

Sécurisation de l'Internet des objets

Mohamed Tahar Hammi

▶ **To cite this version:**

Mohamed Tahar Hammi. Sécurisation de l'Internet des objets. Réseaux et télécommunications [cs.NI]. Université Paris Saclay (COmUE), 2018. Français. NNT : 2018SACLT006 . tel-01997261

HAL Id: tel-01997261
https://pastel.hal.science/tel-01997261

Submitted on 28 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université PARIS-SACLAY

Numéro NNT : 2018SACLT006

Thèse de doctorat

Sécurisation de l'Internet des objets

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom Paristech

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Réseaux et sécurité informatique

Thèse présentée et soutenue à Paris, le 17 Septembre 2018, par

Mohamed Tahar HAMMI

Composition du Jury :

Pr. Al Agha Khaldoun Université Paris-Sud (– Unité de recherche)	Président
Pr. Ken Chen Université Paris 13 (– Unité de recherche) ,	Rapporteur
Pr. Pascal Lorenz Université de Haute-Alsace (– Unité de recherche)	Rapporteur
Pr. Pascale Minet Inria-Paris, EVA team (– Unité de recherche)	Examinateur
Pr. Guy Pujolle Université Pierre et Marie Curie (– Unité de recherche)	Examinateur
Pr. Patrick Bellot LTCI, Télécom ParisTech (– Unité de recherche)	Directeur de thèse
Pr. Ahmed Serhrouchni LTCI, Télécom ParisTech (– Unité de recherche)	Co-Directeur de thèse

Table des matières

Table des matières iii

Liste des figures v

Liste des tableaux vii

Liste des publications 1

I Introduction générale 5

II l'IoT et la sécurité 23

1 Introduction 27

2 Les notions de base de la sécurité 31

3 Technologies de communication de l'IoT et leurs mécanismes de sécurité 35
3.1 Introduction . . . 35
3.2 Réseaux sans fil et leurs mécanisme de sécurité . . . 36
3.3 Discussion . . . 63

4 Les Blockchains 69
4.1 Introduction . . . 69
4.2 Bitcoin . . . 71
4.3 Ethereum . . . 72
4.4 Hyperledger Fabric . . . 73

III Contribution 75

5 Version 1 : Authentification des devices 79
5.1 Approche . . . 79
5.2 Évaluation et résultats . . . 83
5.3 Conclusion . . . 85

6 Version 2 : Authentification mutuelle des objets 87
6.1 Approche . . . 87
6.2 Évaluation et résultats . . . 88
6.3 Conclusion . . . 90

7 Version 3 : Sécurisation des systèmes IoT **91**
7.1 Approche . . . 91
7.2 Évaluation et résultats . . . 96
7.3 Conclusion . . . 103

8 Version 4 (BCTrust) : Système d'authentification décentralisé (*blockchain* privée) **105**
8.1 Approche . . . 105
8.2 Évaluation et résultats . . . 109
8.3 Conclusion . . . 113

9 Version 5 (*Bubbles of trust*) : Système de sécurité décentralisé (*blockchain* publique) **115**
9.1 Introduction . . . 115
9.2 Le rapport entre *BBTrust* et nos approches précédentes . . . 115
9.3 Éxigences de sécurité . . . 116
9.4 Modèle de menace . . . 117
9.5 Travaux connexes . . . 119
9.6 Approche (principe de fonctionnement de *BBTrust*) . . . 121
9.7 Évaluation et résultats . . . 126
9.8 Conclusion . . . 135

IV Conclusion et perspectives **137**

10 Références **141**

Index **153**

Liste des figures

1 Problème de migration des objets 16

3.1 Catégories des technologies de communication 35
3.2 L'architecture de *LoRaWAN* 37
3.3 Le protocole de sécurité de *LoRaWAN* 40
3.4 L'architecture du LTE 41
3.5 Une présentation simplifiée du protocole d'authentification et d'échange de clés AKA 42
3.6 Génération de paramètres et de clés 43
3.7 Un système de communication satellitaire (LEO) 46
3.8 Mécanisme d'authentification d'un système satellitaire (LEO) 46
3.9 La topologie point à multi-points du *WiMAX* 47
3.10 L'opération d'autorisation de *WiMAX* 50
3.11 Modèle en couches de l'IEEE 802.11 52
3.12 Modes d'interconnexion *Wi-Fi* 53
3.13 Processus d'envoie des messages sécurisés en WPA1 55
3.14 Processus de réception des messages sécurisés en WPA1 56
3.15 Hiérarchie des clés en WPA2 57
3.16 Une architecture d'un réseau *6LoWPAN* 58
3.17 Mécanisme d'authentification mutuelle EAP-GPSK (proposé pour *6LoWPAN*) 59
3.18 Mécanisme d'authentification mutuelle du *ZigBee* 61
3.19 Cycles d'*OCARI* 62
3.20 Topolologie d'un réseau *OCARI* 62
3.21 Une architecture simplifiée d'un réseau *OCARI* 63

4.1 Exemple simplifiée d'une *blockchain* 69

5.1 Mécanisme d'authentification (cas 1) 80
5.2 Mécanisme d'authentification (cas 2) 81
5.3 Usurpation d'identité par un objet interne 81
5.4 Mécanisme de personnalisation de clés 82
5.5 Mécanisme d'authentification (design corrigé) 82
5.6 Protocole de sécurité (version 1) 83
5.7 Format d'une trame (version 1) 83
5.8 Architecture de test 84
5.9 Temps d'exécution de l'opération d'association 84

6.1 Protocole de sécurité (version 2) 87
6.2 Mécanisme d'échange de clé de diffusion (*hidden key broadcast*) 88
6.3 Temps d'association de notre approche et celui du protocole *ZigBee* 89

7.1 Protocole de sécurité (version 3) 92
7.2 Le mécanisme de fonctionnement d'AES 93
7.3 Résultats de la validation formelle 100

8.1 Problème de migration sécurisée des devices 105
8.2 Le mécanisme de fonctionnement de *BCTrust* 106
8.3 Architecture en couche de *BCTrust* 110
8.4 Temps d'exécution totaux des associations 111

9.1 *BBTrust* et ses services de sécurité. 116
9.2 Principe de fonctionnement de *BBTrust* 123
9.3 Principe de fonctionnement du mécanisme d'authentification de *BBTrust* . 125
9.4 L'impact du traitement des messages de données sur le Laptop 130
9.5 L'impact du traitement des messages de données sur le Raspberry Pi 131

Liste des tableaux

1 Évaluation (temps moyen d'association) de la version 1 de notre approche . 11
2 Temps d'association de notre approche (version 2) 13
3 Temps d'association avec authentification 13
4 Évaluation de la version 3 de notre approche 15
5 Évaluation des performances de *BCTrust* 17
6 Résultats de l'évaluation de *bubbles of trust* (moyennes des temps et énergie consommés). 20

3.1 Les clés de confidentialités et d'intégrités utilisées entre les différentes entités LTE 43
3.2 Sécurité et technologies de communications sans fil IoT 64

7.1 Les services de sécurité de notre protocole 98
7.2 Résultats d'évaluation 103

8.1 Matériels utilisés 111
8.2 La valeur moyenne du temps d'exécution de chaque type d'association en *m*S111
8.3 L'énergie consommée par le device de chaque type d'association 112

9.1 Les critères principaux de l'évaluation de *BBtrust* 119
9.2 Résumé des les travaux connexes 120
9.3 Caractéristiques des objets 128
9.4 Moyennes (M) et Écart-Types (É-T) des résultats obtenus. 129
9.5 L'estimation du coût financier de *BBTrust* par mois 134

Liste des publications

— **BCTrust : A decentralized authentication blockchain-based mechanism**
Auteur(s) : Mohamed T. Hammi, Patrick Bellot et Ahmed Serhrouchni ;
Conférence : IEEE Wireless Communications and Networking Conference (WCNC) ;
Lieu : Barcelone, Espagne ;
Date : Avril 2018 ;
Catégorie : Article de colloque avec actes ;
Domaine(s) : Informatique ;
Langue : Anglais ;
Audience : internationale ;
État : publié ;
Département : INFRES ;
Groupe : SR.

— **Bubbles of Trust : a decentralized Blockchain-based authentication system for IoT**
Auteur(s) : Mohamed T. Hammi, Badis Hammi, Patrick Bellot et Ahmed Serhrouchni ;
Revue : Computers & Security, Elsevier ;
Date : 2018 ;
Catégorie : Article de revue avec comité de lecture ;
Domaine(s) : Informatique ;
Langue : Anglais ;
Audience : internationale ;
État : publié ;
Département : INFRES ;
Groupe : SR.

— **A Safe Communication Protocol for IoT Devices**
Auteur(s) : Mohamed T. Hammi, Erwan Livolant, Patrick Bellot, Pascale Minet et Ahmed Serhrouchni ;
Revue : Annals of Telecommunications ;
Date : 2018 ;
Catégorie : Article de revue avec comité de lecture ;
Langue : Anglais ;
Audience : non spécifiée ;
État : soumis ;
Département : INFRES ;
Groupe : SR.

— **A Lightweight IoT Security Protocol**
Auteur(s) : Mohamed T. Hammi, Erwan Livolant, Patrick Bellot, Ahmed Serhrouchni et Pascale Minet ;
Conférence : Cyber Security in Networking Conference (CSNet) ;

Lieu : Rio de Janeiro, Brésil ;
Date : Octobre 2017 ;
Catégorie : Article de colloque avec actes ;
Langue : Anglais ;
Audience : internationale ;
État : publié ;
Département : INFRES ;
Groupe : SR.

