d'attaque numéro de version basée sur la cryptographie.

III.2 Attaque numéro de version VNA

Dans RPL, le VN est un champ important du message DIO. Seul le root est responsable de sa mise à jour (augmentation) et sa propagation dans tout le graphe DODAG. Lorsque le root a besoin de reconstruire l'ensemble du DODAG, ce qui est également appelé réparation globale, il incrémente la valeur du VN et l'envoie dans le champ du VN du message DIO aux nœuds enfants, où les nœuds sont déplacés de l'ancienne version du DODAG vers la nouvelle version,

Dans RPL, il n'y a aucun mécanisme pour vérifier si l'intégrité du VN est maintenue dans les messages DIO reçus. Un nœud malveillant peut modifier le VN en augmentant illégitimement ce champ dans les messages DIO et le transmettre aux voisins, afin de faire croire aux nœuds voisins que leur table de routage et les données de configuration du réseau sont obsolètes. Le but de l'attaquant est de forcer le nœud récepteur à appeler à une réparation globale et à réorganiser le DODAG, ce qui produit des boucles, la non-existence de routes, des retards et des abandons de paquets et la forte circulation des messages de contrôle. Par conséquent, les nœuds seront obligés à consommer une grande partie de leur énergie. De plus, le modèle de cette attaque la rend difficile à détecter ou à atténuer localement, car il n'y a aucune possibilité d'identifier précisément le nœud malveillant [56].

III.3 La quantification des conséquences de VNA dans RPL

L'attaque par VN est classée comme une menace d'intégrité, c'est-à-dire une attaque de modification non autorisée. Les auteurs dans [73] ont évalué la gravité de cette attaque et ont constaté qu'elle a un impact négatif sur la surcharge de contrôle et le taux de livraison, de telle sorte que la surcharge peut augmenter jusqu'à 18 fois, ce qui a un impact sur la consommation d'énergie et la disponibilité des canaux. Cela peut à son tour réduire le taux de livraison des paquets jusqu'à 30% et presque doubler le délai de bout en bout dans un réseau.

Aris et al. [74] ont découvert une forte corrélation entre la position de l'attaquant et l'effet sur le réseau. Lorsque l'attaquant est situé aussi loin que possible du root, il provoque une augmentation plus élevée de la surcharge de contrôle, de la perte de paquets, et de la consommation d'énergie des nœuds.

Dans [24], les chercheurs ont fait une analyse sur l'effet de plusieurs attaquants en tenant compte du nombre et des positions des attaquants. Les résultats des simulations ont montré

que l'augmentation du nombre d'attaquants affecte uniquement le taux de livraison des paquets et n'affecte pas le délai moyen du réseau et la consommation d'énergie moyenne.

III.4 Les travaux connexes

Dans la littérature, L'attaque par numéro de version n'a été étudié que dans quelques travaux puisqu'il s'agit d'un type assez nouveau d'attaque par déni de service (DoS) pour les LLN basés sur RPL. Nous citerons cinq mécanismes qui sont les suivants :

III.4.1 VeRA - VN et authentification de rang dans RPL

Dvir et al. [57] ont proposé le mécanisme VeRA qui utilise des chaînes de hachage à sens unique et des codes d'authentification de message (MAC) pour sécuriser les champs de rang et VN véhiculés dans les messages DIO pour empêcher les nœuds compromis de se faire passer pour le root et d'envoyer un numéro de version augmenté [57].

Tous d'abord, le root DODAG génère un nombre aléatoire r pour calcule une chaîne de hachage de numéro de version : $V_n, V_{n+1} \dots, V_1, V_0$, de taille n+1 où $V_n = h(r)$ et $V_i = h(V_{i+1})$. De plus, pour chaque Vi de la chaine de hachage de VN, le root génère également un nouveau nombre aléatoire X_i pour calcule une chaîne de hachage de rang : $R_{i,0}, R_{i,1} \dots, R_{i,l-1}, R_{i,l}$ et de taille $l + 1$ où $R_{i,0} = h(x_i)$ et $R_{i,j} = h(R_{i,j-1})$. Ensuite, afin d'amorcer la sécurité, le root DODAG diffuse un message DIO $\{V_0, VN_0, MAC_{V1}(R_{1,l}), \alpha\}$ à tous les nœuds, où $MAC_{V1}(R_{1,l})$ est une valeur de code d'authentification de message, VN_0 est une valeur initiale du VN et $\alpha = (V_0, MAC_{V1})_{sign}$ est la signature numérique [57].

Le nœud vérifie ces données et en cas de succès, les enregistre. Lors de la réception d'un message DIO avec un nouveau numéro de version ou une nouvelle valeur de rang où le message DIO = $\{V_i, VN_i, MAC_{Vi+1}(R_{i+1,l}), \alpha\}$, chaque nœud vérifie d'abord si le nouveau VN est supérieur au VN actuel (c'est-à-dire si $VN_i > VN_{i-1}$). Si tel est le cas, il continue de vérifier si le root est celui qui a mis à jour le VN en vérifiant si $h(V_i) = V_{i-1}$ est valide et vérifie la valeur Rank en vérifiant si $MAC_{V_i}(R_{i,l}) = MAC_{V_{i-1}}(R_{i-1,l})$ [57].

III.4.2 Détection des VNAs à l'aide d'une architecture de surveillance distribuée

Mayzaud et al. [8] ont proposé un mécanisme de détection de VNA qui utilise une architecture de surveillance distribuée pour préserver les ressources des nœuds contraints [8].

Cette architecture utilise deux types de nœuds, des nœuds surveillés qui effectuent la tâche de détection et constituent le réseau dit régulier, et des nœuds de surveillance qui ont des capacités plus élevées et peuvent effectuer des activités de surveillance et de détection en formant une seconde topologie de routage sans affecter leur capacité à acheminer les informations dans le réseau normal [8].

Pour détecter l'attaque et identifier les nœuds malveillants, ils ont proposé des algorithmes de détection et de localisation ; un algorithme détecte l'attaque et rassemble les informations de tous les nœuds de surveillance, et l'autre identifie l'attaquant en analysant les informations collectées. Les expériences ont démontré que la stratégie proposée donnait des résultats faussement positifs élevés, autrement dit, le nœud normal est considéré comme malveillant. À cet égard, les auteurs ont quantifié le nombre de nœuds de surveillance nécessaires pour une taille de topologie donnée et ont analysé l'évolutivité en fonction du placement des nœuds [8].

III.4.3 Mécanisme de vérification distribué pour contrer VNA

Ahmed et Ko [2] ont proposé un mécanisme de vérification distribué et coopératif pour détecter les nœuds malveillants. Le schéma proposé comprend deux cas [2].

- **Le premier cas** : un nœud malveillant près du root (à un saut) envoie des messages DIO avec un VN incrémenté à son voisin. Dans ce cas, il est facile à détecter par le root DODAG. Lors de la réception du message DIO d'un nœud malveillant, le nœud root compare son propre VN avec le Y contenu dans la réception du message DIO [2].
- **Le deuxième cas :** lorsque le nœud reçoit un message DIO, il vérifie si le VN reçu est supérieur à son propre VN. Si c'est le cas, le nœud commence une vérification pour savoir si le nœud expéditeur est malveillant ou non. Pour la vérification, le nœud de vérification stocke temporairement le VN et envoie un message DIO à ses voisins. Il sélectionne des voisins à deux sauts comme destination via son nœud intermédiaire. Il envoie ensuite un message CVQReq (demande de requête de vérification coopérative) = (adresse du nœud de destination, adresse du nœud de vérification, horodatage) pour obtenir leur VN actuel. À la réception d'un message CVQReq, les voisins à deux sauts du nœud de vérification répondent CVQRep (réponse à la requête de vérification coopérative) = (adresse du nœud de vérification, adresse du nœud de destination, horodatage, VN actuel) via les nœuds intermédiaires coopératifs vers le nœud de vérification. Après avoir reçu de nombreux messages CVQRep, le nœud de

vérification enregistre l'adresse du nœud intermédiaire, nœud de destination et VN correspondant dans une table de stockage. Puis il vérifie le VN de la table de stockage avec le VN stocké temporairement dans le nœud de vérification, s'ils sont différents alors le nœud expéditeur est un nœud malveillant. Si le nœud de vérification reçoit deux VNs différents de la même destination, il vérifie alors les deux nœuds intermédiaires de la table de stockage, car l'un d'eux est jugé malveillant [2].

III.4.4 Solution de détection des VNAs au niveau du Cloud

Dans cette étude [56], les chercheurs ont proposé un cadre pour détecter les VNAs dans l'IoT, qui a été déployé au niveau du Cloud par un service Cloud. En l'absence de contrainte de ressources, des algorithmes sont utilisés pour la détection des attaques et l'identification des nœuds malveillants. Le service de détection d'attaque comprend 6 modules [56].

Ils se sont appuyés sur certains phénomènes qui signalent la présence d'une attaque par VN (des baisses soudaines des intervalles de balise, une augmentation rapide de la consommation d'énergie et une augmentation de la métrique de routage). Ils ont fait des captures en direct sur des données LLN afin d'analyser les paramètres de détection des nœuds malveillants la fréquence de changement de VN initiée par chaque nœud est estimée pour trouver la liste des nœuds attaquants probables. Les nœuds avec des probabilités élevées sont marqués comme malveillants et leurs informations sont envoyées au routeur de périphérie et enregistrées dans la base de données. Le nœud récepteur à la réception de la liste des nœuds malveillants probables peut les lister en noir [56].

III.4.5 Nouvelles techniques d'atténuation légères pour les VNAs

Aris et al. [6] ont proposé deux techniques d'atténuation contre VNA.

- **La première technique « Elimination » :** consiste à élimine les mises à jour VN provenant de la direction des nœuds feuilles de meilleurs rangs (c'est-à-dire qu'il a une valeur inférieure) que son rang. Cependant, pour le reste des positions d'attaque, il ne peut pas atténuer les effets de l'attaque. Par conséquent, ils ont proposé une deuxième technique [6].
- **La deuxième technique « Shield » :** cette technique est capable d'atténuer l'effet de l'attaque, quels que soient les emplacements de l'attaquant. Un nœud change son VN si seulement si la majorité de ses voisins qui ont un meilleur rang que lui-même demande de mettre à jour le VN. Cela se fait grâce à l'utilisation de sa propre table appelée Shield-List, qui contient l'identité des nœuds voisins (adresse IP) qui ont un

rang supérieur à son rang et les champs VN. Lorsque le nœud reçoit un message DIO avec un VN supérieur, il vérifie si l'expéditeur existe dans sa table Shield-List. S'il est présent, le nœud met à jour le VN de l'expéditeur dans sa table et vérifie si au moins la moitié des entrées de sa propre table partagent la même mise à jour VN, il met à jour son VN. S'il n'existe pas dans la table, le message DIO portant ce nouveau VN est ignoré [6].

III.5 Comparaison entre les travaux

Le tableau III.1, présente une comparaison entre les différents mécanismes proposés mentionnés ci-dessus, en identifiant les avantages et les inconvénients de chaque mécanisme.

Travail	Avantages	Inconvénients
Mayzaud et al. [8]	Leur architecture de supervision distribuée préserve les ressources des nœuds contraints en formant une seconde topologie de routage. Détecte les attaques et identifie les nœuds malveillants.	Ce mécanisme souffre de faux positifs élevés et malgré leurs tentatives pour réduire le nombre de faux positifs par des stratégies de placement de nœuds, il semble difficile quand le réseau est grand, car il a besoin de plus de nœuds de surveillance de haut niveau. Ce qui ajoute des coûts supplémentaires. De plus, ils dépendent de la couverture des nœuds réguliers par les nœuds de surveillance.
Dvir et al. [57]	Ce schéma offre une protection contre l'attaque numéro de version capable d'envoyer des messages DIO avec des valeurs de numéro de version plus élevées, en sécurisant le numéro de version.	L'inconvénient de VeRA est le manque de la mise en œuvre du mécanisme proposé et aucune évaluation n'existe en termes de surcoût des ressources, mémoire, CPU, le temps et la consommation d'énergie n'ont pas été pris en compte. De plus, il a été découvert qu'elle était vulnérable à un nombre d'attaques de rang par Perrey et al. [100]. Il nécessite une surcharge de message de contrôle élevé en raison de l'utilisation de la signature numérique et des opérations MAC.
Aris et al.	La première technique assure que	Les techniques proposées ne considèrent

[6]	les positions les plus fortes pour les attaques par numéro de version sont atténuées. De plus, il ne prend pas beaucoup d'espace mémoire. Bien que la deuxième technique prenne plus d'espace mémoire que la première technique, mais elle est efficace pour atténuer l'effet de l'attaque sans tenir compte des emplacements de l'attaque. La consommation d'énergie diminue à 63%, Délai moyen diminue à 87 % et les frais généraux de contrôle réduit à 71 % en présence de VNA. Cela signifie que les techniques proposées étaient fiables.	pas la situation d'attaques multiples.
Ahmed et Ko [2]	Les résultats de l'évaluation ont montré que la surcharge de contrôle était réduite. Cela indique que le schéma proposé est efficace pour réduire l'impact de l'attaquant.	L'envoi fréquent des messages de demande de vérification par le nœud peut rendre le mécanisme proposé en lui-même une source d'attaque DOS. Aucune évaluation des besoins supplémentaires de communication, de RAM et de ROM du mécanisme proposé n'a été faite. Leur mécanisme proposé a besoin que chaque nœud connaisse les adresses de ses voisins à deux sauts.
Rashmi et al. [56]	Les résultats ont montré que leur approche est efficace pour détecter les attaques par numéro de version et identifier les nœuds malveillants. Les mécanismes proposés ne placent aucune surcharge de calcul sur les nœuds contraints.	

Tableau III.1: Comparaison entre les travaux.