— **A lightweight mutual authentication protocol for the IoT**

Auteur(s) : Mohamed T. Hammi, Erwan Livolant, Patrick Bellot, Ahmed Serhrouchni et Pascale Minet ;
Conférence : iCatse International Conference on Mobile and Wireless Technology (ICMWT) ;
Lieu : Kuala lumpur, Malaisie ;
Date : Juin 2017 ;
Catégorie : Article de colloque avec actes ;
Langue : Anglais ;
Audience : internationale ;
État : publié ;
Département : INFRES ;
Groupe : SR.

— **MAC Sub-Layer Node Authentication in OCARI**

Auteur(s) : Mohamed T. Hammi, Erwan Livolant, Patrick Bellot, Ahmed Serhrouchni et Pascale Minet ;
Conférence : IFIP International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN) ;
Lieu : Paris, France ;
Date : Novembre 2016 ;
Catégorie : Article de colloque avec actes ;
Langue : Anglais ;
Audience : nationale ;
État : publié ;
Département : INFRES ;
Groupe : SR.

Remerciement

Tout d'abord je voudrais remercier mon directeur de thèse *Pr. Patrick Bellot* d'avoir été tout le temps disponible et attentif, ce qui m'a permis de venir à bout de cette thèse. Je le remercie pour ses bons conseils, ses bons réflexes, sa disponibilité et surtout pour tout le savoir qu'il m'a transmis.

Je tiens à exprimer, aussi, mes plus vifs remerciements à mon co-directeur de thèse *Pr. Ahmed Serhrouchni* qui m'a soutenu tout au long de mon cursus à Télécom Paristech.

Sans pour autant oublier d'exprimer tous mes remerciements à l'ensemble des membres de mon jury : *Pr. Pascal Lorenz*, *Pr. Ken Chen*, *Pr. Pascale Minet*, *Pr. Guy Pujolle* et *Pr. Al Agha Khaldoun*.

Je remercie mes chers parents : *Ali* et *Baya*, mon frère *Badis*, ma petite sœur *Feyrouz* et toute ma famille pour leur soutien sans faille et leur encouragement continu.

J'adresse toute ma gratitude à *Jean Philippe*, à *Cécile* et à *Michel* pour leurs soutiens indéfectibles, ainsi qu'à tous mes ami(e)s et toutes les personnes qui ont contribué de loin ou de près à la réalisation de ce travail.

Je ne saurais terminer sans remercier toutes les personnes formidables que j'ai rencontrées ou croisées à Télécom Paristech, à l'INRIA et à EDF avec qui j'ai vécu des moments agréables de savoir, de sympathie et de partage.

Première partie

Introduction générale

Contexte

L'Internet des Objets ou en anglais *the Internet of Things* (IoT) représente aujourd'hui une partie majeure de notre vie quotidienne. Des milliards d'objets intelligents et autonomes, à travers le monde sont connectés et communiquent entre eux. D'après [152], plus de 50 milliards d'objets seront connectés en 2020. L'union internationale des télécommunications (*International Telecommunication Union* (ITU)) définit l'IoT comme étant : « une infrastructure mondiale pour la société de l'information, permettant la fourniture des services avancés en interconnectant des objets physiques et virtuels. Elle est basée sur des technologies d'information et de communication existantes, évoluées et interopérables » [80]. Ce paradigme révolutionnaire crée une nouvelle dimension qui enlève les frontières entre le monde réel et le monde virtuel. Son succès est dû à l'évolution des équipements matériels et des technologies de communication notamment sans fil.

L'IoT est le fruit du développement et de la combinaison de différentes technologies. Il englobe presque tous les domaines de la technologie d'information (*Information Technology* (IT)) actuels tel que les villes intelligentes (*smart cities*), les systèmes machine à machine (*Machine to Machine*), les véhicules connectés, les réseaux de capteur sans fil (*Wireless Sensor Networks* (WSN)), etc. Et exploite d'autres technologies de pointe tel que le *Cloud Computing*, le *Big data*, ou encore les chaînes de blocs (*the blockchains*). Par définition, un objet est une machine physique ou virtuelle, qui doit être : (1) intelligente, donc elle doit avoir une certaine capacité de calcul et de mémorisation. (2) autonome, c'est à dire qu'elle peut faire des traitements et parfois même prendre des décisions sans une intervention humaine. (3) qui peut être connectée avec n'importe quel autre objet d'une manière flexible et transparente.

Les WSNs représentent une pièce maîtresse du succès de l'IoT. Car en utilisant des petits objets qui sont généralement limités en terme de capacité de calcul, de mémorisation et en énergie, des environnements industriels, médicaux, agricoles, et autres peuvent être couverts et gérés automatiquement.

L'IoT fournit des services avancés tels que le monitoring en temps réel des environnements, la gestion des systèmes de contrôle commande, ou encore l'automatisation totale des machines. Par conséquent, elle apporte beaucoup de gains économiques aux fournisseurs et aux entreprises en particulier, et à la société d'une manière générale. Ce qui a incité des milliers de chercheurs et développeurs à travers le monde entier à travailler sur ce domaine en essayant de développer et améliorer cette gigantesque infrastructure d'objets. En 2014 *Dell*, en partenariat avec *Intel*, a ouvert son laboratoire d'IoT à la *Silicon Valley*. Leurs services sont basées principalement sur la connectivité des équipements et le traitement des données. En 2015, *Amazon Web Services* a lancé sa plate-forme IoT, qui offre un service de streaming en temps réel et un service de stockage de données. *Cisco* à son tour a créé une gamme de services IoT tel que des services de connectivité réseau, de gestion, d'analyse et d'automatisation de données. *General Electric* est crédité d'avoir cité le terme de l'Internet des objets industriels. Son objectif est de connecter les machines de fabrication à Internet via une plate-forme de monitoring en temps réel pour éviter les temps d'arrêt imprévus. En utilisant des plate-formes IoT, basées sur le *Cloud Computing*, *Microsoft*, *IBM* et *Oracle* offrent des services d'analyse et de gestion de données et des systèmes de contrôle à distance. Aujourd'hui, il existe même des systèmes d'exploitation dédiés à l'IoT comme le cas du *Liteos* crée par l'entreprise chinoise *Huawei* qui vise à interconnecter les maisons intelligentes, les voitures, les téléphones, et d'autres objets, d'ici l'an 2020. *Samsung* en tant qu'un grand producteur de téléphones mobiles, appareils électroménagers, télévisions et robots, sera aussi l'une des sociétés leader de la connexion

des objets à Internet. La société développe une série de puces et d'applications logicielles pour ses appareils afin de permettre à ces derniers de se connecter à n'importe quelle plate-forme IoT. Ces plate-formes utilisent principalement des technologies de communication à faible consommation énergétique et des objets qui sont généralement limités en énergie, en puissance de calcul et en mémorisation (ex. capteurs, smartphones, drones, etc).

Problématique

La grande puissance de l'IoT repose sur le fait que ses objets communiquent, analysent, traitent et gèrent des données d'une manière autonome et sans aucune intervention humaine. Cependant, les problèmes liés à la sécurité freinent considérablement l'évolution et le déploiement rapide de cette haute technologie. L'usurpation d'identité, le vols d'information et la modification des données représentent un vrai danger pour ce système des systèmes. Les failles dans les mécanismes d'authentification des serrures de portes connectées, des ordinateurs ou des téléphones sont à l'origine de plusieurs cyberattaques. En 2016, une certaine Anna Senpai a crée un programme malveillant, appelé *Mirai* [81], qui permet de prendre le contrôle des objets connectés vulnérables tel que des caméras de surveillance et des routeurs, et de générer des attaques de déni de services distribuées (DDos) massives. *Mirai* transforme les objets infectés en bots, autrement dit, il les transforme en agents informatiques autonomes et intelligents contrôlés à distance. En 2017, un autre programme malveillant du nom de *BrickerBot* est apparu. Ce dernier s'attaque par force brute aux objets en utilisant des systèmes d'identification par mot de passe classique [96] afin de les tuer et donc de supprimer leurs données. La prospérité de l'IoT ne peut être réalisée que lorsque on assure une bonne sécurité aux objets et aux réseaux de communication utilisés. Il est primordial de mettre en place une politique de sécurité qui empêche tout objet malicieux ou non autorisé d'avoir accès aux systèmes IoT, de lire leurs données ou de les modifier. Pour qu'un objet ait la possibilité d'exploiter un service ou de s'associer à un réseau, il doit d'abord prouver son identité et avoir les droits d'accès nécessaires. Les objets connectés sont généralement très limités en capacité de calcul et de stockage. Ils sont également contraints par la consommation d'énergie. Dès lors, on ne peut pas employer les mécanismes de sécurité classiques tel que l'authentification avec certificats numériques ou l'utilisation des algorithmes cryptographiques asymétriques comme Rivest Shamir Adleman (RSA) ou Diffie-Hellman [91] car ils sont très coûteux, voire non supportés par les objets. De ce fait, il faut créer un nouveau mécanisme léger et robuste, qui assure les services d'authentification des objets et de protection des données, tout en étant adapté aux capacités des objets et des technologies de communication.

Contribution

Le sujet de ma thèse consiste en la création d'un système de sécurité permettant d'assurer les services d'authentification des objets connectés, d'intégrité des données échangées entres ces derniers et de confidentialité des informations. Cette approche doit prendre en considération les contraintes des objets et des technologies de communication utilisées.

Pour réaliser une telle approche, nous avons opté pour les WSNs (voir section 3.2.4 page 57) comme cas d'utilisation IoT. Ce choix est motivé par (1) le grand succès et le

fort déploiement des réseaux de capteurs sans fil dans différents secteurs (ex. industriel, environnemental, médical, militaire). Et aussi (2) par leurs évolutions et leurs développements continues.
Remarque 1 : dans ce manuscrit le mot “device” fait référence aux objets limités en ressources.
Remarque 2 : dans ce manuscrit le mot “CPAN” (voir section 3.2.4.3 page 62) fait référence aux objets non limités en ressources qui ont pour rôle principal, la gestion d’un réseau.
Remarque 3 : dans ce manuscrit le mot “objet” signifie équipement informatique intelligent qui a une capacité de calcul et de mémorisation. Il peut être un device ou un CPAN.

Nous avons utilisé une technologie appelée OCARI (voir section 3.2.4.3 page 62). Comme c’est le cas de plusieurs systèmes IoT et WSN, un réseau OCARI est composé d’un ensemble de sous réseaux, où chacun est géré par une entité principale (ex. passerelle, serveur, CPAN). Pour qu’un device puisse être joint à un réseau et échanger des données, il doit établir une phase d’association avec le CPAN du réseau.

Notre approche vise à améliorer les travaux existants, étudiés dans le chapitre 3 page 35, qui sont proposés pour sécuriser les systèmes IoT. La section 3.3 page 63 synthétise ces travaux et met au clair les avantages et inconvénients de chacun.

La conception de notre approche est passée par différentes phases (versions). D’abord, nous avons créé un protocole qui assure l’authentification du device lors de son association et qui protège l’intégrité des données échangées en mode unicast. Ensuite, ce protocole a été amélioré pour permettre l’authentification mutuelle entre le device et le CPAN, ainsi que la protection d’intégrité de tout type de paquet (unicast et broadcast). Pour compléter cette version, le service de confidentialité des données échangées a été ajouté. Après, afin de permettre une mobilité et une migration sécurisée des objets, nous avons rajouté un mécanisme d’authentification décentralisé basé sur la *blockchain* privée. Enfin, étant donné le manque d’évolutivité et de flexibilité dans la *blockchain* privée et afin de rendre notre approche plus performante, nous avons conçu un autre mécanisme d’authentification totalement décentralisé basé sur la *blockchain* publique.

Version 1 : Authentification des devices

Dans un système IoT, plus particulièrement dans un système WSN, pour qu’un objet puisse communiquer et échanger des données avec un autre objet, une phase d’association doit être établie entre ces derniers. Dans le cas d’OCARI, quand il n’y a aucun mécanisme de sécurité, cette association se réalise par l’échange de deux messages entre le device et le CPAN. Le premier représente une “*association request*” (envoyé par le device) et le deuxième une “*association response*” (envoyé par le CPAN). Une fois que l’association est réalisée, les deux objets créent une session non sécurisée pour échanger des informations.

Tout comme pour la plupart des protocoles de communication, OCARI ne protège pas les identités des objets ni l’intégrité des données échangées. Autrement dit, un device malicieux γ peut usurper l’identité d’un autre device légitime α et s’associer à son réseau géré par un CPAN β. Ainsi, γ peut falsifier les données échangées durant la session de communication. De ce fait, il est nécessaire de mettre en œuvre des mécanismes de sécurité afin de protéger les systèmes et les réseaux IoT.

Lors de la conception de notre approche, nous avons d'abord réalisé une première version qui permet d'assurer l'authentification d'un device pendant la phase d'association. Le mécanisme d'authentification utilisé est basé sur le One Time Password (OTP), mot de passe à usage unique, défini dans la RFC 2289 [66] et la RFC 4226 [120], ainsi que sur le principe de challenge/réponse décrit par la RFC1994 [145]. En effet, un OTP est un mot de passe qui n'est valide que pendant une seule opération d'authentification. La raison pour laquelle, il représente un mécanisme d'authentification très résistant contre les attaques de rejeu et les attaques de cryptanalyse décrites dans la section 2 page 33. Les mots de passe à usage unique peuvent être utilisés en mode synchrone ou asynchrone.

Le mode synchrone est basé sur (1) un secret partagé (ex. clé symétrique) entre deux objets pour prouver l'identité d'une ou des deux entités communicantes et (2) un paramètre pré-partagé tel que le temps ou un compteur qui change d'une manière synchronisée après chaque opération d'authentification.

Quant au mode asynchrone, il est basé sur (1) un secret partagé et (2) un nombre aléatoire appelé aussi défi (challenge) envoyé par l'authenticateur.

Nous avons utilisé le mode asynchrone car contrairement au mode synchrone, il n'exige aucune approbation préalable entre les entités communicantes (ex. compteur). Et à cause de l'instabilité de la plupart des réseaux sans fil, si un message est perdu alors ceci engendrera un problème de synchronisation car les valeurs des compteurs deviennent différentes. De plus, beaucoup de technologies sans fil ne supportent pas le temps absolu.

Notre protocole d'authentification rajoute l'échange de deux messages en plus, qui sont "*authentication request*" et "*authentication response*". Par conséquent, afin qu'un device s'associe d'une manière sécurisée, d'abord, il envoie une *association request* au CPAN. Ce denier répond par une *authentication request* contenant un challenge. Ensuite, via une fonction cryptographique décrite par la RFC 4226 [120], que nous avons adaptée pour le mode asynchrone, le device calcule un OTP en utilisant une clé secrète partagée et le challenge reçu. Puis il dérive et stocke une clé de session k_u qui sera utilisée pour sécuriser les données de session échangées en mode unicast. Après quoi, il envoie l'OTP via une *authentication response*. Enfin, à la réception du message, le CPAN calcule un autre OTP en se basant sur les mêmes paramètres et fonctions utilisés par le device, puis compare les deux OTPs. S'ils sont identiques alors ceci prouve l'identité du device. Ainsi, l'authentification de ce dernier est réussie et son association est confirmée par une *association response*. Après l'authentification du device, une clé de session symétrique k_u identique à celle stockée dans le device est générée au niveau du CPAN afin d'assurer l'intégrité des messages en mode unicast.

Le partage de la clé principale, utilisée pour l'authentification des objets et la génération des clés de session, représente un défi pour les concepteurs des protocoles de sécurité. Car pour avoir un protocole sûr, cette clé doit être unique et personnalisée pour chaque device. Cette personnalisation ne doit pas influencer négativement les performances du réseau et le bon fonctionnement du CPAN. De ce fait, nous avons crée un mécanisme de gestion de clés appelé *"Personnalisation"* de clés (voir section 5.1.1 page 80). Ce dernier représente une méthode sécurisée, flexible et optimale de distribution des clés pré-partagées qui protège les objets contre les attaques d'usurpation interne. En effet, le principe de ce mécanisme repose sur le fait d'installer une clé initiale k_i au niveau du CPAN et d'en dériver une clé personnalisée k_d pour chaque device. k_d est calculée à partir de k_i et l'identifiant unique (UI) du device en utilisant une fonction de hachage (HMAC-SHA256 [126]). Cette dernière représente une fonction à sens unique qui empêche l'obtention des paramètres en entrée (ex. clé secrète) à partir du résultat. Elle donne des ré-

sultats totalement différents à partir de différentes données en entrée. Cette personnalisation offre beaucoup d'avantages :

— le CPAN n'a pas besoin de stocker la clé k_d de de chaque device appartenant à son réseau, mais plutôt de la déduire automatiquement grâce à sa clé k_i et à l'UI du device demandant l'association ;

— quand on rajoute un nouveau device muni d'une k_d, le CPAN n'a pas besoin d'être mis à jour. Ce qui permet une grande transparence et flexibilité lors de l'ajout des nouveaux devices ;

— contrairement à certaines approches qui proposent une authentification basée sur une clé de diffusion (voir section 3.2.4.2 page 60), le fait que chaque device possède sa propre clé, qui est liée à son identité, protège le système contre les problèmes d'usurpation d'identité interne (voir section 5.1.1 page 80).

Enfin, une fois que l'association se termine, un canal sécurisé se crée entre les entités communicantes. Ce canal permet d'assurer l'intégrité des données en signant tous les messages à l'aide de la clé k_u. La signature représente les n premiers $octets$ du HMAC-SHA256 [126] de la trame à envoyer. Due à la limitation de la taille de la charge utile de la trame dans les réseaux IoT, il est préférable que n ne dépasse pas 16 $octets$.

De cette façon, si un message est modifié ou falsifié, le système peut automatiquement détecter le problème.

Pour l'évaluation de l'approche, nous nous sommes intéressé principalement au temps d'exécution du mécanisme d'authentification, établi pendant la phase d'association. Une étude plus complète est réalisée lors de l'évaluation de la version 3, voir section 7.2 page 96.