III.6 Fonctionnement du RPL pour la vérification du VN : Rappel

Dans le RPL, le numéro de version (VN) est utilisé comme un indicateur de fonctionnement de la réparation globale. Selon la spécification, seul le root DODAG devrait le changer et ainsi lancer une réparation globale du réseau. Il est incrémenté de manière monotone par le root chaque fois qu'il décide de former une nouvelle version du DODAG afin de revalider l'intégrité et permettre des réparations globales. Si un nœud détient une ancienne copie du numéro de version, cela signifie que ses entrées de table de routage sont obsolètes. Quand il reçoit un message DIO avec un nouveau VN, d'après l'implémentation du RPL dans le système d'exploitation de l'IoT appelé Contiki, avant de participer directement à la réparation globale il suit les étapes de vérification suivantes comme illustrées dans la Figure III.1:

- Tout d'abord, il vérifie si le nouveau VN est supérieur au VN actuel. Si c'est vérifié :
  - Si le nœud récepteur est le root, il met à jour le VN et incrémente sa valeur, par la suite, il réinitialise le minuteur Trickle.
  - Sinon, il met à jour certaines informations, ensuite il participe à la réparation globale.
- Sinon, si le nouveau VN est inférieur au VN actuel, il considère que l'expéditeur DIO est sur une ancienne version de DAG et réinitialise le minuteur Trickle.

Nous remarquons que ce processus de vérification n'est pas efficace, car il n'y a aucune étape qui garantit que le root est celui qui a incrémenté le VN. Cela permet aux attaquants (nœuds malveillants) d'incrémenter constamment le VN et provoquer une instabilité du DODAG. Notre objectif est d'améliorer ce processus pour assurer que le root est celui qui a lancé une réparation globale.

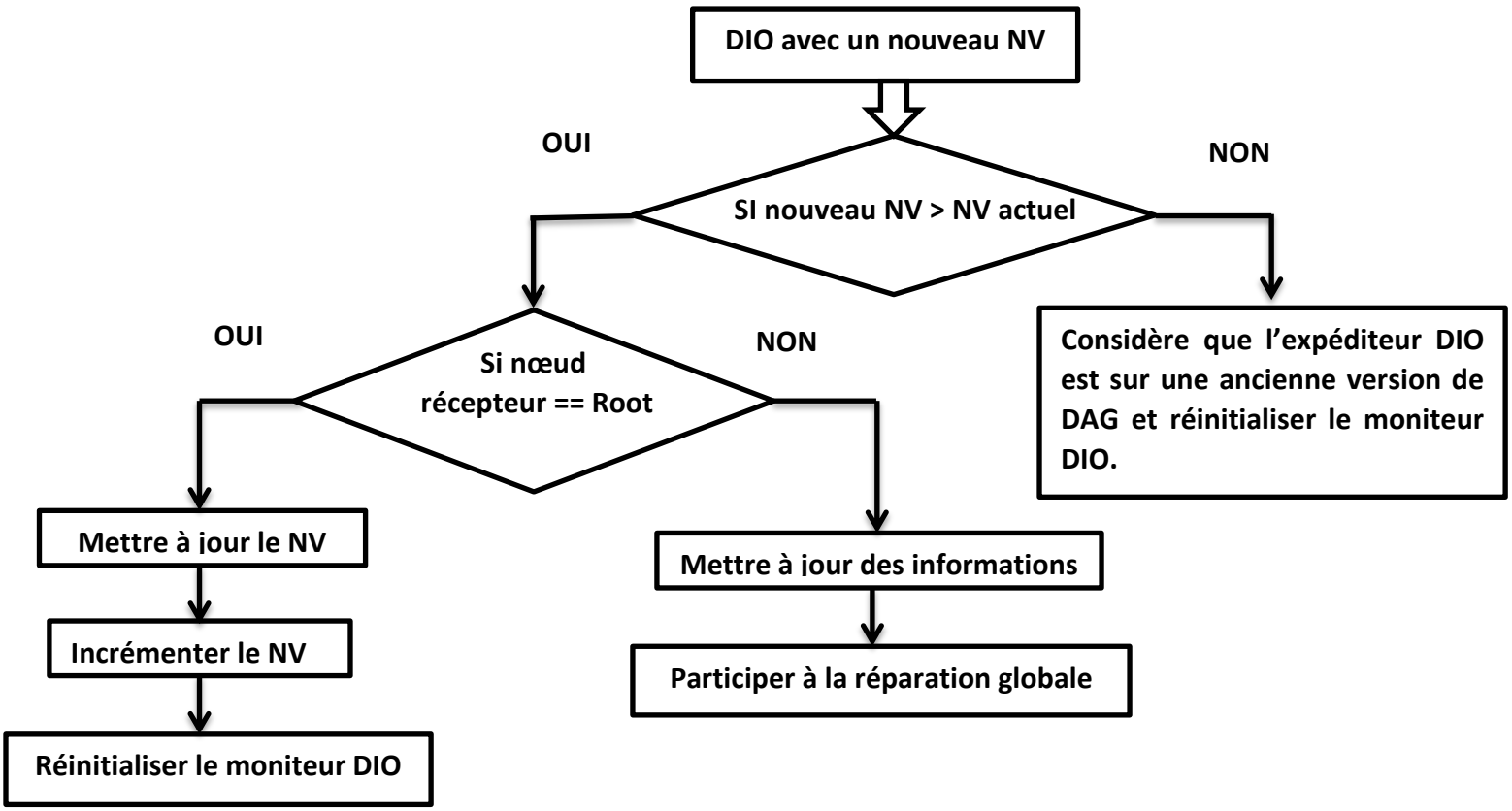


Figure III.1: La vérification du VN dans RPL.

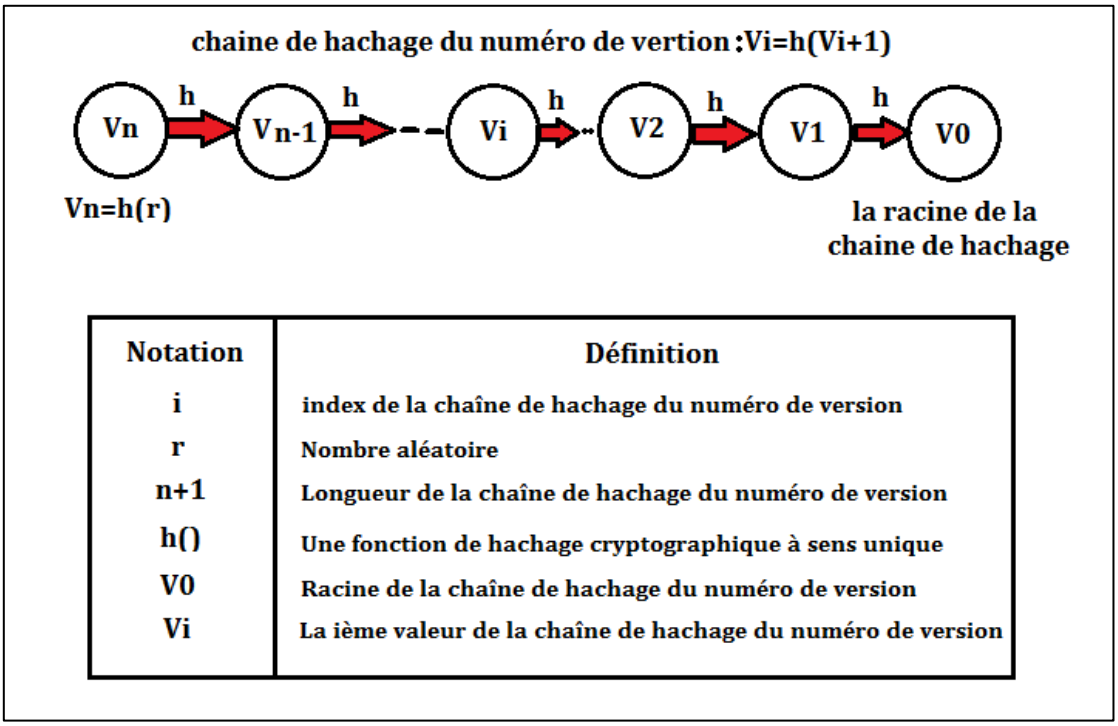
III.7 Notre approche

Dans le présent travail, nous proposons d'implémenter des algorithmes et des fonctions qui permettent de protéger un réseau basé sur RPL de l'attaque VN. Il faut noter que nous avons proposé une approche similaire à celle qui a été évoquée prédicedament par les chercheurs dans [57] et [75] pour contrer deux attaques en même temps qui sont l'attaque du VN et l'attaque sur le rang. Cependant, les chercheurs n'ont pas fait d'implémentation pour expérimenter leurs propositions. De plus, ils ont proposé d'utiliser des algorithmes de cryptographie trop gourmands pour des objets LLNs.

Dans ce qui suit nous présenterons notre approche. Dans le système proposé, il y a deux acteurs : le root et les nœuds collecteurs de données. Chaque acteur maintient une liste des nœuds suspects (statut = 0 ou 2) et malicieux (statut=1). De plus, les deux acteurs implémentent des fonctions de hachages différentes.

Tout d'abord, le root va calculer une chaine de hachage Vi=h(Vi+1) à partir d'une fonction « **h** » comme il est montré dans la Figure III.2. Le dernier hachage V0 représente la

valeur initiale de la version qui sera attribuée au DODAG par le root et envoyée dans le premier message DIO.



Notation	Définition
i	index de la chaîne de hachage du numéro de version
r	Nombre aléatoire
n+1	Longueur de la chaîne de hachage du numéro de version
h()	Une fonction de hachage cryptographique à sens unique
V0	Racine de la chaîne de hachage du numéro de version
Vi	La ième valeur de la chaîne de hachage du numéro de version

Figure III.2: La génération de la chaine de hachage.

Les nœuds du réseau implémentent la fonction de hachage de vérification « **h** » et ont une copie sauvegardée de l'ancien VN en cas où le nœud expéditeur n'a pas encore adhéré au DODAG actuel. Lors de la conception de notre approche, nous avons trouvé qu'elle peut être vulnérable à d'autres attaques, ou un nœud malveillant renvoie à chaque fois l'ancien VN, donc nous avons considéré deux types de nœuds suspects, l'un est de l'attaque du VN (statut =0) et l'autre de l'attaque par l'ancien VN (statut=2).

Quand un nœud reçoit un message DIO avec un VN différent de celui du DODAG actuel, comme il est résumé dans la Figure III.3, il suit les étapes de vérification suivantes :

- En premier lieu (vérification approfondie), il va vérifier si le nœud expéditeur n'existe pas dans la liste des nœuds suspects/malveillants, si c'est le cas il exécute l'algorithme dans la Figure III.4.
    - Tout d'abord, il commence par vérifier si le nouveau VN = ancien VN, si c'est vérifier :
        - Il envoie au nœud expéditeur un message DIO unicast.
        - Ensuite, il l'ajoute à la liste des nœuds malveillants avec un statut=2 c.-à-d. comme un suspect.

- Sinon (VN =/= ancien VN), il continue de vérifier si h (nouveau VN) = VN actuel, si c'est vérifier :
  - Il met à jour l'ancien VN.
  - Ensuite, il met à jour le VN ; c.-à-d. il participe à la réparation globale.
- Sinon (h (nouveau VN) =/= VN) :
  - Si le nœud récepteur est le root, il ajoute le nœud expéditeur à la liste des nœuds suspects/malveillants avec un statut=1 ; c.-à-d. comme un nœud malveillant.
  - Sinon, il l'ajoute à la liste des nœuds malveillants avec un statut=0 ; c.-à-d. comme suspect.

- En deuxième lieu (vérification initiale), si le nœud expéditeur existe dans la liste des nœuds suspecrts/malveillants il exécute l'algorithme dans la Figure III.5.
  - Si son statut = 0 ; c.-à-d. le nœud expéditeur est suspect dans l'attaque du VN
    - Tout d'abord, le nœud récepteur envoie une notification au root et augmente le nombre de notifications envoyées,
    - ensuite, si le nombre égale à un seuil, il va rendre le statut=1 ; c.-à-d. que le nœud expéditeur n'est plus un suspect, il a devenu un nœud malveillant.
  - Sinon, si son statut = 2 ; c.-à-d. le nœud expéditeur suspect dans l'attaque par l'ancien VN
    - Il vérifie d'abord si le nombre des messages DIO envoyés = 2 ; Si tel est le cas, il met le statut =1.
    - Sinon, le nœud récepteur envoie un message DIO unicast à celui-ci, et augmente le nombre des messages envoyés.

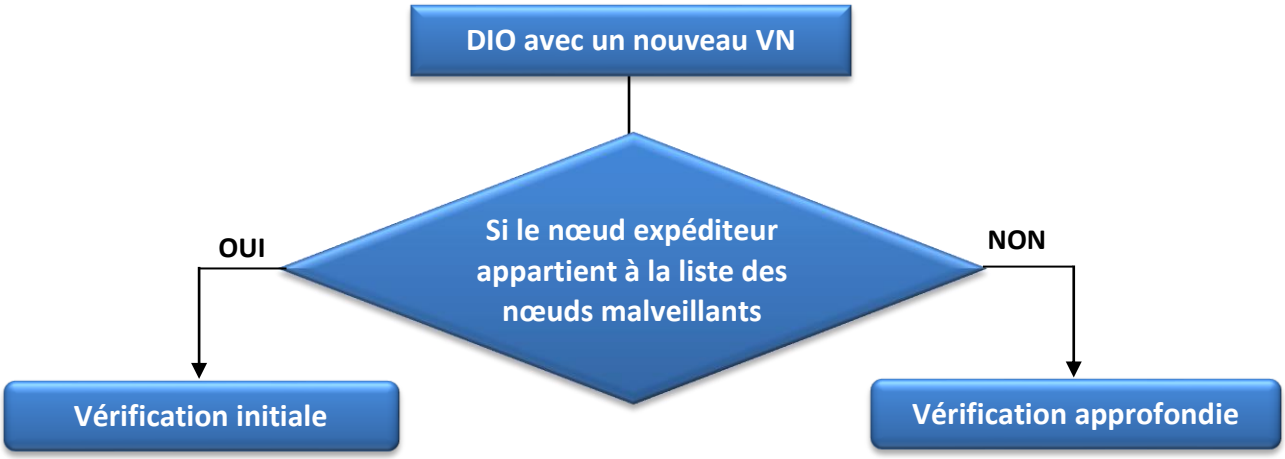


Figure III.3: Organigramme de l'approche proposé**.**

Dans les chapitres suivants, nous allons présenter plus de détails sur le fonctionnement de notre approche, ainsi qu'une estimation de la surcharge des algorithmes implémentés.