Nous avons réalisé une implémentation en langage C appliquée sur la plateforme d'OCARI (une technologie de communication basée sur le standard 802.15.4) et déployé sur des vrais capteurs. Ces derniers représentent des modules Atmel Dresden Elektronik deRFsam3 23M10-R3, ayant 48 ko de RAM, 256 ko de ROM et un processeur Cortex-M3. Nous avons mis en place une architecture composée de 3 objets qui représentent un CPAN et deux devices. Pendant la phase d'association, le premier device s'associe directement au CPAN (association à 1 saut), tandis que le deuxième s'associe au CPAN via le premier device (2 sauts). Le Tableau 1 présente les résultats obtenus de 10 tests.

	1 saut	**2 sauts**
sans authentification	0.524 (mS)	34.508 (mS)
avec authentification	35.535 (mS)	87.194 (mS)

TABLEAU 1 – Évaluation (temps moyen d'association) de la version 1 de notre approche

Comme expliqué dans la section 5.2 page 83, le temps d'association est très satisfaisant et l'augmentation du temps causée par l'authentification est due principalement aux messages supplémentaires et de manière insignifiante au temps de traitement.

Bien que cette version assure un service d'authentification léger et robuste, notamment contre les attaques de rejeu et de cryptanalyse, et protège l'intégrité des données, elle reste très basique. En effet, elle ne permet pas l'authentification du CPAN, par conséquent, il est possible qu'un utilisateur s'associe à un réseau malicieux (géré par un CPAN

malicieux). De plus, le service d'intégrité des données concerne uniquement les messages échangés en mode unicast.

Version 2 : Authentification mutuelle des objets

Pour améliorer notre approche et avoir de meilleurs performances, nous avons créé une deuxième version du protocole de sécurité. Cette dernière rajoute l'authentification du réseau. Autrement dit, elle assure une authentification mutuelle entre le device et le CPAN. En outre, elle améliore la fonction de génération de clés de session, et fournit un mécanisme sécurisé d'échange de clé de broadcast (k_b) appelé hidden key broadcast, voir chapitre 6 page 87.

hidden key broadcast est basé sur des opérations de hachage (HMAC-SHA256) et de ou exclusif ($\oplus$). Le but derrière la création d'un nouveau mécanisme d'échange de clé basé sur les mêmes algorithmes que ceux utilisés dans notre approche (au lieu de rajouter des algorithmes de chiffrement) est de :

— créer une version très légère du protocole qui utilise le minimum possible d'algorithmes ; en effet, plus le code source est petit, plus il est intégrable au niveau des objets limités en ressources ;

— l'authentification du CPAN est faite grâce à un deuxième OTP (otp_2) ; en plus de l'authentification, otp_2 permet d'assurer l'intégrité de la clé k_b qui est envoyée, avec otp_2, via une *association response*. En effet, sachant que le challenge est un paramètre aléatoire/pseudo-aléatoire qui ne peut être utilisé que pour calculer un seul OTP et qu'il ne peut pas se répéter pendant une longue période indéfinie, otp_2, pour des raisons d'optimisation, exploite une variable appelée "*hiddenKeyBroadcast*" générée grâce au mécanisme hidden key broadcast. Cette variable est calculée à partir d'une $signature$ qui est générée à son tour à partir d'un hach de k_u et du premier OTP envoyé par le device (otp_1).

Notre approche est fiable et permet de protéger les informations secrètes (clés) des objets. (1) Si un attaquant externe intercepte toutes les informations ($challenge$, $otp1$, $otp2$ et $hiddenKeyBroadcast$), il ne peut avoir ou falsifier aucune information car il n'a pas le couple (k_d, k_b) ni le couple (k_u, k_b). Et (2) pour un attaquant interne qui possède k_b en plus de toutes les informations échangées, il ne peut pas non plus obtenir les données secrètes d'autres objets. Car, lorsqu'il essaie d'avoir k_u d'un autre device, il calcule le $\oplus$ entre k_b et $hiddenKeyBroadcast$ pour extraire la $signature$, et comme cette dernière est générée à partir d'une fonction à sens unique, même en utilisant $otp1$, l'attaquant ne peut pas obtenir k_u.

Pour finir, le canal de communication qui utilise désormais k_u et k_b, assure via HMAC-SHA256 l'intégrité de tout type de transmission (unicast et broadcast).

Pour l'évaluation de cette version, en utilisant la même architecture et le même hardware présenté ci-dessus, nous avons mesuré les temps d'exécution de la phase d'association sans et avec authentification à 1 saut et à 2 sauts. Une évaluation plus complète est réalisée dans la version 3, voir section 7.2 page 96.

Le Tableau 2 page ci-contre présente une comparaison entre les moyennes de temps d'association en utilisant les modes avec authentification et sans authentification de la version 2. Tandis que le Tableau 3 page suivante présente les résultats obtenus pour les temps moyens d'une association qui utilise la version 1 de notre protocole et une association qui utilise la version 2.

Due à la particularité de la sous-couche MAC d'OCARI, nous ne pouvons pas comparer notre temps d'association sécurisée avec les technologies 802.15.4 qui existent et moins encore avec d'autres technologies basées sur d'autres standards car elles sont très différentes. Néanmoins, à titre indicatif, le Tableau 3 présente également le temps d'association avec authentification (handshake) du protocole *Datagram Transport Layer Security* (DTLS) appliqué sur un système WSN et déployé sur un hardware similaire [92]. D'après la RFC6347 [137], le DTLS est un protocole qui assure la confidentialité et l'intégrité des protocole de communication.

Nous avons présenté aussi le temps d'association avec authentification obtenu lors d'une évaluation du protocole *ZigBee* (expliqué dans la section 3.2.4.2) qui est déployé sur un hardware similaire [17].

	1 saut	**2 sauts**
sans authentification	0.524 (mS)	34.508 (mS)
avec authentification	37.504 (mS)	45.876 (mS)

TABLEAU 2 – Temps d'association de notre approche (version 2)

	1 saut	**2 sauts**
version 1	35.535 (mS)	87.194 (mS)
version 2	37.504 (mS)	45.876 (mS)
DTLS handshake	4000 (mS)	- (mS)
ZigBee	500 (mS)	- (mS)

TABLEAU 3 – Temps d'association avec authentification

D'après ces résultats, nous prouvons que notre protocole est optimal. En effet, on constate une légère augmentation du temps d'exécution pour le mode avec authentification à 1 saut de cette version par rapport à celui de la version 1. Et bien que la version 2 rajoute plus d'options, le temps d'exécution du mode avec authentification à 2 sauts a diminué dans cette version par rapport à la première. Ceci est dû aux optimisations du code source que nous avons établi.

En utilisant une clé RSA d'une taille de 2048 $bits$, la phase handshake du DTLS prend 4000 mS. Cette longue durée s'explique par le grand nombre de messages échangés pendant cette phase (6 messages obligatoires), et par l'utilisation des algorithmes asymétriques qui consomment généralement beaucoup de temps.

Enfin, en se référant au temps d'association du protocole *ZigBee* qui est considéré comme l'un des protocoles qui se rapproche le plus d'OCARI, on constate que les résultats obtenus de notre approche sont satisfaisants. En effet, même si on ne peut pas comparer notre temps d'association avec celui du *ZigBee*, on remarque que l'opération d'association de notre approche est très rapide.

Cette version améliore considérablement sa précédente. Cependant, la confidentialité des données n'est pas assurée. C'est la raison pour laquelle nous avons crée une troisième version.

Version 3 : Sécurisation des systèmes IoT

La troisième version rajoute le service de confidentialité des données. Elle utilise l'algorithme de chiffrement *Rijndael* connu sous le nom d'*Advanced Encryption Standard* (AES). Afin d'assurer à la fois la confidentialité et l'intégrité des données, nous utilisons au choix le *Galois/ Counter Mode* (GCM) et le Counter with CBC-MAC (CCM) comme mode d'opération.

Comme expliqué ci-dessus Partie I page 12, une fois qu'une association sécurisée entre deux objets s'effectue, un canal de communication se créé. Avec cette version tous les messages échangés sont chiffrés et authentifiés au même temps grâce à l'algorithme AES-GCM/CCM. Par conséquent, l'utilisation de HMAC-SHA256 pour assurer l'intégrité n'est plus nécessaire.

AES est un algorithme de chiffrement par bloc. Il utilise une même clé symétrique pour chiffrer/déchiffrer des blocs de données. Chaque bloc passe par plusieurs opérations de permutation, de décalage et de xor. Pour plus de détails voir chapitre 7 page 91.

En déployant le mode GCM, une entité voulant échanger des données doit les chiffrer puis les signer. Cette opération utilise k_u pour le mode unicast, ou k_b pour le mode broadcast. Ces deux clés sont générées pendant la phase d'association, expliquée ci-dessus. Avec ce mode d'opération, lors d'une opération de chiffrement authentifié, on découpe chaque message en clair (*plaintext*) P en n blocs $(P_1, \cdots, P_n)$. Ensuite comme donnée d'entrée (*input*), on met les blocs obtenus, une clé de chiffrement (k_u ou k_b), une donnée additionnelle authentifiée A qui peut être n'importe quelle donnée ajoutée pour renforcer l'algorithme de chiffrement. La donnée additionnelle authentifiée A est d'une taille de 16 $octets$ qui doit être partagée entre le CPAN et le device. De ce fait, dans notre approche elle représente une séquence créée à partir de la duplication de l'identifiant unique (8 $octets$) du device. L'opération de chiffrement nécessite également un vecteur d'initialisation IV. Nous avons rajouté un compteur qui sert à modifier la valeur de l'IV afin de protéger le protocole contre les attaques de cryptanalyse. Comme résultat (*output*), on obtient une donnée appelée *ciphertext* C composée de n blocs chiffrés $(C_1, \cdots, C_n)$ et un tag d'intégrité T. Ce dernier sert à assurer l'intégrité des blocs. Enfin, on associe le *ciphertext* avec le tag T.