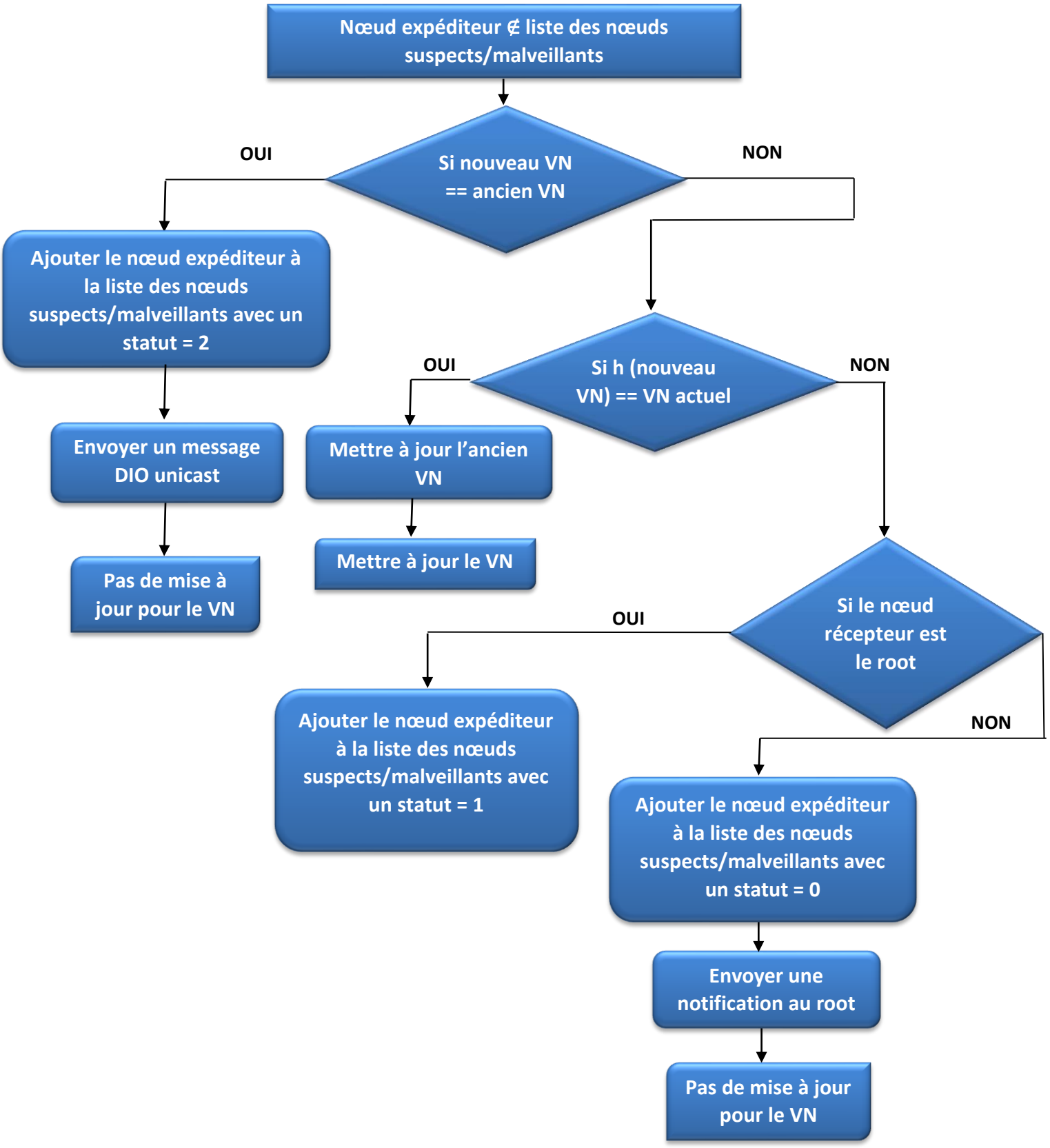


Figure III.4: Vérification approfondie.

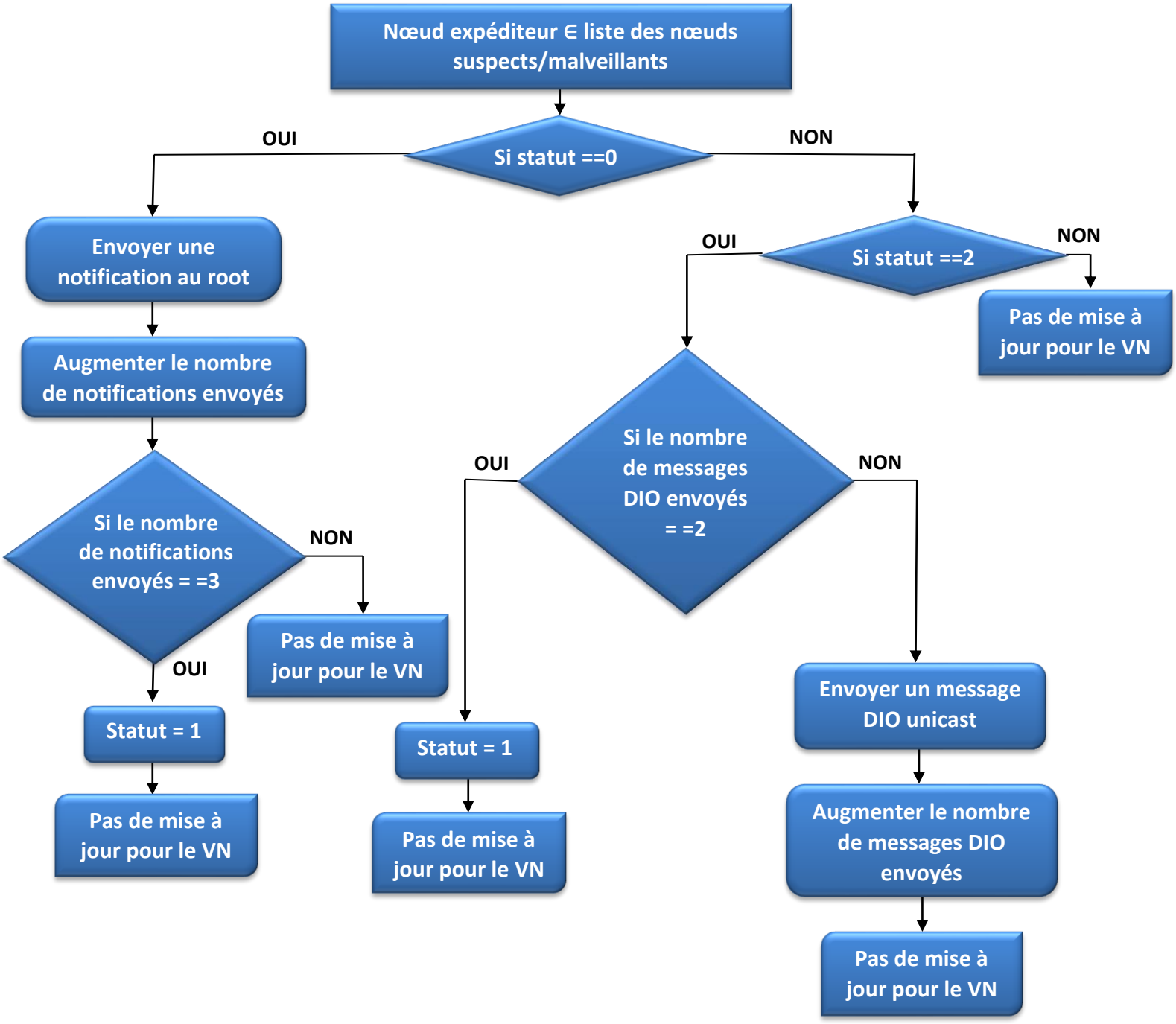


Figure III.5: Vérification initiale.

III.8 Conclusion

Dans ce chapitre, après avoir présenté une étude complète sur l'attaque VN. Il s'est avéré que cette attaque a de graves conséquences qui sont : perturbation du trafic, raccourcissement de la durée de vie du réseau, épuisement des ressources des nœuds, etc. Il a également été constaté qu'il existe une forte corrélation entre l'emplacement de l'attaquant et l'impact sur le réseau, où l'attaque est plus efficace lorsque l'attaquant est situé loin du 6LBR. Nous avons présenté et examiné ensuite certains mécanismes suggérés dans la littérature qui sont des mécanismes cryptographiques et des systèmes de détection d'intrusion pour atténuer ces attaques. Puis, nous avons fait une comparaison entre ces mécanismes, où nous avons identifié les avantages, les inconvénients et les limites possibles de chaque mécanisme. Enfin, nous avons présenté les détails de notre approche légère (lightweight) et efficace pour détecter et contrer l'attaque, et identifier les nœuds malveillants à l'origine de l'attaque.

Dans le chapitre qui suit, nous allons présenter les outils d’implémentation de notre solution et nous définirons le système d’exploitation ContikiOS.

Chapitre IV : Implémentation de L’approche

Table des matières

IV.1 Introduction

Ce chapitre est question de présenter notre stratégie de détection de l’attaque d'incohérence du DODAG, connue sous le nom attaque numéro de version, ciblant les réseaux LLN basés sur le protocole RPL. Ce présent chapitre couvre tous les détails relatifs à l’aspect implémentation de notre approche qui s'appuie sur la cryptographie définie dans le chapitre précédent. Pour ce faire, nous avons eu recours à l’outil d’implémentation ContikiOS qui nous a permis de simuler notre extension.

IV.2 L'outil d'implémentation Contiki OS

Contiki est un système d'exploitation léger, intégré, modulaire, flexible et générique, basé sur un modèle d'exploitation hybride pour les réseaux à mémoire limitée tels que les LLNs. Ce système a été conçu en 2004 par un groupe de développeurs de l'industrie et par Adam Dunkels de l'Institut Suédois d'Informatique. Comme tout système d'exploitation, son rôle est de gérer les ressources physiques telles que le processeur, la mémoire, les périphériques informatiques. Il occupe peu d'espace mémoire - moins de 10 Ko de mémoire vive (RAM) et 30 Ko de mémoire en lecture (ROM). Nous avons choisi Contiki, car il présente les avantages suivants :

- Simple.
- Open source.
- Implémente la pile protocolaire et une librairie complète du protocole RPL (ContikiRPL).
- Programmé en langage c.
- Prend en charge la réduction de la consommation d'énergie.

IV.3 Les fichiers et les fonctions de l'implémentation

Il est encore très difficile de trouver des approches efficaces pour simuler et évaluer le comportement de RPL. Nous avons opté pour la version 3.0 de Contiki OS. Cette version est trouvée sur le site « GitHub » elle fournit une implémentation open source de RPL. Elle est stable et met à disposition de nouvelles bibliothèques et fonctionnalités par rapport aux versions précédentes. Pour simuler notre travail recherché, nous avons apporté des modifications au code source du protocole RPL. Dans cette partie nous présenterons les fichiers, les fonctions et les structures concernés par les changements apportés dans le Tableau IV.1, Tableau IV.2, et le Tableau IV.3, respectivement.

Chemin d'accès	Fichiers	Description
Contiki/core/net/rpl/	rpl-dag.c	Logique pour les graphes acycliques dirigés en RPL. Contient des fonctions qui assurent la stabilité du DODAG. Par exemple : la sélection du parent, les calculs du rang, la réparation globale, la vérification du numéro de version, etc.
	rpl-icmp6.c	Contient les fonctions qui permettent d'envoyer et de recevoir les messages de contrôle RPL (DIO, DIS, DAO..) Avec le format ICMPv6.

	rpl-private.h	Contient la structure des messages DIO, ainsi que les variables et constantes utilisées par les fonctions qui gèrent ce type de message.
	rpl.h	Contient la structure des nœuds RPL.
Contiki/exampels/ip v6/rpl-collect /	Udp-sender.c	Contient des fonctions qui permettent de créer une connexion UDP et de gérer l'envoi et la réception des paquets périodiquement au niveau du **root**.
	Udp-sink.c	Contient des fonctions qui permettent de créer une connexion UDP et de gérer l'envoi et la réception périodique des paquets au niveau du **nœud**.

Tableau IV.1: Description des fichiers utilisés.

Fichier	Fonction	Description
rpl-dag.c	rpl_set_root(…)	Initialiser le root.
	global_repair(...)	Permet de reconstruire la topologie de routage DODAG.
	rpl_process_dio(…)	Permet de vérifier la crédibilité du numéro de version.
rpl-icmp6.c	dio_input(…)	Traite le buffer qui contient les informations du message DIO reçu, et faire une mise à jour des éléments du nœud si elle est nécessaire.
	dio_output(...)	Remplit le buffer du message DIO par les informations nécessaires afin d'être prêt à l'envoi.

Tableau IV.2: Description des fonctions utilisées.

Fichiers	Structure	Description
Rpl.h	Rpl_dag_t	Représente la structure d'un nœud du DODAG.
Rpl-private	Rpl_dio_t	Représente la structure du message DIO.

Tableau IV.3: Description des structures utilisées.

IV.4 Les étapes de l'implémentation

IV.4.1 Augmentation de la taille du VN

Le numéro de version est un entier non signé de 8 bits défini par le root DODAG. Comme notre solution est basée sur le hachage, on sait bien que plus la taille du résultat du hachage est grande, plus l'attaquant trouve une difficulté pour trouver le mot haché. Par conséquent, nous avons étendu la taille du numéro de version jusqu'à 32 bits. Pour ce faire,

nous avons apporté des modifications sur les structures et les fonctions mentionnées dans le Tableau IV.2 et le Tableau IV.3, respectivement.

IV.4.2 Procédures de hachage

Dans l'objectif d'avoir une solution robuste qui serait difficile à casser, nous avons opté à l'utilisation de plusieurs algorithmes de hachages simultanément. Nous avons randomisé le nombre de rotations des calculs comme le principe de **BCrypt**[1] [76] de manière à ne pas surcharger le nœud, ainsi que l'ajout du **salage**[2] [77] dans le but de compliquer la détermination de la donnée hachée.

De plus, nous avons optimisé le code du **MD5**[3]. Des lignes de code dans les fonctions de l'algorithme **MD5** [78] ont été changées pour permettre de réduire l'espace mémoire utilisé par l'algorithme et ainsi consommer moins de ressources de stockage et de calcul en comparaison avec ses homologues. Par exemple, les opérations de division et de multiplication ont été substituées par des opérations de décalage de bit quand applicable.

Les modifications apportées sont avérées légères rependant ainsi aux besoins de limitation de ressources de l'IoT tout en respectant les exigences de sécurité.

Nous avons ajouté un nouveau fichier dans la librairie RPL appelé **version.c**. Dans ce fichier, nous avons implémenté une fonction pour les nœuds et une procédure pour le root. Les deux se composent de deux fonctions principales : **MD5** et **adler32**[4] [79]. Elles fonctionnent comme il est montré dans le Tableau IV.4.