Pour le déchiffrement, nous avons besoin du *ciphertext* C, le tag reçu T, l'IV et la donnée additionnelle authentifiée A. En tant que output, nous obtenons le *plaintext* P et un tag T' qui sera comparé à T afin de vérifier l'intégrité des données. Si l'opération de vérification échoue alors le bloc sera rejeté.

Le mode CCM (ou CCM*) assure également les services d'intégrité et de confidentialité à la fois. Mais contrairement au GCM, il authentifie puis chiffre les données. Il nécessite une clé, un nombre aléatoire N, le *plaintext* P, la donnée additionnelle authentifiée A et un compteur. Comme résultat, on obtient les données chiffrées C suivis par U. Ce dernier est une valeur d'une taille de M $octets$ (ex. 16 $octets$) qui nous permet d'avoir le tag T : $T \leftarrow first(M, S_0) \oplus U$. La fonction *first()* permet d'avoir les M premiers $octets$ de S_0.

L'opération de déchiffrement nécessite la clé symétrique (k_u ou k_b), N, la donnée additionnelle authentifiée A et les données chiffrées et authentifiées (C concaténé à U). On déchiffre C et on recalcule T' afin de le comparer avec T. Si T n'est pas valide alors l'opération de déchiffrement authentifié échoue.

Avec cette version, notre protocole assure tous les services de sécurité de base. Pour prouver la sûreté, l'efficacité et la robustesse de notre approche, nous avons établi une validation formelle en utilisant *Scyther*. Ce dernier représente à la fois un langage et un outil d'analyse, qui permet de détecter les failles dans les protocoles de sécurité. En *Scyther*, un protocole est définie par des rôles joués par des agents qui exécutent une suite d'événements. Pour vérifier la validité du protocole, on définit également les objectifs de la validation formelle. Ces objectifs concernent principalement la sûreté des mécanismes d'authentification, d'échange de clé et de confidentialité des données.

Nous avons effectué 100 itérations (exécutions) de notre code formel. Les résultats de la validation ont prouvé la sûreté de notre protocole.

Après avoir prouvé formellement la validité de notre protocole, nous avons évalué la quantité d'énergie consommée pendant la phase d'association. Dans cette étude, nous n'avons pas évalué la consommation énergétique du canal de communication car les standards AES-GCM et AES-CCM sont connus par leurs optimalités en temps et en consommation d'énergie et ils sont fortement recommandés par le NIST [51][52].

Le Tableau 4 présente le temps écoulé pendant l'opération de communication, le temps de traitement, la puissance et la quantité moyenne en énergie consommée par un objet pendant son association. Nous avons déployé le même hardware qui a été utilisé pour évaluer la version 1 et 2 du protocole (voir I page 9).

	sans sécurité	**avec sécurité**
temps de communication	0.524 (*m*S)	34.508 (*m*S)
temps de traitement	0 (*m*S)	2.996 (*m*S)
puissance totale consommée	634.07 (*m*W)	1273.88 (*m*W)
énergie totale consommée	0.332 (*m*J)	43.779 (*m*J)

TABLEAU 4 – Évaluation de la version 3 de notre approche

Le *temps de traitement* est lié aux traitements rajoutés par le protocole de sécurité. Une comparaison avec un travail réalisé par [93] concernant l'énergie consommée par un device utilisant une autre approche de sécurité est établie dans la section 7.2.5 page 101.

D'après ces résultats, on peut constater que l'échange de message est ce qui consomme le plus de temps et d'énergie. Et par rapport à une association de base (sans sécurité), notre protocole consomme uniquement 43.447 *m*J en plus. Ainsi, nous avons montré que notre protocole est économique en énergie et adapté pour les systèmes IoT.

Version 4 (BCTrust) : Système d'authentification décentralisé (*blockchain* privée)

Il est vrai qu'on peut considérer que la version 3 comme étant une version complète. Cependant elle ne fournit aucun mécanisme sécurisé permettant la migration transparente des objets d'un réseau vers un autre. Autrement dit, comme illustré par la Figure 1 page suivante, un device α appartenant à un réseau géré par un CPAN β ne peut pas migrer et se joindre à un réseau géré par un autre CPAN γ.

Pour résoudre ce problème, nous avons conçu un mécanisme d'authentification décentralisé basé sur la *blockchain* privée, appelé *BCTrust*. Ce mécanisme se base sur le principe de « *l'ami de mon ami est mon ami* ». Autrement dit, si un device α a été déjà authentifié par un CPAN légitime β alors tous les autres CPANs qui font confiance à β feront confiance à α.

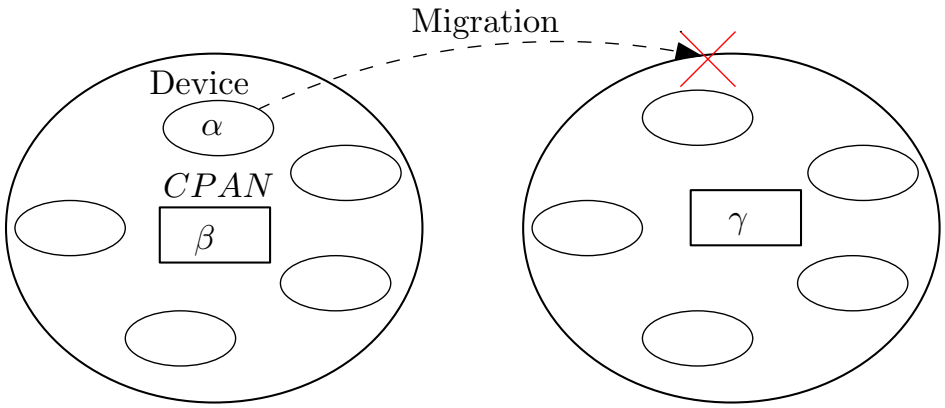


FIGURE 1 – Problème de migration des objets

Lors de la première association de α à son réseau principal, il s'authentifie en utilisant les mécanismes décrits dans la version 3. Si l'authentification est réussie alors β envoie une transaction indiquant que α est légitime et qu'il a été authentifié par β. Lorsque α migre vers un autre réseau géré par un CPAN γ, ce dernier vérifie dans la *blockchain* si α était déjà authentifié auparavant par un autre CPAN. Quand γ trouve l'information concernant la dernière authentification de α par β, il demande à β le transfert des dernières clés de session utilisées via un canal sécurisé. Une fois qu'une opération de chiffrement/déchiffrement entre α et γ s'effectue avec succès, γ envoie à son tour une autre transaction indiquant que c'est lui le dernier CPAN qui a authentifié α. Si α décide encore de quitter le réseau de γ et de migrer vers un $3^{ème}$ réseau géré par δ alors on effectue la même opération que celle faite avec γ et les dernières clés de session utilisées entre α et γ sont transférées vers δ. Il faut noter que les clés de session peuvent être mises à jour par le CPAN en cours.

Pour réaliser ce mécanisme, nous avons opté pour la *blockchain* d'*Ethereum*. Ce choix est motivé par le fait que :

— *Ethereum* possède le deuxième plus grand *ledger* après *Bitcoin* [123] ;

— il assure des transactions sécurisées ;

— il est suivit par une grande communauté ;

— *Ethereum* peut être déployée comme une *blockchain* privée, qui est une copie indépendante de la *blockchain* publique d'*Ethereum*. Dans le cas d'une *blockchain* privée, la validation des transactions est limitée à certains objets connus par la *blockchain* et l'utilisation d'un mécanisme de validation tel que la *Proof of Work* (PoW) ou la *Proof of Stake* (PoS) qui représente un défi mathématique difficile à calculer mais facile à vérifier (voir section 4.1.1 page 70), n'est pas nécessaire, par conséquent, le temps de validation des transactions est très court ;

— *Ethereum* permet la création des applications décentralisées (*dApp*) via des programmes courts appelés *smart contracts* ;

— il est suivit par une grande communauté ;

— les développeurs d'*Ethereum* et sa communauté travaillent sur la régulation et la stabilisation des montants des frais liés à l'utilisation des *smart constracts*[1].

1. https://smartereum.com/6777/buterin-expresses-concern-over-stabilizing-ethereum/

Pour évaluer notre approche, nous avons effectué une démonstration sur une architecture (voir section 8.2.1 page 109) représentant un réseau d'objets. Pour réaliser cette expérience, d'abord, nous avons utilisé *TestRPC* (voir section 8.2.1 page 109) pour déployer une *blockchain Ethereum* en local munie d'un client qui interagit avec.

Le code source des objets était développé en langage C . De ce fait, nous avons créé une interface qui permet aux programmes écrits en C d'interagir avec la *blockchain* par l'intermédiaire du client *TestRPC*. Autrement dit, notre interface encode/décode les données entre les programmes écrits en C et *Ethereum*.

Après avoir préparé les outils et les programmes nécessaires, nous avons créé le *smart contract* de notre approche en langage Solidity [57]. Pour finir, nous avons rajouté des restrictions concernant les objets ayant la possibilité d'écrire sur la *blockchain*. Ainsi, nous avons établi une liste des adresses des objets ayant le droit d'établir des transactions. Et donc rendre la *blockchain* privée.