[1] BCrypt est une fonction de hachage créée par Niels Provos et David Mazières. Elle est basée sur l'algorithme de chiffrement Blowfish. En plus de l'utilisation d'un sel et le coût (nombre d'itérations pour créer le hash) pour se protéger des attaques par table arc-en-ciel (Rainbow table).

[2] Le salage, est une méthode permettant de renforcer la sécurité des informations qui sont destinées à être hachées en y ajoutant des données aléatoires avant qu'elles ne soient soumises à une fonction de hachage cryptographique.

[3] MD5 (Message Digest 5) est une fonction de hachage cryptographique, qui prend en entrée un message de longueur arbitraire et produit en sortie une « empreinte digitale » ou un « condensé de message » de 128 bits (32 caractères en notation hexadécimale).

[4] Adler32 est un algorithme de somme de contrôle qui a été inventé par Mark Adler et dérivée de la fonction de Fletcher. Il est une fonction rapide à calculer et a un petit peu d'espace.

La signature	Principe de Fonctionnement
Void vector_hash(const char *msg, hash_data_t *hashs) ;	L'appelle à cette procédure se fait au niveau du root dans la fonction **rpl_set_root()**, elle permet de remplir un vecteur vide de taille **N** et de type **hash_data_t** par une chaine de hachage calculé à partir d'un random donné. Tout d'abord, elle calcule aléatoirement le nombre d'itération et l'enregistre dans le vecteur, puis elle applique la fonction **MD5 nombre_d'itération** fois sur le random donné. Par la suite, elle fait une concaténation entre le résultat final du **MD5** et le **salage.** Enfin, elle minimise la taille du VN en hachant le résultat de la concaténation par la fonction **Adler32 et** enregistre le hachage final en parallèle avec le nombre d'itérations. Elle réexécute ce traitement N fois de telle sorte que le random prend à chaque fois la valeur du hachage final. (Voir la Figure IV.1.)
Unit32_t hash (unit8_t number, uint32_t tohash) ;	Chaque nœud fait appel à cette fonction au niveau de la fonction **rpl_process_dio()**. Tout d'abord, elle recalcule **md5(nouveau_VN)** selon la valeur du nombre d'itérations donnée. Par la suite, elle fait une concaténation entre le résultat final du md5 et le **salage.** Enfin, elle minimise la taille du résultat par la fonction **adler32** et retourne le résultat de cette dernière. (Voir la Figure IV.2.)

Tableau IV.4: Fonctionnement des fonctions de hachage.

```
1   Procédure vector_hash
2
3   entrées:
4   random :uint8_t;
5   *table_hachages: hash_data_t;
6
7   Début
8     pour (i=0;i<nombre des hachages souhaité ;i++) faire
9
10      calculer aléatoirement le nombre d'itération;
11      table_hachages[i].nombre_d'itération= le nombre d'itération calculé;
12
13      hachage_initial =md5(random);
14
15        pour(i=1;i<nombre_d'itération;i++) faire
16          hachage_secendaire =md5(hachage_initial);
17          hachage_initial= hachage_secendaire ;
18        fin-pour
19
20      Ajouter le salage;
21      hachage_final=adler32(hachage_secendaire + le salage);
22      table_hachages[i].hachage = hachage_final;
23
24    fin-pour
25  Fin
26
```

Figure IV.1: Code source de la fonction vector_hash.

```
fonction hash

entrées :
Nouveau_NV: uint32_t;
nombre_d'itération: uint8_t;

Début
 *salage: chaine de caractère;

  hachage_initial =md5(Nouveau_NV);
  pour(i=1;i<nombre_d'itération;i++) faire
    hachage_secendaire =md5(hachage_initial);
    hachage_initial= hachage_secendaire ;
  fin-pour

  hachage_final=adler(hachage_secendaire + salage);

  Retourne hachage_final;

Fin
```

Figure IV.2: Code source de la fonction hash.

IV.4.3 Les listes des nœuds suspects/malveillants

Lors de la réception d'un message DIO avec un nouveau VN, chaque nœud vérifie si le nœud root est celui qui a mis à jour le numéro de version. Si ce n'est pas le cas, le nœud va ajouter le nœud qui lui a envoyé le message DIO avec un VN illégitime à la liste des nœuds suspects/malveillants de type « **node_malv** » (Voir Figure IV.3). D' autre part, quand le root reçoit des notifications concernant un nœud malveillant, il va le mettre dans une liste de type « **node_mal »** (Voir Figure IV.3). Le tableau ci-dessous (Tableau IV.5) montre la différence entre les deux listes.

Niveau	Liste	Les éléments	Signification
Nœud	**node_malv**	Id_node_malv	L'identifiant du nœud suspect/malveillant.
		nbr_msg_send	Le nombre des notifications envoyé.
		State	Si prendre les valeurs 0 ou 2 c.-à-d. le nœud suspect. Sinon (la valeur 1) le nœud est malveillant.
Root	**node_mal**	Malv	L'identifiant du nœud suspect/malveillant
		Nbr	Le nombre des notifications reçu
		node[]	Vecteur qui contient les identifiants des nœuds qui ont signalé le nœud suspect/malveillant.
		ipaddr[]	Vecteur qui contient les adresses des nœuds qui ont signalé le nœud suspect/malveillant.

Tableau IV.5: Les éléments de la liste des nœuds malveillants.

```
struct node_malv {
    uint8_t idnode_malv;
    uint8_t nbr_msg_send;
    uint8_t state;
    struct node_malv *next;
};

struct node_mal {
    uint8_t malv;
    uint8_t nbr;
    uint8_t node[NBR_TABLE_CONF_MAX_NEIGHBORS];
    uip_ipaddr_t ipaddr[NBR_TABLE_CONF_MAX_NEIGHBORS];
    struct node_mal *next;
};
```

Figure IV.3: Code source des structures node_malv et node_mal.

IV.4.4 La vérification du VN et l'envoi des notifications

À la réception d'un message DIO, le nœud récepteur peut passer par deux niveaux de vérification du numéro de version.

V.4.4.1 Vérification Approfondie

La Figure IV.4 présente les blocs d'instruction qui résument l'implémentation des étapes de vérification du VN dans la fonction « **rpl_process_dio** ». Nous les détaillons par la suite (h : la fonction de hachage) :

- Le nœud commence par vérifier si le nouveau VN est égal à l'ancien VN, si c'est vérifier :
  - Il envoie au nœud expéditeur un message DIO unicast.
  - Ensuite, il l'ajoute à la liste des nœuds suspects/malveillants avec un statut=2 ; c.-à-d. comme un suspect.
- Sinon (différent de l'ancien VN), il continue de vérifier si h(nouveau VN) est égal au VN actuel, si c'est vérifier :
  - Il met à jour l'ancien VN.
  - Ensuite, il met à jour le VN ; c.-à-d. il participe à la réparation globale.
- Sinon (le hash est différent),
  - Si le nœud intermédiaire est le root, il ajoute le nœud expéditeur à la liste des nœuds suspects/malveillants avec un statut =1 ; c.-à-d. comme un nœud malveillant.
  - Sinon, il l'ajoute à la liste des nœuds suspects/malveillants avec un statut=0 ; c.-à-d. comme suspect.

```
Algorithme code source de la vérification au niveau 1

  Si (ancien_NV == nouveau_NV) alors
    dio_output(NV_actuel, noeud_expidéteur);
  Fin-Si

  Si (nouveau_NV != NV_actuel) alors
    Si (noeud_intermédiaire == root) alors
      insert(noeud_expidéteur,statut=1);
    Sinon
      hachage_nouveau_NV = hash(nombre_itération, nouveau_NV) ;

      Si (NV_actuel == hachage_nouveau_NV) alors
        global_repair();
      Sinon
        envoyer_signal(id_noeud_expidéteur);
        insert(noeud_expidéteur,statut=0);

      Fin-Si
    Fin-Si
  Fin-Si

```

Figure IV.4: Les étapes de vérification dans la fonction rpl_process_dio.

V.4.4.2 Vérification initiale

Lorsqu'un nœud reçoit un message DIO avec un nouveau VN, avant de faire appel à la fonction " **rpl_process_dio** " pour effectuer une vérification approfondie, il vérifie si le nœud expéditeur est déjà dans la liste des nœuds malicieux. Nous avons ajouté une fonction **test_malicious** qui se trouve dans le fichier **rpl-dag.c** (Voir Figure IV.5) et qui fait des vérifications au niveau des listes, elle se base sur les tests suivants :

- Si son statut = 0 ; c.-à-d. le nœud expéditeur est un suspect dans VNA
  - Le nœud récepteur envoie une notification au root et incrémente le nombre des notifications envoyées,
  - ensuite, si le nombre = 3, il va rendre le statut =1 ; c.-à-d. que le nœud expéditeur n'est plus un suspect, il est devenu un nœud malveillant.
  - Retourne 0
- Sinon, si son statut = 2 ; c.-à-d. le nœud expéditeur suspect dans l'attaque par l'ancien VN,
  - Il vérifie d'abord si le nombre des messages DIO envoyés = 2. Si tel est le cas, il met le statut =1.
  - Sinon, le nœud récepteur envoie un message DIO unicast à celui-ci, et incrémente le nombre des messages envoyés.
  - Retourne 0

- Sinon, c.-à-d. le nœud récepteur n'existe pas dans la liste des nœuds malveillants, la fonction retourne 1 pour que le nœud récepteur puisse appeler la fonction " rpl_process_dio " pour continuer la vérification approfondie.

Lorsque le statut d'un nœud suspect devient égal à 1, le nœud récepteur peut toujours recevoir des messages DIO de celui-ci, mais il ne fait jamais l'appel à la fonction « **rpl_process_dio** ».

```
1   fonction  test_malicious
2
3   entrées:
4   id-noeud-expéditeur: uint8_t;
5   @noeud_expidéteur :uip_ipaddr_t;
6
7   Début
8    *ptr :node_malv;
9       ptr= la tête de la liste des noeuds suspects ou  malveilants;
10   tant que (ptr != NULL) faire
11     Si (ptr->idnode_malv == id-noeud-expéditeur ) alors
12        Si (ptr->state==1) alors Retourne 0;
13         Fin-Si
14        Si (ptr->state==0) alors
15           envoyer_signal(id-noeud-expéditeur);
16           nombre_signaux_envoyés ++;
17           Si (nombre_signaux_envoyés==3) alors ptr->state=1;
18           Fin-Si
19        Sinon
20              Si(ptr->state==2) alors
21                 Si (nombre_msgDIO_envoyés==2) alors ptr->state=1;
22                 Sinon
23                   dio_output(NV_actuel, @noeud_expidéteur);
24                   nombre_msgDIO_envoyés++;
25                 Fin-Si
26               Fin-Si
27           Fin-Si
28       Retourne 0;
29     Fin-Si
30   Fin-tant que
31   Retourne 1;
32  Fin
```

Figure IV.5: Code source de la fonction test_malicious.

V.4.5 Le rôle du root

Le root a deux rôles principaux :

- Le premier rôle est de construire la chaine de hachage par l'appel à la fonction vector_hash en donnant un random et un pointeur vers un tableau vide de type hash_data_t . La Figure IV.6 illustre la génération d'une chaine de hachage.

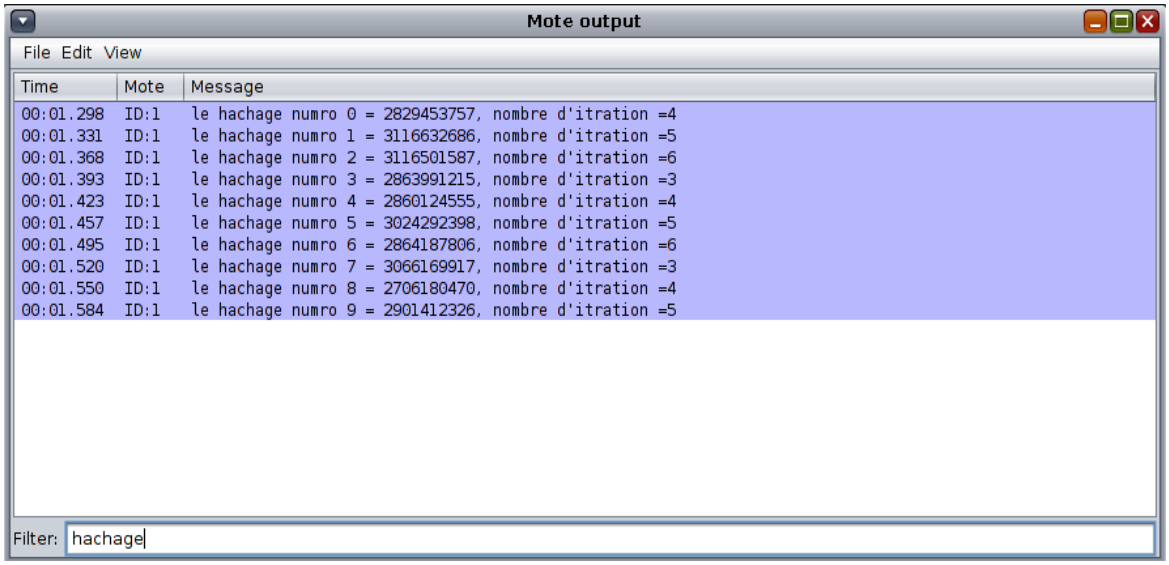


Figure IV.6: Exemple sur le résultat de la fonction vector_hash.

- Le deuxième rôle et de recevoir et gérer les notifications des nœuds et les enregistrer dans une liste de type **node_mal** par l'appel à la procédure statique **tcp_handler1**. Si le nombre de notifications concernant un nœud suspect devient égal au seuil défini par l'administrateur du réseau, le root va envoyer un message unicast (notification) aux voisinages du nœud malveillant pour qu'ils mettent à jour son statut=1 ; c.-à-d. qu'il n'est plus suspect, il est inévitablement malveillant, comme il est montré dans la Figure IV.7.

```
1  Procédure  tcpip_handler1
2  Début
3      noeud_existe : boolen;
4      signal_before: boolen;
5      *ptr :node_mal;
6        ptr= la tête de la liste des noeuds suspects ou malveilants;
7        noeud_existe=0;
8    Si(ptr == NULL) alors
9      insert(id_node_suspect,id_noeud_expéditeur,@noeud_expéditeur);
10     noeud_existe=1;
11   Sinon
```

```
12    tant que (ptr != NULL) faire
13      Si (ptr->malv == id_node_suspect) alors
14        noeud_existe =1;
15        signal_before=0;
16        i=0;
17          tantque(ptr->id_noeud_expéditeur[i]!=0 && signal_before==0)faire
18            Si (id_noeud_expéditeur==ptr->id_noeud_expéditeur[i]) alors
19                signal_before=1;
20            Fin-Si
21            i++;
22          Fin-tant-que
23          Si (signal_before==0) alors
24            prt->nbr++;
25            ptr->node[i]=id_noeud_expéditeur;
26            ptr->ipaddr[i]=@noeud_expéditeur;
27              Si(prt->nbr >= seuil) alors
28                  send_packet(id_node_malv,@noeud_expéditeur);
29              Fin-Si
30          Fin-Si
31      Fin-Si
32      ptr = ptr->next;
33    Fin-tant-que
34  Fin-Si
35  Si(noeud_existe==0)alors
36    insert(id_node_suspect,id_noeud_expéditeur,@noeud_expéditeur);
37  Fin-Si
38 Fin
```

Figure IV.7: Code source de la fonction tcp_handler1.

IV.4.6 Autres modifications

Nous avons ajouté quelques fonctions et variables clés qui ont un rôle actif dans la coordination des différentes opérations, parmi eux, les fonctions **rpl_process_malicious (**rpl-dag.c) et **send_signal (**Contiki/exampels/ipv6/rpl-collect /collect_common.c) qui gèrent les notifications ainsi que la variable **version_old** qui sauvegarde la valeur de l'ancien VN.

De plus, comme nous l'avions mentionné, la fonction de hachage dépend du principe de **BCrypt**, où chaque VN est associé à un nombre d'itérations. Nous avons utilisé le champ **reserved** dans le message DIO pour envoyer le nombre d'itérations avec le VN. En fait, le champ **reserved** prenant toujours la valeur 0 et n'était pas utilisé par les nœuds.

IV.5 L'implémentation d'un modèle d'attaque VN

Dans le chapitre suivant, nous allons évaluer les performances de notre solution. Pour cela, nous avons besoin de commencer par la simulation de l'attaque VN. Pour ce faire, nous avons ajouté un autre type de nœud (nœud malveillant) qui a les caractéristiques suivantes :

- Dans un instant t le nœud se transforme en un nœud malveillant en utilisant la fonction **get_time ()** qui se trouve dans le fichier Contiki/exampels/ipv6/rpl-collect /**collect-common.c**.
- Envoyer un faux VN : lors de l'envoi du message DIO par la fonction **dio_output ()**, au lieu d'envoyer le VN actuel, il va envoyer VN actuel +1. (Voir la Figure IV.8, Algorithme1).
- Réinitialiser le minuteur DIO chaque une minute : Pour cela, nous avons ajouté un appel à la fonction **rpl_reset_dio_timer()** dans la fonction **collect_common_send** qui se trouvent dans les fichiers **core/net/rpl/rpl-timers.c** et **udp-sender.**c, respectivement (Voir Figure IV.8, Algorithme2).

```
1  Algorithme 1
2
3  switch(node_id)
4  {
5  case 10:
6          Si(get_time()>= 300 )alors
7            buffer[pos++] = dag->version + 1;
8          Sinon  buffer[pos++] = dag->version ;
9          Fin-Si
10         break;
11 case 17:
12         if(get_time()>= 420){
13        buffer[pos++] = dag->version+ 1;}
14         else  buffer[pos++] = dag->version ;
15        break;
16 }
```

```
1  Algorithme 2
2
3  switch(node_id)
4  {
5  case 10:
6          Si(get_time()>= 300 ) alors
7            rpl_reset_dio_timer(rpl_get_default_instance());
8          Fin-Si
9          break;
10 case 5:
11         Si(get_time()>= 420 ) alors
12          rpl_reset_dio_timer(rpl_get_default_instance());
13         Fin-Si
14         break;
15 }
```

Figure IV.8: Partie du code de l'attaque.

IV.6 Conclusion

À travers ce chapitre, nous avons commencé par présenter l'outil d'implémentation Contiki OS. Comme nous avons défini les différents fichiers et fonctions concernés par VN. De plus, nous avons détaillé les étapes de l'implémentation en expliquant les modifications apportées pour détecter les nœuds malveillants à l'origine de l'attaque.

Le prochain et dernier chapitre définira les paramètres et les scénarios de la simulation qui ont été déroulés pour obtenir des résultats nous permettant de faire des études comparatives afin d'évaluer les capacités de performance et de fonctionnement de notre solution.

Chapitre V : Evaluation des Performances

Table des matières

V.1 Introduction

Dans les chapitres précédents, nous avons expliqué les détails de conception et d'implémentation de notre stratégie qui a été proposée dans le but de détecter les nœuds qui lancent des attaques par numéro de version ciblant les réseaux RPL. Nous allons consacrer ce chapitre pour présenter et analyser les résultats obtenus lors de la simulation de notre approche. Avant d'effectuer divers tests de simulations portant sur différents scénarios, nous présentons le simulateur Cooja, les paramètres et les scénarios pris en charge. Aussi, nous proposons ici, d'évaluer les performances des modules de détection à travers plusieurs métriques.

V.2 Environnement de simulation

V.2.1 Le simulateur Cooja

Pour dérouler les tests de simulations, nous avons utilisé le simulateur Cooja fourni par Contiki. Ce dernier permet d'émuler des nœuds simultanément à différents niveaux, y compris le niveau du système d'exploitation et le niveau du réseau, et de charger un programme compilé. Avec son modèle de simulation détaillé et très proche de la réalité, il est particulièrement utile pour tester les programmes avant de les mettre dans la mémoire flash des nœuds réels. Parmi les caractéristiques de Cooja nous citons [80] :

- COOJA combine des simulations de capteur matériel de nœud et simulation du comportement de haut niveau en une seule simulation.
- COOJA est flexible et extensible de sorte que tous les niveaux du système peuvent être modifiés ou remplacés.
- COOJA est une application Java, toutes les interactions avec le code Contiki se font à travers Java Native Interface (JNI).
- Chaque nœud simulé possède trois propriétés basiques : sa mémoire de donnée, son type de nœud et ses périphériques matériels.

L'interface du simulateur Cooja est composée de plusieurs fenêtres. La Figure V.1 montre l'interface de nos simulations qui se compose de six fenêtres :

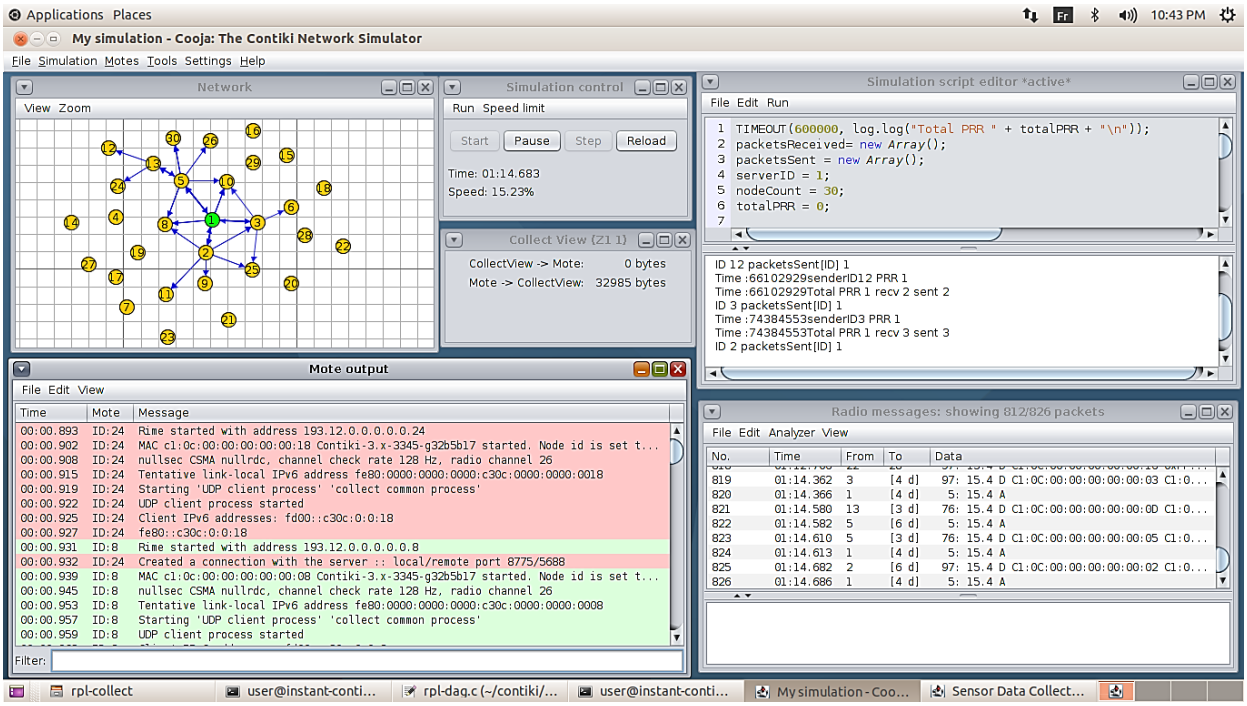


Figure V.1: Interface du simulateur Cooja.

- **La fenêtre Network :** présente la représentation graphique du réseau simulé et le flux de communication durant la simulation. Elle visualise l'état de chaque nœud du réseau (identifiant, type, champ de transmission, etc.).

- **La fenêtre Mote Output** : c'est dans cette partie où nous obtenons les affichages de notre implémentation. Elle nous permet de voir tous les messages d'entrée/sortie échangés entre les nœuds du réseau. Cette fenêtre contient un champ de texte qui permet d'entrer un filtre pour cibler un nœud ou un type de message en particulier.

- **La fenêtre Simulation control** : cette fenêtre contient quatre boutons :
    1) Start : pour démarrer une simulation
    2) Pause : pour arrêter la simulation.
    3) Reload : pour recharger une simulation.
    4) Step : pour régler la vitesse de la simulation.

- **La fenêtre simulation script editor** : l'éditeur de script peut être utilisé simplement pour afficher des messages dans le volet de sortie de l'éditeur de script et définir une minuterie sur la simulation. On peut utiliser cette fonction pour effectuer une analyse fine de la sortie du réseau.

- **La fenêtre Radio messages** : elle permet d'enregistrer le trafic réseau en tant que fichier PCAP pour pouvoir l'ouvrir, l'analyser et l'exporter au format CSV avec un programme d'analyse du trafic réseau.

- **La fenêtre Sensor data collect** : Une fois la simulation est terminée, une grande quantité de données sera disponible à partir de cette fenêtre, qui pourraient ensuite permettre à un utilisateur de produire ses propres analyses et graphiques. Parmi ces informations la consommation d'énergie par les nœuds du réseau.

V.2.2 Foren6

Foren6 est un outil de diagnostic et d'analyse pour des réseaux 6LoWPAN. Il capture tous les paquets de données en temps réel et détecte les activités anormales du réseau déclenchées par certains nœuds de mauvais comportement. Aussi, foren6 est capable de traiter les paquets à partir d'autres plaques-formes telles que RIOT, mais ils doivent être enregistrés en tant que fichiers PCAP [81]. La Figure V.2 montre son interface qui se compose de deux fenêtres :

- La fenêtre Network Visualizer offre une représentation graphique du réseau.
- La fenêtre d'informations : contient toutes les informations sur un nœud sélectionné.

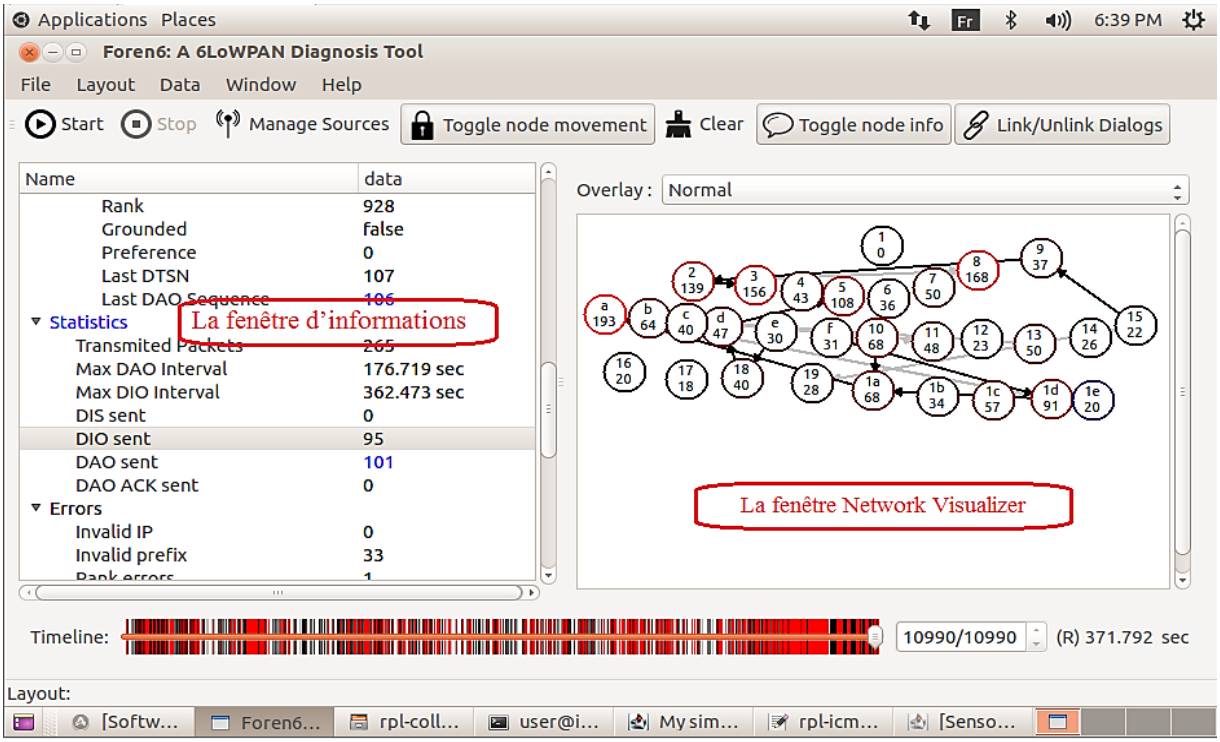


Figure V.2: Interface de Foren6.