Pour l'évaluation de cette version, nous nous intéressons uniquement au temps d'exécution et à la consommation d'énergie des devices car on considère que les CPANs sont illimités en ressources. De ce fait, nous avons mesuré le temps d'exécution et calculé la consommation d'énergie d'un device pour : (1) une association non sécurisée "*without security*". (2) Une "*classical association*" qui représente la $1^{ère}$ association d'un device à son réseau principal et qui utilise la version 3 de notre protocole. (3) Une "*smart association*"qui représente la $n^{ème}$ association (n ≥ 2) d'un device à son réseau principal (sans migration), et qui utilise le mécanisme d'association fourni par *BCTrust*. Et enfin, (4) une "*smart migration*" qui représente l'association d'un device pendant une opération de migration, et qui utilise également le mécanisme d'association fourni par *BCTrust*.

Le Tableau 5 présente les résultats obtenus.

	Temps	**Énergie**
without security	0.52 (*m*S)	0.33 (*m*J)
classical association	37,75 (*m*S)	43,779 (*m*J)
smart association	0.77 (*m*S)	0.33 (*m*J)
smart migration	14.26 (*m*S)	0.345 (*m*J)

TABLEAU 5 – Évaluation des performances de *BCTrust*

Les résultats montrent que cette version est encore plus optimale que sa précédente car elle diminue le nombre de message et la quantité de traitement effectué par le device lors de la phase d'association. Elle permet une migration sécurisée, transparente et flexible des objets, et en exploitant la force de la *blockchain* elle fournit des services de disponibilité, d'intégrité d'information, de traçage (log) et de non répudiation.

L'utilisation d'une *blockchain* privée permet d'établir des transactions gratuitement, avec un temps de validation très rapide. Cependant le fait que les objets ayant la possibilité d'écrire sur la *blockchain* -qui dans *BCTrust* représentent les CPANs- doivent être connus à l'avance par la *blockchain* privée, diminue considérablement l'évolutivité du réseau.

Version 5 (*Bubbles of trust*) : Système de sécurité décentralisé (*blockchain* publique)

Pour résoudre le problème de l'évolutivité rencontré dans *BCTrust*, nous avons réalisé une autre alternative basée sur la *blockchain* publique, appelée *bubbles of trust* ou *BBTrust*. Contrairement à *BCTrust* qui dépend des versions précédentes de notre protocole, *bubbles of trust* représente un protocole totalement indépendant qui peut être déployé de deux façons :

— il peut représenter à lui seul un protocole complet qui assure un service d'authentification mutuelle complètement décentralisé et évolutif. Il assure également l'intégrité, la confidentialité, le traçage et la disponibilité des données. Dans ce cas les objets le déployant doivent avoir une capacité acceptable en ressources ;

— il peut également compléter les versions 1,2,3 et 4, ainsi, on obtient un protocole de sécurité complet qui offre un service d'authentification mutuelle complètement décentralisé et évolutif permettant une migration transparente et sécurisée des objets. Ce deuxième cas de déploiement est celui qui est appliqué pour notre approche car en utilisant le minimum possible de transaction, il ne consomme pas beaucoup de ressources.

bubbles of trust permet de créer des zones virtuelles sécurisées (bulles) dans un environnement d'une *blockchain* publique couvrant des objets IoT. Chaque zone assure à ses membres un service d'identification et d'authentification, ainsi qu'un service de protection de données.

Une bulle est protégée et n'est accessible que par les objets membres qui communiquent uniquement entre eux. Car on considère que tout objet n'appartenant pas à la bulle comme étant malicieux.

Nous avons développé notre système de sécurité en utilisant une *blockchain* publique qui implémente des *smart contrats*. Toutes les communications sont sous forme de transactions validées. Autrement dit, pour qu'un objet α envoie un message à un autre objet β alors il doit d'abord envoyer le message à la *blockchain* (*bc*) sous forme d'une transaction. *bc* vérifie la légitimité de α puis vérifie l'exactitude de la transaction avant de la valider. Et enfin, β peut lire le message.

Remarque : *BBTrust* peut être appliquée sur un grand nombre de scénarios IoT tel que les maisons/villes intelligentes, les réseaux de capteurs sans fil et les usines automatisées.

L'IoT couvre différents domaines, par conséquent elle déploie des objets très variés utilisant des technologies différentes.

Afin de sécuriser un système informatique -composé de plusieurs objets connectés- par notre approche, d'abord, on désigne un objet comme étant maître (*Master*) et on lui attribue une pair de clés privée/publique (*privM*/*pubM*). Le maître représente l'autorité de certification de l'ensemble des objets (bulle) du système.

Ensuite on configure le reste des objets qu'on appelle désormais disciples (*Followers*) par des pairs de clés personnalisées privée/publique (*privF*/*pubF*).

Pour chaque *Follower*, on fournit également un certificat léger que nous avons appelé *ticket*. Ce dernier représente une structure de données qui permet d'associer l'identité de l'objet à sa paire de clés. Un *ticket* est composé de : (1) un identifiant de la bulle -choisi par le *Master*- nommé *GrpID*, (2) un *ObjID*, qui est l'identifiant du *Follower* dans la bulle,

(3) *PubAddr*, qui est l'adresse publique du *Follower*. Cette adresse est obtenue à partir de *pubF*. Et (4) une *Signature* du *GrpID*, *ObjID* et *PubAddr* calculée en utilisant la clé privée du *Master* de la bulle.

Une fois que le groupe d'objets (bulle) est préparé, on crée la bulle au niveau de la *blockchain*. En effet, le *Master* envoie une transaction qui contient son *ObjID* et le *GrpID* choisi. La *blockchain* vérifie l'intégrité de la transaction, puis l'unicité des paramètres (*GrpID*, *ObjID* et *PubAddr*), si la transaction est valide, alors la bulle est crée.

Après, les *Followers* à leurs tours, envoient des transactions afin de se joindre à leurs bulles respectives. Lors d'une demande d'association à une bulle, en appliquant les règles du *smart contract*, la *blockchain* vérifie que la bulle existe déjà, que l'identifiant du *Follower* est unique, et que le *ticket* est valide. Ce dernier est vérifié en utilisant la clé publique du *Master* de la bulle. Si l'une de ces conditions n'est pas satisfaite alors la demande d'association de l'objet à la bulle est rejetée.

Une fois que le *Follower* réussit sa $1^{ère}$ transaction (*association request*), il n'aura plus besoin de mettre son *ticket* dans une transaction pour s'authentifier.

Pour mieux comprendre, la procédure d'authentification des *Followers* en utilisant *BBTrust* est comme suit :

1. la $1^{ère}$ transaction d'un *Follower* - signée par sa clé privée *privF*- représente une requête d'association à la bulle. Cette transaction doit contenir un *ticket* d'authentification signé par le *Master* de la bulle ;
2. à la réception de la $1^{ère}$ transaction du *Follower*, la *blockchain* vérifie son intégrité en utilisant la clé publique du *Follower*. Ensuite, elle vérifie la validité des informations contenues et l'authenticité du *ticket* à l'aide de la clé publique du *Master* ;
3. une fois que la validité du *ticket* est vérifiée, la *blockchain* enregistre une information qui associe le *GrpID*, l'*ObjID* et la clé *pubF* ;
4. toute n ($n \geq 2$) transaction (contenant la charge utile et l'identifiant de l'objet (*GrpID*, *ObjID*)) doit être signée par la clé privée du *Follower* (*privF*) ;
5. la *blockchain* vérifie l'intégrité de la transaction en vérifiant la validité de la signature par *pubF* ;
6. si la transaction est intègre alors la *blockchain* vérifie si *pubF* est associé avec les paramètres *GrpID* et *ObjID* trouvés dans la transaction ;
7. si la clé publique est associé à l'identifiant de l'objet alors ce denier est authentifié avec succès, sinon il est rejeté.

Les objets légitimes (ayant des *tickets*) peuvent être ajoutés à leurs bulles à tout moment. Théoriquement, le nombre d'objets par bulle est illimité car *BBTrust* s'appuie sur une architecture complètement décentralisée. Les objets ayant des faux *tickets* ou qui n'en possèdent pas, ne peuvent pas être associés aux bulles. Par conséquent, ils ne peuvent pas recevoir ou envoyer des messages depuis/vers des objets appartenant à ces zones sécurisées. Pour finir, toutes les transactions doivent être signées. De ce fait, l'authentification des objets et l'intégrité de leurs communications sont assurées. Pour plus de détails sur *BBTrust* voir chapitre 9 page 115.

Par rapport au mécanisme de migration des objets, en appliquant *bubbles of trust*, un ensemble de CPANs appartenant à un même service et qui échangent des informations entre eux, peut être protégé par une bulle. Pour plus de détails voir section 9.7.1 page 126.

Notre approche est implémentée en utilisant *Ethereum* comme *blockchain*. S'appuyer sur une *blockchain* publique apporte beaucoup d'avantages. En effet, les *blockchains* sont

des systèmes décentralisés qui assurent leur propre fonctionnement d'une manière autonome (validation des blocs, consensus, etc). De plus, ils sont très robustes et résistants contre la falsification et l'altération des données, ainsi, les informations stockées liées à l'authentification et concernant les objets de confiance sont fiables.

Il est à noter également qu'une fois que le *smart contract* est déployé, les utilisateurs ne peuvent pas le modifier car le contrat est envoyé et validé en tant que transaction.

Nous avons établi une implémentation réelle [2] et réalisé une démonstration en utilisant des Raspberry Pi et des ordinateurs pour avoir un concept de preuve. Le programme des objets est écrit en langage C++. Le *Smart contract* en *Solidity* et enfin, pour le déploiement de la *blockchain* et le client qui interagit avec, nous avons utilisé *TestRPC*. Et tout comme dans le cas de *BCTrust*, nous avons développé une deuxième interface qui encode/décode le code C++ vers/depuis *Ethereum*.

Ensuite, nous avons fait une évaluation complète qui concerne principalement le temps d'exécution et l'énergie consommée de *bubbles of trust*. Les résultats était satisfaisants. Le Tableau 6 présente les résultats concernant 100 tests, où nous avons mesuré :

1. t.assoc : le temps nécessaire pour préparer une requête d'association ;
2. t.data : le temps nécessaire pour préparer un message de données ;
3. CPU.assoc : la puissance consommée du processeur (CPU) pour préparer une requête d'association ;
4. CPU.data : la puissance consommée du processeur pour préparer un message de données ;
5. NIC.assoc : la puissance consommée du contrôleur d'interface réseau (*Network Interface Controller* (NIC)) pour établir une association (envoyer une requête + recevoir une réponse) ;
6. NIC.data : la puissance du NIC pour un message de données (envoyer un message + recevoir un reçu).

Objet	**t.assoc**	**t.data**	**CPU.assoc**	**CPU.data**	**NIC.assoc**	**NIC.data**
Ras.PI	28.03 (mS)	0.82 (mS)	64.16 (mW)	16.29 (mW)	89.24 (mW)	31.22 (mW)
Laptop	1.56 (mS)	0.04 (mS)	9.76 (mW)	3.35 (mW)	16.14 (mW)	12.54 (mW)

TABLEAU 6 – Résultats de l'évaluation de *bubbles of trust* (moyennes des temps et énergie consommés).

Nous avons également estimé le coût financier de notre approche pour différents cas d'utilisation (voir section 9.7.2 page 128). D'après les résultats obtenus concernant le temps, l'énergie et le coût financier de *BBTrust*, nous avons prouvé que notre approche est applicable sur la plupart des cas d'utilisation de l'IoT.

Nous avons développé un protocole de sécurité robuste, flexible et pas coûteux, qui est léger, rapide et adapté aux contraintes de l'IoT, permettant d'assurer les services d'authentification et de migration transparente des objets, d'intégrité, de confidentialité et de disponibilité des données, tout en gardant une haute flexibilité et évolutivité des systèmes qui le déploient.

2. `https://github.com/MohamedTaharHAMMI/BubblesOfTrust-BBTrust-`

Plan du manuscrit

Ce manuscrit est organisé comme suit : d'abord, nous allons commencer par la partie II page 25, qui définie un état de l'art sur les différentes technologies de communication utilisées par les systèmes d'Internet des objets et leurs protocoles et approches de sécurité. Ces approches seront étudiées et évaluées à la fin de cette partie afin de dégager leurs (1) avantages et leurs (2) problèmes et failles de sécurité.

Ensuite, en se basant sur ces travaux, dans la partie III page 77, nous allons proposer un nouveau protocole de sécurité utilisant des algorithmes robustes et rapides qui permet de résoudre ces problèmes liés à la sécurité, tout en étant adapté aux contraintes des technologies IoT. Notre protocole est passé par plusieurs versions, chacune améliore et complète sa précédente. Dans la version 1 (publiée dans [71]) chapitre 5 page 79, nous proposons un protocole de sécurité de base qui assure l'authentification d'un objet lors de son association à un réseau. Cette version assure également le service d'intégrité des données. La version 2 (publiée dans [72]) chapitre 6 page 87 permet une authentification mutuelle entre un objet voulant s'associer à un réseau et le gestionnaire du réseau, et assure également l'intégrité des données. De plus, elle rajoute un mécanisme de génération et échange de clés de session (pour le mode *unicast* et le mode *broadcast*). La version 3 (publiée dans [69] et dans [68]) chapitre 7 page 91 rajoute le service de confidentialité des données, et améliore/optimise le service d'intégrité des données. La version 4 (publiée dans [67]) chapitre 8 page 105 permet d'avoir un système d'authentification décentralisé (basée sur une *blockchain* privée) et rajoute un service de migration sécurisée des objets d'un réseau vers un autre. Enfin, la *version 5* (publiée dans [70]) chapitre 9 page 115 (basée sur une *blockchain* publique) améliore la version 4 du protocole, et offre un service d'authentification totalement décentralisé qui fournit plus d'évolutivité au réseau, une grande disponibilité d'information, et une bonne flexibilité et transparence aux systèmes IoT.

La partie IV page 139 sera une évaluation générale et une conclusion finale de ce travail de recherche.

Deuxième partie

l’IoT et la sécurité

« *The value of a telecommunications network is proportional to the square of the number of connected users of the system* »

Robert Metcalfe

Chapitre 1

Introduction

L'Internet des Objets est une évolution majeure dans le domaine de la technologie d'information (*Information Technology* (IT)) qui domine et règne de plus en plus sur le marché des systèmes informatiques. D'après *Statista*[1] la taille du marché mondial de l'Internet des objets en 2017 était de 1130.3 milliards de dollars, soit une augmentation de 16.12 % comparé à l'année 2009. Et que, par rapport à l'année 2017, cette taille augmentera de 66.08 % en 2019 pour atteindre une valeur égale à 1710.4 milliards de dollars.

L'IoT, appelé également Web 3.0, représente une extension d'Internet à des choses et à des lieux du monde physique. Chaque objet physique -qui peut être une personne, un ordinateur, un smartphone, une voiture, un capteur, une maison intelligente, etc- est associé à une entité virtuelle qui se comporte comme une entité active dans le système. Selon *IBM*, neuf milliards d'objets sont aujourd'hui connectés, et le nombre augmente sans cesse et très rapidement. La réussite de ce nouveau paradigme s'explique (1) par la variété des équipements (objets) qu'on utilise dans notre vie quotidienne et leur développement. En IoT, chaque objet doit avoir les caractéristiques suivantes :

L'existence : selon le *Larousse*[2], l'existence est le fait d'avoir une réalité et d'avoir une présence en un lieu. Un objet tel qu'un véhicule existe dans le monde physique, mais grâce à un équipement de communication intégré et des technologies spécifiques associées, le véhicule possède également une identité (une existence / une réalité) dans le monde virtuel.

L'autonomie : l'autonomie est la capacité d'un objet à se contrôler soi-même. Elle représente les propriétés d'une entité qui est capable de fonctionner de manière indépendante et sans être commandé de l'extérieur. En effet, chaque objet possède une identité lui représentant, peut librement rassembler, analyser, traiter, générer et échanger des informations. Il peut également prendre des décisions sans aucune intervention humaine.

La connectivité : c'est à dire ce qu'une entité offre comme connexion à d'autres entités de son environnement. Ainsi, n'importe quel objet autorisé peut initier une communication avec d'autres objets et utiliser leurs informations.

L'interactivité et l'interopérabilité : l'internet des objets est un concept général qui en-

1. *Statista* est un site de statistiques, études de marché et un portail de business intelligence : `https://www.statista.com/statistics/485136/global-internet-of-things-market-size/`
2. *Larousse* est une encyclopédie en langue française : `http://www.larousse.fr/dictionnaires/francais/existence/32144`

globe différents domaines utilisant des systèmes, architectures et matériels différents. Par conséquent, un objet doit pouvoir interagir et collaborer avec d'autres objets hétérogènes. Ces derniers peuvent être des humains ou des machines, réels ou virtuels, qui produisent et consomment une grande variété de services.

La flexibilité : un objet peut interagir avec d'autres objets à n'importe quel moment (*Any time*), n'importe où (*Any where*), et n'importe comment (*Any how*), et fait des calculs pour n'importe quel objet (*Anything*), pour n'importe qui (*Anyone*), et pour n'importe quel service (*Any service*).

Ce succès s'explique aussi (2) par l'évolution des technologies de communication, notamment celle qui sont sans fils. Ces technologies sont devenues plus fiables, plus optimales, et moins coûteuses en terme de calcul, de mémorisation et de consommation d'énergie.

Cependant, malgré ce succès, à cause de la menace des cyberattaques, la sécurité de l'IoT reste encore un des problèmes majeurs qui freine l'évolution et le déploiement rapide de cette technologie de technologies.

Le 20 Décembre de l'année 2017, l'*Identity Theft Resource Center*[3] (ITRC) a enregistré 1293 violations de données aux États-Unis, visant plus de 174 millions de dossiers confidentiels. Ce nombre représente 21% de plus que ce qui avait été enregistré au même moment en 2016. À la fin de l'année, le nombre total de violations a atteint un record de 1300 violations, comparé à 1093 violations en fin d'année 2016. D'après le rapport de l'ITRC, les résultats obtenus par rapport aux violations concernent principalement 5 secteurs : le secteur de gestion (50.5 %), le secteur médical/santé (28.3%), le secteur éducatif (8.8 %), le secteur bancaire/financier (7.1 %) et le secteur gouvernemental/militaire (5.3 %). Quant aux particuliers, *Alteryx*[4] affirme que presque tous les foyers Américains sont touchés par une violation de données massive.