V.3 La configuration de la simulation paramètres et métriques

Dans nos simulations, nous avons effectué une évaluation du schéma proposé par rapport au RPL standard et au RPL avec nœuds malveillants avec notre solution. Le mot émulé Z1 a été utilisé comme plate-forme de développement, car ses ressources, bien que limitées, sont suffisantes pour que notre implémentation fonctionne. Nous avons mis en place une topologie de 30 nœuds statiques répartis d'une manière aléatoire, où chaque nœud envoie des relevés de température et d'autres informations au root (nœud avec id =1) périodiquement (c'est-à-dire toutes les minutes). Nous avons utilisé le modèle de perte basé sur la distance comme support radio avec des portées de transmission et d'interférence de 35 m et 55 m

respectivement. La topologie de simulation et la forme du DODAG sont illustrées sur la Figure V.3. Les paramètres de simulation et leurs valeurs sont donnés dans le Tableau V.1.

Paramètre	Valeur
Système opérateur	Contiki OS
Simulateur	Simulateur Cooja
Couche d'adaptation	6LOWPAN
Protocole de routage	RPL
Couche physique	IEEE 802.14.4
Métrique de routage	Nombre de transmissions attendu (EXT)
Type de mot	Mote Z1
Radio moyen modèle	Unit Disk Graph Medium (UDGM) : Perte de distance
Taille de la zone de déploiement	100m x 100m
Nombre de nœuds	30
Nombre de nœuds malveillants	2
Champ de transmission	35
Champ de l'interférence	55
Taux de réception	80%
Taux de transmission	100%
Fonction objective	MRHOF

Tableau V.1: Paramètres de simulation.

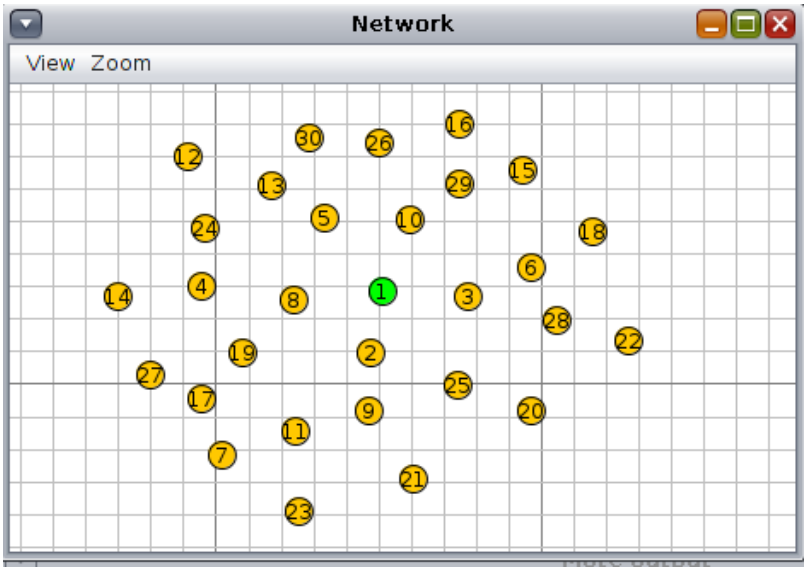


Figure V.3: La topologie de simulation sans nœuds malicieux.

V.4 Scénarios de simulation

Afin d'analyser l'effet de l'attaque du numéro de version et d'évaluer les performances de la solution proposée, nous nous sommes basés sur les six scénarios suivants :

1. **RPL :** nous avons d'abord exécuté des simulations sans aucune modification pour obtenir des résultats de performance de base.
2. **RPL avec solution (RPL+Sol).**
3. **RPL avec réparation globale (RPL+ RG) :** nous avant lancé une réparation globale après six minutes de temps de simulation, de sorte que le réseau ait suffisamment de temps pour se stabiliser et atteindre une topologie RPL stable.
4. **RPL avec solution et réparation globale (RPL+Sol+RG).**
5. **RPL avec attaque numéro de version (RPL+ANV) :** dans un scénario d'attaque, nous avons fixé l'emplacement des attaquants au niveau des nœuds 10 et 17 comme il est montré dans la Figure V.4. Ils commencent respectivement après cinq et sept minutes de temps de simulation en exécutant le modèle d'attaque introduit dans le chapitre précédent.
6. **RPL avec solution et attaque numéro de version (RPL+Sol+ANV).**

Cet ensemble complet de scénarios de simulation est répété dix fois pour des raisons de précision en modifiant l'emplacement des attaquants. Chaque simulation a duré dix minutes ce qui est suffisant pour réaliser les tests et obtenir suffisamment de résultats pour la comparaison, car le modèle d'attaque qui a été implémenté affecte rapidement et significativement la stabilité du DODAG.

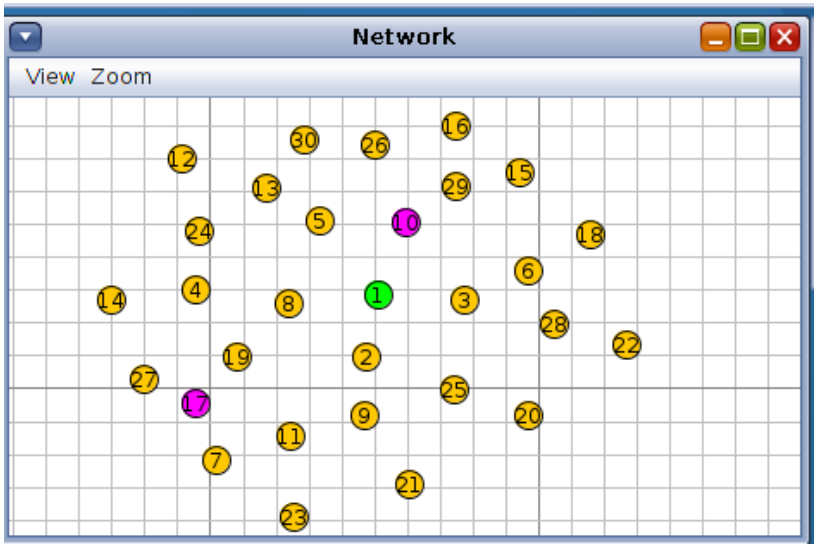


Figure V.4: La topologie de la simulation avec les nœuds malveillants (10 et 17).

V.5 Mesures de performance

Les mesures de performance utilisées peuvent être présentées en deux classes :

- Mesures de performance du réseau :
  - **Le taux de paquets délivrés (Packet Delivery Ratio : PDR) :** correspond au rapport entre le nombre des paquets de données reçus par la destination et le nombre de paquets de données envoyés par la source. Le PDR est calculé comme suit :

$$PDR=\frac{\sum Paquets\_reçus}{\sum Paquets\_envoyes} \quad (V.1)$$

  - **La surcharge du trafic de contrôle :** Constitue le total des messages DIO transmis dans le réseau, ce sont l'un des messages nécessaires pour former et maintenir un RPL DODAG. Nous avons utilisé foren6 pour calculer la somme des messages DIO envoyés par chaque nœud du réseau.
- **Statistiques des besoins en ressources :**
  - **Consommation moyenne d'énergie:** est une autre mesure de performance importante lorsque nous ciblons les LLN où les nœuds ont des contraintes de batterie. Grâce à la fenêtre, **Sensor data collect** de Cooja, nous pouvons récupérer le pourcentage de consommation d'énergie de chaque nœud durant la simulation ainsi que la consommation moyenne du réseau.

V.6 Résultats de simulation et discussion

Dans toutes les expériences, dès que l'attaque se produit, notre stratégie de détection a réussi à localiser rapidement l'attaquant, ce dernier a été bien identifié par ses voisins. D'après les listes des nœuds suspects/malveillants (des nœuds) nous n'avons pas eu des résultats faux positifs, pour cette raison, nous n'avons pas pris en considération les mesures de précision. Nous avons effectué nos simulations pour vérifier si la solution proposée est efficace en termes de la stabilité du taux de livraison des paquets et de la surcharge des messages de contrôle, ainsi, pour apercevoir si les algorithmes ajoutés consomment beaucoup d'énergie pendant leur fonctionnement. Les résultats obtenus lors de la réalisation des tests sont représentés dans la section suivante.

V.6.1 Résultats relatifs aux taux de paquets délivrés PDR

Les résultats obtenus dans la Figure V.5 montrent que le PDR prend des valeurs presque identiques dans les quatre premiers scénarios qui veulent dire que notre solution n'a pas d'effet négatif sur l'échange de paquets de données dans l'état normal et pendant la réparation globale. Cependant, lorsque le VNA est en place, nous pouvons voir que le PDR commence à baisser à 32% dans le scénario sans solution. Ceci est évident, car les attaquants entraînent l'abandon ou la perte de paquets pour diverses raisons telles que la collision de paquets, la rupture de lien, etc. Par contre, avec la solution, quasiment tous les paquets ont été livrés au root DODAG avec succès.

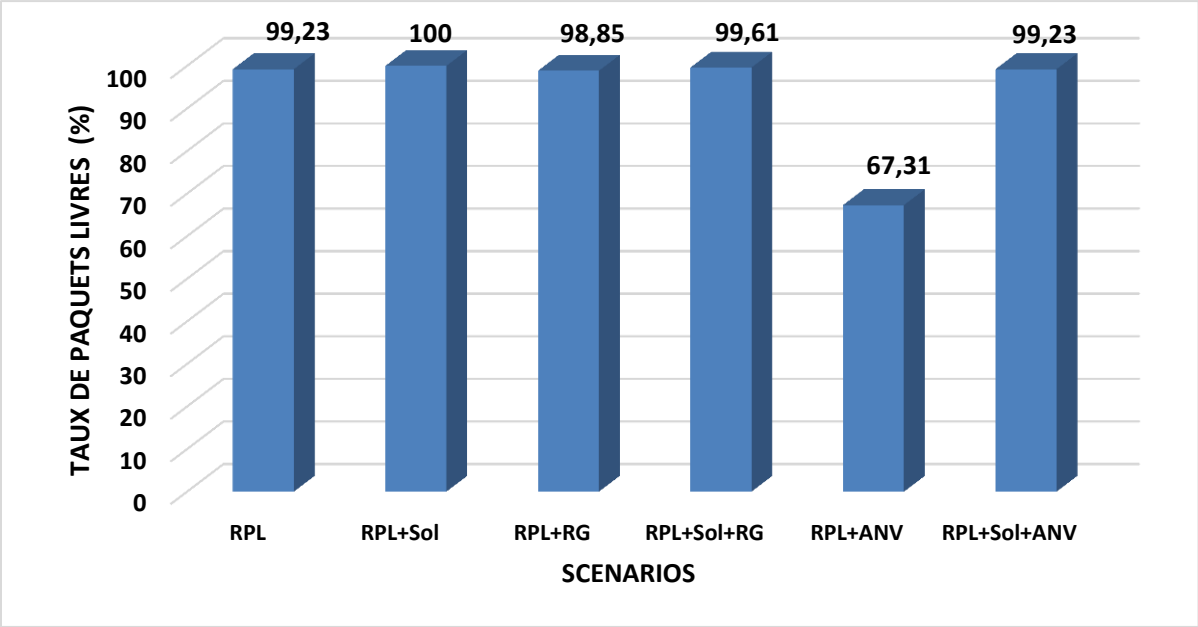


Figure V.5: Le pourcentage de PDR selon les scénarios de simulation.

V.6.2 Résultat relatif à la surcharge du trafic de contrôle

En se référant à la Figure V.6, la surcharge de message de contrôle dans le troisième et le quatrième scénario augmente lorsqu'une réparation globale est lancée, ce qui est normal, car les nœuds du réseau diffusent des DIO pour reconstruire le graphe DODAG.

En présence des nœuds malveillants, la surcharge augmente de 2346 messages en raison des caractéristiques de l'attaque par numéro de version. En revanche, le schéma proposé empêche la diffusion des DIO suspects pour reconstruire le DODAG, par conséquent, la surcharge de contrôle est bien inférieure à celle du RPL avec VNA.

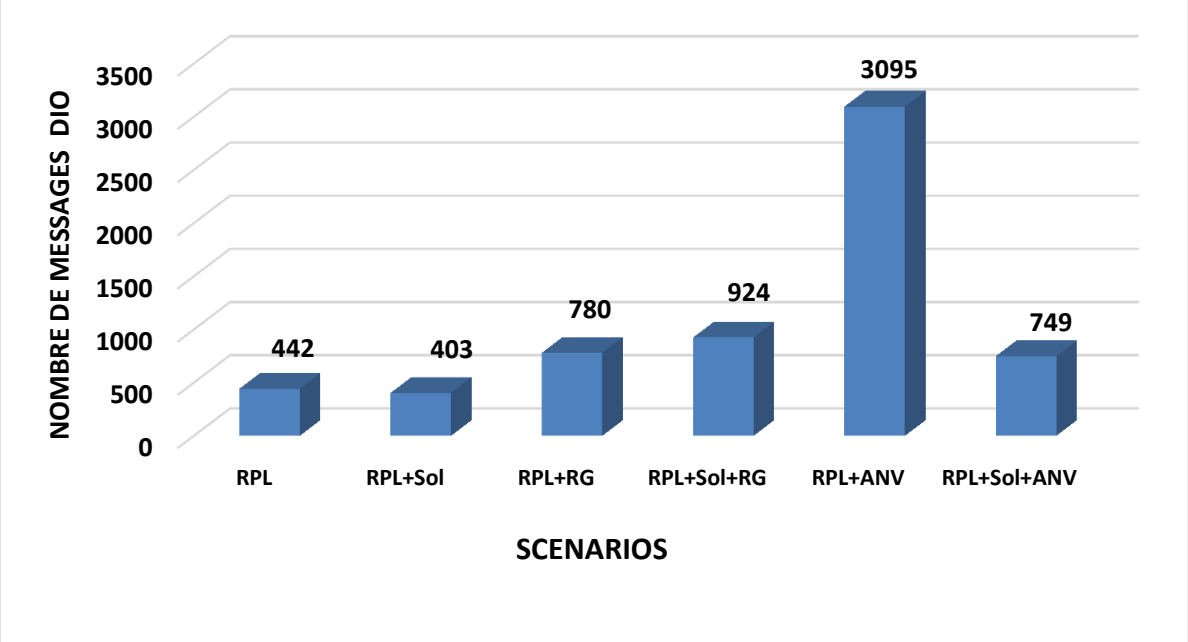


Figure V.6: Nombre de messages DIO dans les différents scénarios.

Nous avons fait une autre étude plus approfondie à propos de cette métrique pour les nœuds voisins du nœud malveillant 17. Le nombre de messages DIO sortants de chaque nœud voisin durant trois scénarios, est illustré dans la Figure V.7. Nous pouvons observer que dès que le nœud 17 se transforme en attaquant, la surcharge produite par ses voisins peut accroitre jusqu'à 3 fois pendant une courte durée (3 min). Cela provoque une augmentation significative des paquets de contrôle en cascade sur tout le réseau. En effet, nous pouvons clairement voir que notre approche a pu réduire le nombre de messages DIO transmis par les voisins.

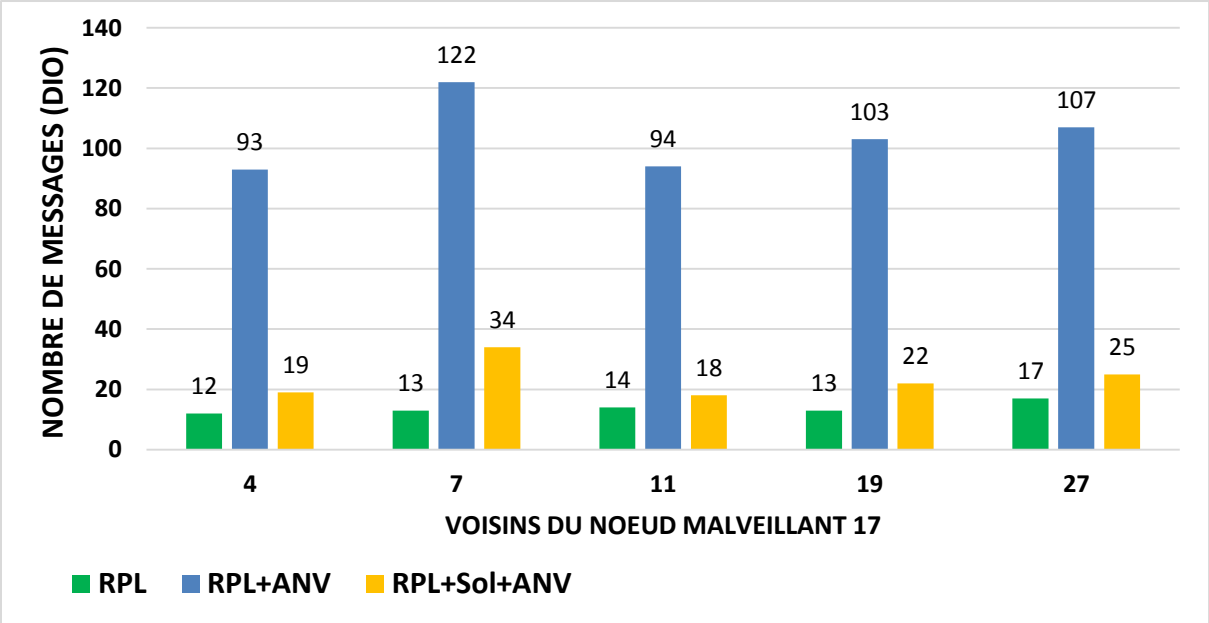


Figure V.7: Nombre de messages DIO des nœuds voisins du nœud malveillant 17.

V.6.3 Résultat de la consommation moyenne d'énergie

Les résultats de consommation moyenne d'énergie par rapport à différents scénarios sont indiqués sur la

Figure V.8. D'après le deuxième et le quatrième scénario, nous avons réalisé que les fonctions de hachage ont consommé très peu d'énergie, ce résultat est acceptable. Dans le cas de RPL sous attaque, nous avons observé que VNA peut affecter de manière significative la consommation d'énergie des nœuds et réduit leur durée de vie moyenne. Par contre, dans le cas de RPL avec solution et sous attaque, nous pouvons voir que l'énergie consommée est moins élevée, ce qui nous permet de dire que notre approche a réussi à conserver l'énergie.

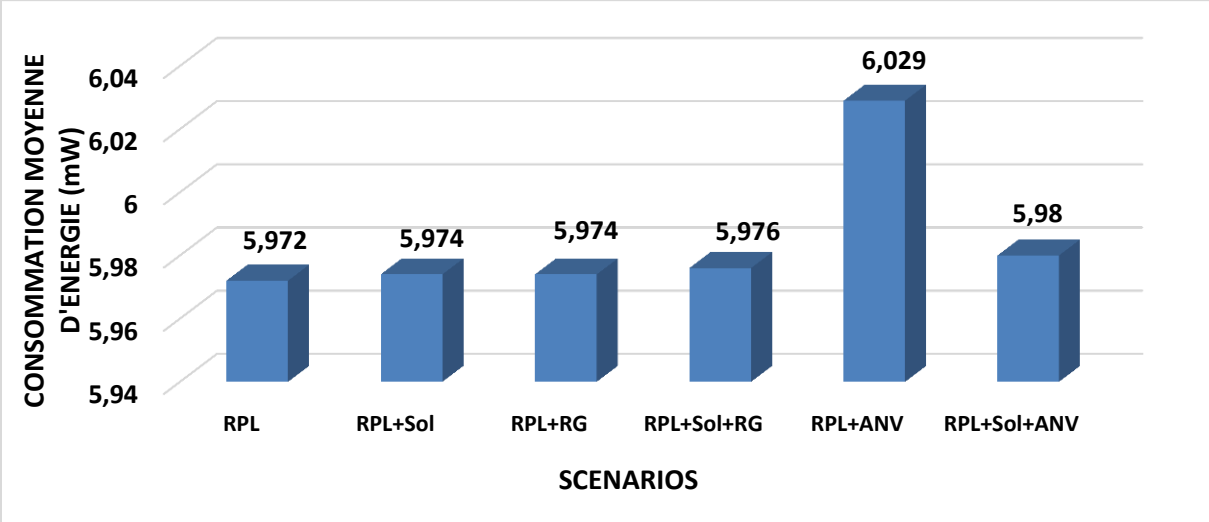


Figure V.8: La consommation d'énergie moyenne pour les différents scénarios**.**

La Figure V.9 montre également que les voisins du nœud malveillant 17 ont consommé peu d'énergie pour découvrir et arrêter l'attaquant. De même, VNA n'a pas affecté leur consommation d'énergie.

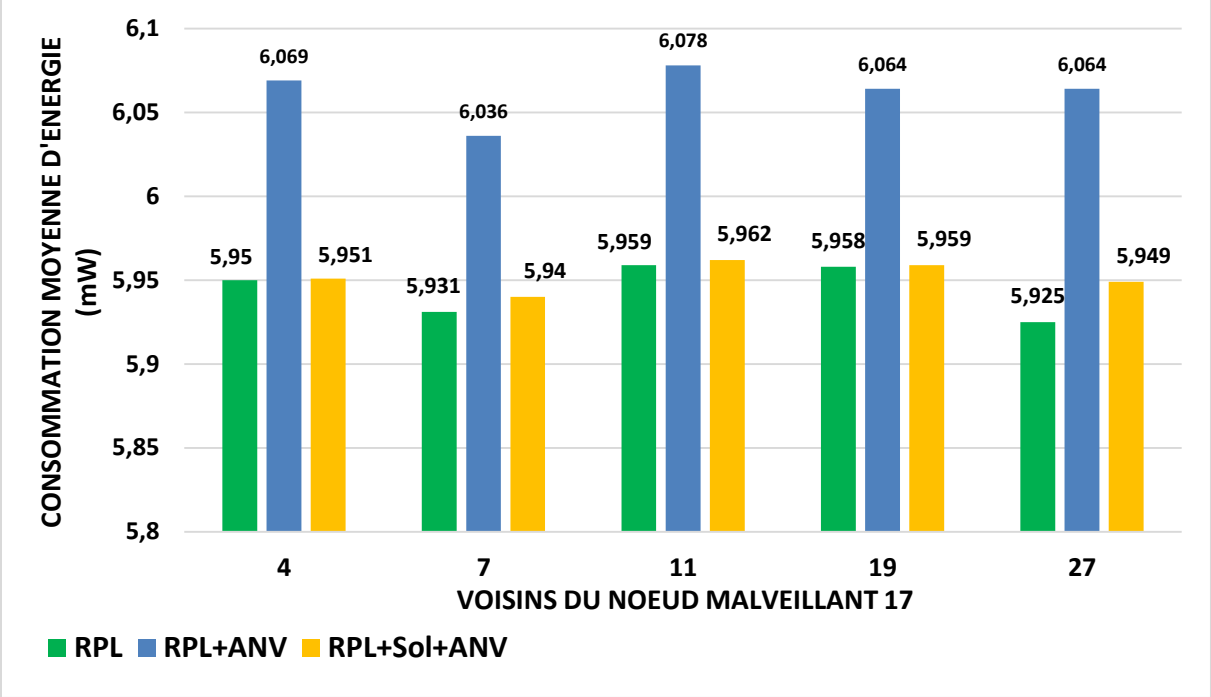


Figure V.9: La consommation moyenne d'énergie des nœuds voisins du nœud malveillant 17.

V.7 Comparaison avec les travaux connexes

Comme nous l'avons vu précédemment, il n'y a pas eu beaucoup d'études ciblant l'attaque par numéro de version, puisqu'il s'agit d'un nouveau type d'attaque par déni de service basé sur la spécification RPL. Dans le troisième chapitre, nous avons mentionné certains travaux qui ont étudié cette attaque. Toutefois, certaines études analysent l'impact de l'attaque, et d'autres sécurisent le champ de numéro de version, ou bien proposent un système de détection d'intrusion.

Dans notre travail, nous avons proposé une approche légère et efficace pour détecter et contrer l'attaque VN, où nous identifions les nœuds malveillants à l'origine de l'attaque pour les arrêter au début. Contrairement à d'autres études, leurs modules de détection ont besoin d'un certain temps pour détecter l'attaque.

D'une part, les études précédentes n'ont pas pris en compte le cas d'un nœud qui n'a pas encore migré vers la nouvelle version du DODAG, donc il peut être considéré comme un nœud malveillant, alors que nous avons tenu attention à ce cas.

D’une autre part, par rapport à VeRA, nous avons mis en œuvre notre mécanisme proposé et réalisé une évaluation en termes de la surcharge de contrôle, la consommation d'énergie et le taux de paquets livrés. Un attaquant peut exploiter la situation en envoyant fréquemment un numéro de version illégitime pour forcer le nœud à recalculer le hachage à plusieurs reprises, par conséquent, cela conduira à l'épuisement des ressources du nœud cible. En réponse, nous avons évité de recalculer le hachage de numéro de version à chaque fois que le nœud cible reçoit un message DIO avec un nouveau NV, par l’utilisation des listes, qui vise à différencier entre les nœuds intermédiaires, les nœuds suspects et les nœuds malveillants, aussi, par une vérification à deux niveaux : initiale et approfondie.

V.8 Conclusion

Dans ce chapitre, après avoir implémenté le mécanisme de la vérification du numéro de version, nous avons montré qu’il était capable d’empêcher VNA de nuire au réseau. En particulier, nous avons quantifié les performances des modules de détection à partir d’une analyse comparative entre RPL et RPL avec l’approche. Cette évaluation démontre que nous avons réussi à détecter les deux variantes de l’attaque VN ciblant RPL tout en ayant un impact minimal sur les nœuds cibles.

Conclusion et perspectives

Conclusion générale

L'Internet des objets (IoT) est un réseau de divers appareils comprenant des capteurs, des actionneurs, des appareils mobiles, etc. qui peuvent se connecter à Internet. Les réseaux à faible puissance et avec perte (LLN) sont une classe de réseaux sans fil et filaires, dans laquelle les appareils fonctionnent généralement avec des contraintes de puissance de traitement, de mémoire et de batterie. RPL a été conçu comme un protocole de routage efficace et évolutif pour les LLNs. En effet, le manque de ressources de tels réseaux les rend particulièrement vulnérables aux menaces de sécurité compromettant leur disponibilité. Cependant, Les services de sécurité RPL natifs ne protégeront pas contre un nœud interne compromis.

Par conséquent, dans ce mémoire, nous présentons un mécanisme de vérification pour nous défendre efficacement et de manière fiable contre les attaques par numéro de version DODAG dans le protocole RPL. Nous avons développé une solution algorithmique apte à satisfaire les besoins de sécurité (réparation globale) basés sur la cryptographie. Cette méthode a été réalisée à travers l'implémentation et l'exécution de certains algorithmes avec le simulateur Cooja sous ContikiOS. Nous avons évalué notre solution à travers plusieurs scénarios de simulation et l'analyser les performances selon des métriques définies. Les résultats d'expérimentation obtenus ont prouvé l'efficacité de la méthode, où nous avons eu de bons résultats sur tous les plans.

Le travail que nous avons mené, nous a permis de découvrir le domaine de l'internet des objets et d'acquérir une vaste connaissance non seulement sur ses réseaux à faible puissance et avec perte (LLN) mais aussi sur le protocole de routage pour les réseaux à faible puissance et avec perte (RPL), nous avons aussi appris à programmer sur la plateforme Contiki et à simuler des réseaux LLN sous le simulateur Cooja.

Perspectives

Plusieurs perspectives de recherche futures peuvent être envisagées sur la base de la solution présentée dans le mémoire. Tout d'abord, il serait très intéressant d'utiliser des fonctions de hachage plus sécurisé (par exemple : SHA3) avec une consommation d'énergie plus réduite.

De plus, nous souhaitons prochainement ajouter des nœuds mobiles dans la topologie de simulation, pour que l'attaquant puisse se déplacer. Par conséquent, au lieu d'envoyer des messages unicast, il faudra envoyer des messages multicast afin d'avertir tous les nœuds du réseau de la présence de nœuds malveillants.

D'une autre par, pour que le root puisse calculer et sauvegarder un nombre significatif et suffisant de hachages, nous avons l'intention de déployer la construction de la chaine de hachages au niveau du Cloud par un service Cloud.

Bibliographie

[1] Nikravan, M., Movaghar, A., & Hosseinzadeh, M. (2018). A lightweight defense approach to mitigate version number and rank attacks in low-power and lossy networks. Wireless Personal Communications, 99(2), 1035-1059.

[2] Ahmed, F., & Ko, Y. B. (2016, July). A Distributed and Cooperative Verification Mechanism to Defend against DODAG Version Number Attack in RPL. In PECCS (pp. 55-62).

[3] Kim, H. S., Ko, J., Culler, D. E., & Paek, J. (2017). Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey. IEEE Communications Surveys & Tutorials, 19(4), 2502-2525.

[4] Kharrufa, H., Al-Kashoash, H. A., & Kemp, A. H. (2019). RPL-based routing protocols in IoT applications: A Review. IEEE Sensors Journal, 19(15), 5952-5967.

[5] Ma, G., Li, X., Pei, Q., & Li, Z. (2017, October). A security routing protocol for Internet of Things based on RPL. In 2017 International Conference on Networking and Network Applications (NaNA) (pp. 209-213). IEEE.

[6] Arış, A., Yalçın, S. B. Ö., & Oktuğ, S. F. (2019). New lightweight mitigation techniques for RPL version number attacks. Ad Hoc Networks, 85, 81-91.

[7] Almusaylim, Z. A., Alhumam, A., & Jhanjhi, N. Z. (2020). Proposing a secure RPL based internet of things routing protocol: a review. Ad Hoc Networks, 101, 102096.

[8] Mayzaud, A., Badonnel, R., & Chrisment, I. (2016, October). Detecting version number attacks in RPL-based networks using a distributed monitoring architecture. In 2016 12th International Conference on Network and Service Management (CNSM) (pp. 127-135). IEEE.

[9] Ashton, K. (2009). That 'internet of things' thing. RFID journal, 22(7), 97-114.

[10] Mattern, F., & Floerkemeier, C. (2010). From the Internet of Computers to the Internet of Things. In From active data management to event-based systems and more (pp. 242-259). Springer, Berlin, Heidelberg.

[11] IoT Analytics State of the IoT Q4 2020 & 2021 perspectives, IoT & LPWA Market Tracker 2010-2025

[12] Benghozi, P. J., Bureau, S., & Massit-Folea, F. (2008). L'Internet des objets. Quels enjeux pour les Européens?.

[13] Mekriou, R., Mazari, W., & Berrah, S. (2016). Introduction à l'internet de l'objet et réalisation D'un système domotique (Doctoral dissertation, Université abderrahmane mira béjaia).

[14] Mendez Mena, D., Papapanagiotou, I., & Yang, B. (2018). Internet of things: Survey on security. Information Security Journal: A Global Perspective, 27(3), 162-182.

[15] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. IEEE communications surveys & tutorials, 17(4), 2347-2376.

[16] Patel, K. K., & Patel, S. M. (2016). Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. International journal of engineering science and computing, 6(5).

[17] Medjek, F., Tandjaoui, D., Romdhani, I., & Djedjig, N. (2018). Security Threats in the Internet of Things: RPL's Attacks and Countermeasures. In Security and privacy in smart sensor networks (pp. 147-178). IGI Global.

[18] Mayzaud, A. (2016). Monitoring and Security for the RPL-based Internet of Things (Doctoral dissertation, Université de Lorraine).

[19] Witwit, A. J., & Idrees, A. K. (2018, October). A comprehensive review for RPL routing protocol in low power and lossy networks. In International conference on new trends in information and communications technology applications (pp. 50-66). Springer, Cham.

[20] Perazzo, P., Vallati, C., Varano, D., Anastasi, G., & Dini, G. (2018, February). Implementation of a wormhole attack against a rpl network: Challenges and effects. In 2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS) (pp. 95-102). IEEE.

[21] Fondation Télécom, (2011), « L'Internet des objets, objets de l'Internet », Les cahiers de veille de la Fondation Télécom, avril, 27 pages.

[22] Kamma, P. K., Palla, C. R., Nelakuditi, U. R., & Yarrabothu, R. S. (2016, July). Design and implementation of 6LoWPAN border router. In 2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN) (pp. 1-5). IEEE.

[23] Naidu, G. A., Kumar, J., Garudachedu, V., & Ramesh, P. R. (2018, April). 6LoWPAN border router implementation for IoT devices on RaspberryPi. In Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT) (pp. 26-27).

[24] Arış, A., & Oktuğ, S. F. (2020, June). Analysis of the RPL Version Number Attack with Multiple Attackers. In 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA) (pp. 1-8). IEEE.

[25] Capossele, A., Cervo, V., De Cicco, G., & Petrioli, C. (2015, June). Security as a CoAP resource: an optimized DTLS implementation for the IoT. In 2015 IEEE international conference on communications (ICC) (pp. 549-554). IEEE.

[26] Tanganelli, G., Vallati, C., & Mingozzi, E. (2015, December). CoAPthon: Easy development of CoAP-based IoT applications with Python. In 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT) (pp. 63-68). IEEE.

[27] Randhawa, R. H., Hameed, A., & Mian, A. N. (2019). Energy efficient cross-layer approach for object security of CoAP for IoT devices. Ad Hoc Networks, 92, 101761.

[28] Joy Persial, G., Prabhu, M., & Shanmugalakshmi, R. (2011). Side channel attack-survey. Int J Adva Sci Res Rev, 1(4), 54-57.

[29] Grammatikis, P. I. R., Sarigiannidis, P. G., & Moscholios, I. D. (2019). Securing the Internet of Things: Challenges, threats and solutions. Internet of Things, 5, 41-70.

[30] Hummen, R., Hiller, J., Wirtz, H., Henze, M., Shafagh, H., & Wehrle, K. (2013, April). 6LoWPAN fragmentation attacks and mitigation mechanisms. In Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks (pp. 55-66).

[31] Arvind, S., & Narayanan, V. A. (2019, March). An overview of security in coap: Attack and analysis. In 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS) (pp. 655-660). IEEE.

[32] Pongle, P., & Chavan, G. (2015, January). A survey: Attacks on RPL and 6LoWPAN in IoT. In 2015 International conference on pervasive computing (ICPC) (pp. 1-6). IEEE.

[33] Wood, A. D., & Stankovic, J. A. (2002). Denial of service in sensor networks. computer, 35(10), 54-62.

[34] Butun, I., Österberg, P., & Song, H. (2019). Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures. IEEE Communications Surveys & Tutorials, 22(1), 616-644.

[35] Karlof, C., & Wagner, D. (2003). Secure routing in wireless sensor networks: Attacks and countermeasures. Ad hoc networks, 1(2-3), 293-315.

[36] Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., & Culler, D. E. (2002). SPINS: Security protocols for sensor networks. Wireless networks, 8(5), 521-534.

[37] Deng, J., Han, R., & Mishra, S. (2003, April). A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In Information processing in sensor networks (pp. 349-364). Springer, Berlin, Heidelberg.

[38] Di Pietro, R., Mancini, L. V., Law, Y. W., Etalle, S., & Havinga, P. (2003, October). LKHW: A directed diffusion-based secure multicast scheme for wireless sensor networks. In 2003 International Conference on Parallel Processing Workshops, 2003. Proceedings. (pp. 397-406). IEEE.

[39] Tubaishat, M., Yin, J., Panja, B., & Madria, S. (2004). A secure hierarchical model for sensor network. ACM Sigmod Record, 33(1), 7-13.

[40] Alrababah, D., Al-Shammari, E., & Alsuht, A. (2017). A Survey: Authentication Protocols for Wireless Sensor Network in the Internet of Things; Keys and Attacks. In The International Conference on New Trends in Computing Sciences (ICTCS).

[41] Griffin, P. H. (2017, March). Secure authentication on the Internet of Things. In SoutheastCon 2017 (pp. 1-5). IEEE.

[42] Kent, S., & Atkinson, R. (1998). IP authentication header.

[43] Kent, S., & Atkinson, R. (1998). IP encapsulating security payload (ESP).

[44] Raza, S., Voigt, T., & Jutvik, V. (2012, March). Lightweight IKEv2: a key management solution for both the compressed IPsec and the IEEE 802.15. 4 security. In Proceedings of the IETF workshop on smart object security (Vol. 23). Citeseer.

[45] Abdmeziem, M. R., Tandjaoui, D., & Romdhani, I. (2018). Lightweighted and energy-aware MIKEY-Ticket for e-health applications in the context of internet of things. International Journal of Sensor Networks, 26(4), 227-242.

[46] Sherasiya, T., Upadhyay, H., & Patel, H. B. (2016). A survey: Intrusion detection system for internet of things. International Journal of Computer Science and Engineering (IJCSE), 5(2), 91-98.

[47] Aljarrah, E., Yassein, M. B., & Aljawarneh, S. (2016, September). Routing protocol of low-power and lossy network: Survey and open issues. In 2016 International Conference on engineering & MIS (ICEMIS) (pp. 1-6). IEEE.

[48] IEEE Standards Association. (2016). IEEE standard for low-rate wireless networks. IEEE Std, 802, 4-2015.

[49] Vasseur, J. P., & Dunkels, A. (2010). Interconnecting smart objects with ip: The next internet. Morgan Kaufmann.

[50] Winter, T., Thubert, P., Brandt, A., Hui, J. W., Kelsey, R., Levis, P., ... & Alexander, R. K. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. rfc, 6550, 1-157.

[51] Gaddour, O., & Koubâa, A. (2012). RPL in a nutshell: A survey. Computer Networks, 56(14), 3163-3178.

[52] Guy landry DJATCH SIMO lips, "Routing Protocol for Low-Power and Lossy Networks"

[53] Kamgueu, P. O. (2017). Configuration dynamique et routage pour l'internet des objets (Doctoral dissertation, Université de Lorraine).

[54] Boubekeur, F. (2016). Les arbres couvrants de la théorie à la pratique. Algorithmes auto-stabilisants et réseaux de capteurs (Doctoral dissertation, Université Pierre et Marie Curie-Paris VI).

[55] Levis, P., Clausen, T., Hui, J., Gnawali, O., & Ko, J. (2011). Rfc6206: The trickle algorithm. Internet Engineering Task Force (IETF) Request For Comments, http://ietf. org/rfc/rfc6206. txt. 73

[56] Sahay, R., Geethakumari, G., Mitra, B., & Sahoo, I. (2018, December). Efficient framework for detection of version number attack in internet of things. In International Conference on Intelligent Systems Design and Applications (pp. 480-492). Springer, Cham.

[57] Dvir, A., & Buttyan, L. (2011, October). VeRA-version number and rank authentication in RPL. In 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems (pp. 709-714). IEEE.

[58] Kushalnagar, N., Montenegro, G., & Schumacher, C. (2007). IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals.

[59] Djedjig, N., Tandjaoui, D., & Medjek, F. (2015, July). Trust-based RPL for the Internet of Things. In 2015 IEEE Symposium on Computers and Communication (ISCC) (pp. 962-967). IEEE.

[60] Gnawali, O., & Levis, P. (2012). The minimum rank with hysteresis objective function. RFC 6719, 13.

[61] Mayzaud, A., Badonnel, R., & Chrisment, I. (2016). A Taxonomy of Attacks in RPL-based Internet of Things. International Journal of Network Security, 18(3), 459-473.

[62] Kamble, A., Malemath, V. S., & Patil, D. (2017, February). Security attacks and secure routing protocols in RPL-based Internet of Things: Survey. In 2017 International Conference on Emerging Trends & Innovation in ICT (ICEI) (pp. 33-39). IEEE.

[63] Zhang, L., Feng, G., & Qin, S. (2015, June). Intrusion detection system for RPL from routing choice intrusion. In 2015 IEEE International Conference on Communication Workshop (ICCW) (pp. 2652-2658). IEEE.

[64] Hui, J., & Vasseur, J. P. (2012). The routing protocol for low-power and lossy networks (rpl) option for carrying rpl information in data-plane datagrams. RFC 6553, March.

[65] Salman, T., & Jain, R. (2019). A survey of protocols and standards for internet of things. arXiv preprint arXiv:1903.11549.

[66] Le, A., Loo, J., Luo, Y., & Lasebae, A. (2011, October). Specification-based IDS for securing RPL from topology attacks. In 2011 IFIP Wireless Days (WD) (pp. 1-3). IEEE.

[67] Le, A., Loo, J., Lasebae, A., Aiash, M., & Luo, Y. (2012). 6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach. International Journal of Communication Systems, 25(9), 1189-1212.

[68] Chugh, K., Aboubaker, L., & Loo, J. (2012, August). Case study of a black hole attack on LoWPAN-RPL. In Proc. of the Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE), Rome, Italy (August 2012) (pp. 157-162).

[69] Kent, S., & Atkinson, R. (1998). Security architecture for the internet protocol.

[70] Kamgueu, P. O., Nataf, E., & Ndie, T. D. (2018). Survey on RPL enhancements: A focus on topology, security and mobility. Computer Communications, 120, 10-21.

[71] Djedjig, N., Tandjaoui, D., Medjek, F., & Romdhani, I. (2017, April). New trust metric for the RPL routing protocol. In 2017 8th International Conference on Information and Communication Systems (ICICS) (pp. 328-335). IEEE.

[72] Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. Journal of Network and Computer Applications, 84, 25-37.

[73] Mayzaud, A., Sehgal, A., Badonnel, R., Chrisment, I., & Schönwälder, J. (2014, June). A study of RPL DODAG version attacks. In IFIP international conference on autonomous infrastructure, management and security (pp. 92-104). Springer, Berlin, Heidelberg.

[74] Aris, A., Oktug, S. F., & Yalcin, S. B. O. (2016, April). RPL version number attacks: In-depth study. In NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium (pp. 776-779). IEEE.

[75] Perrey, H., Landsmann, M., Ugus, O., Schmidt, T. C., & Wählisch, M. (2013). TRAIL: Topology authentication in RPL. arXiv preprint arXiv:1312.0984.

[76] https://www.openwall.com/crypt/, 16 avril 2021.

[77] https://dementium2.com/securite-de-l-information/cryptage-hachage-salage-quelle-est-la-difference/, 16 avril 2021.

[78] https://github.com/BaseMax/cMD5, 04 mars 2021.

[79] https://thealgorithms.github.io/C/d7/d3b/group__hash.html, 14 avril 2021.

[80] S. Hallab and A. Jraidi. "Développement d'un Système de surveillance de L'environnement à base d'un réseau de Capteurs sans fils". L'institut supérieur de l'informatique Mahdia, Jun 2013.

[81] Samaniego, R. A. F. (2017). Wireless Sensors Networks (WSN) monitoring: application to secure interoperability (Doctoral dissertation, Université Paris-Saclay).