Et selon une étude menée par *Ponemon Institute*[5] et *Accenture*[6], en 2017 les cyberattaques dans le monde coûtent en moyenne 11.7 millions de dollars par an à chaque entreprise. Ce qui représente une augmentation de 62 % comparé à l'année 2012.

De ce fait, il est essentiel d'établir des mesures de protections et d'investir en matière de sécurité afin de mettre en sûreté ses systèmes informatiques. Car dans la plupart des cas le coût des pertes est bien plus important que le coût de la sécurisation.

Dans ce qui suit (chapitre 2 page 31), nous allons expliquer ce que c'est la sécurité informatique, ses notions de base, ses composants, et ses objectifs. Nous allons également citer et définir brièvement les différentes catégories d'attaques informatiques, no-

3. *Identity Theft Resource Center* : est une organisation Américaine à but non lucratif, fondée pour aider les victimes des cyberattaques, et fournir des conseils aux consommateurs par le biais de son centre d'appel gratuit, de son site web et des médias sociaux : `https://itsecuritycentral.teramind.co/2018/01/03/cyber-security-statistics-2017-data-breaches-and-cyber-attacks/`

4. *Alteryx* est une société d'analyse de données : `https://www.upi.com/Report-Nearly-every-American-household-affected-by-massive-data-breach/2341513746646/`

5. *Ponemon Institute :* est un institut qui mène des recherches indépendantes sur la politique de confidentialité, de protection des données et de la sécurité de l'information.

6. *Accenture* est une entreprise internationale de conseil et de technologies : `https://www.accenture.com/fr-fr/company-news-release-cost-of-cyber-crime`

tamment celles qui ciblent les entités communicantes, et celles qui visent la confidentialité, l'intégrité, et la disponibilité des données.

Une fois qu'on présente les mécanismes de sécurité de base et qu'on explique leurs rôles dans la protection des réseaux et des systèmes, en particulier les systèmes IoT, dans le chapitre 3 page 35 nous allons établir un état de l'art sur les différentes technologies de communication utilisées par l'IoT, ainsi que les protocoles de sécurité déployés sur chacune d'entre elles. Le but de cette étude est d'analyser et comparer les solutions existantes, d'en extraire les points forts et les points faibles, afin de pouvoir proposer une meilleur approche de sécurité (voir partie III page 77).

Chapitre 2

Les notions de base de la sécurité

On peut définir la sécurité informatique comme étant le fait d'assurer le bon fonctionnement d'un système et de garantir les résultats attendus de sa conception. Autrement dit, la sécurité représente l'ensemble de politiques et pratiques adoptées pour prévenir et surveiller l'accès non autorisé, l'utilisation abusive, la modification ou le refus d'une opération informatique. A partir de cette définition, on peut extraire les bases de la sécurité qui sont décrites dans ce qui suit.

Authentification
L'authentification est le mécanisme de sécurité qui permet de prouver l'identité d'une entité. En effet, il existe plusieurs méthodes d'authentification qu'on peut classer en 4 catégories :

- L'authentification avec ce qu'on sait, c'est à dire que l'entité prouve son identité avec une information secrète, qui n'est connue que par un nombre limité d'objets légitimes. Généralement le nombre d'objets concernés ne dépasse pas 2 (ex. un client et un serveur). Les mécanismes les plus utilisés dans cette catégorie sont les mots de passe et les numéros personnels d'identité (*Personal Identity Number* (PIN)) ;
- L'authentification avec ce qu'on possède. Dans cette catégorie, une entité s'authentifie grâce à une donnée stockée. Cette donnée peut être secrète comme les clé prépartagé (*Pre-Shared Key* (PSK)), ou publique comme les certificats numériques et les jetons ;
- L'authentification avec ce qu'on est. Ça concerne généralement les utilisateurs humains, qui ont des caractéristiques biométriques qui leurs sont uniques telle que la voix, l'empreinte digitale, l'iris, et les veines ;
- L'authentification avec comment on se comporte. Cette dernière catégorie est basée sur les profiles comportementaux de chaque utilisateurs. Chaque entité à une façon de travaille particulière, par exemple, sa façon de taper sur un clavier, les horaires de travail habituels, l'environnement de travail habituel, etc.

Confidentialité
La confidentialité est le mécanisme qui permet de cacher une donnée, et de cacher même l'information de son existence. Ainsi, empêcher toutes entité(s) non autorisée(s) d'avoir accès à cette donnée. Généralement, on assure ce service en utilisant le chiffrement de données. Ce dernier est basé sur des algorithmes mathématiques permettant de déformer un texte en clair est le remettre à sa forme initiale grâce au à une ou plusieurs clés cryptographiques.

Intégrité
L'intégrité est un mécanisme assurant qu'une donnée ne soit pas : falsifiée, modifiée, altérée ou supprimée par une entité non autorisée. Dans la plupart des cas, ce service est réalisé en utilisant des fonctions de hachages avec des propriétés de signature de données.

Disponibilité
La disponibilité est le mécanisme qui permet de garantir la bonne exécution d'un service, et le bon fonctionnement du système. Afin de garantir la disponibilité d'un service, on utilise des mécanismes qui le protègent contre les arrêts intentionnels telles que les attaques de dénies de service et dénies de service distribués (*Denial/Distributed Denial of service (Dos/DDos)*), et non intentionnels (ex. les erreurs humaines). En outre, on duplique et distribue ce service sur plusieurs serveurs. De cette façon, si l'un des serveurs ne fonctionne plus, les autres maintiennent le service.

Non répudiation
La non répudiation est un mécanisme permettant de garantir qu'une opération ne peut être niée par celui qui l'avait établis. On garantie ce service grâce aux signatures numériques combinées avec des mécanismes qui assurent le non rejeu de données.

Non rejeu
Le non rejeu est un mécanisme garantissant qu'un message échangé entre deux entités A et B, ne doit pas être réutilisé par une entité non autorisée C. La plupart des systèmes intègrent des compteurs et des numéros de séquence différents au niveau des messages échangés, ce qui fait qu'un message ne peut pas avoir le même numéro de séquence que ses n messages précédents (n un nombre de message qui varie selon la politique de sécurité utilisée), sinon il sera automatiquement rejeté.

La résilience
On peut définir la résilience par la capacité d'un système à surmonter une altération de son environnement. Par exemple dans le cas de l'IoT, si un objet est compromis, cela ne devrait pas influencer l'ensemble du réseau.

La confidentialité persistante (forward secrecy)
La confidentialité persistante est une caractéristique cryptographique qui garantit que la découverte d'une information secrète (ex. clé privée) d'un objet légitime par un utilisateur malicieux ne compromet pas la confidentialité des communications passées.

L'évolutivité
L'évolutivité représente l'aptitude d'un système à maintenir des bonnes performances lorsque des ressources (notamment ressources matérielles) lui sont ajoutées.

La tolérance aux fautes
La tolérance aux fautes est un mécanisme permettant à un système de continuer à fonctionner lorsque l'un de ses composants tombe en panne (ex. en dupliquant les serveurs).

L'objectif de la sécurité est de protéger les systèmes informatiques contre les différentes

menaces et attaques qui les ciblent. Ces attaquent consistent en l'exploitation d'une faille au niveau d'un système afin d'atteindre un objectif précis. Ces objectifs peuvent être l'obtention illégale d'un accès au système, le vol des données confidentielles d'une entreprise, l'obtention des informations personnelles sur un utilisateur, récupérer des codes de carte bancaires, etc. Çes attaques peuvent également avoir comme objectif l'interruption ou la perturbation d'un service, la falsification des données, ou l'exploitation des ressources du système. On peut classifier ces attaques par des grandes catégories. La section 2 présente les différentes catégories d'attaques.

Catégories d'attaques

— *attaques d'usurpation d'identité (spoofing attack) :* c'est lorsqu'une entité malveillante réussit à se faire passer pour une autre, obtenant ainsi les droits d'accès et les avantages de la victime ;

— *attaques de rejeu :* c'est quand un utilisateur malicieux copie et renvoie un ou plusieurs message(s) déjà transmis afin d'exploiter les vulnérabilités du système ;

— *attaques par force brute :* en effet le principe de ces attaques consiste à tester un grand nombre de mots de passe dans l'espoir de deviner le bon. Il peut également s'agir d'une opération de déchiffrement de données où l'attaquant essaie toutes les clés possibles jusqu'à ce que la clé correcte soit trouvée (recherche de clé exhaustive) ;

— *attaques par cryptanalyse :* cette catégorie concerne l'étude du flux de chiffrement (cipher), du texte chiffré, ou des crypto-systèmes, afin de trouver des vulnérabilités qui permettent de récupérer le texte en clair à partir du texte chiffré ;

— *attaques de l'homme au milieu (Man In The Middle (MITM)) :* c'est lorsqu'une entité non autorisée se met entre deux ou plusieurs entités communicantes afin d'écouter une communication confidentielle, ou modifier/supprimer des données échangées, voire interrompre le trafic (dénie de service) ;

— *attaques par dénie de service / dénie de service distribué (Dos/DDos) :* elle vise à rendre une ressource ou une information indisponible. Elle peut être réalisée (1) en inondant la machine source ou le réseau par un grand nombre de messages (ex. attaque d'inondation [9]), ou (2) en exploitant une vulnérabilité dans le protocole.

Comparé aux objets utilisés dans l'Internet classique -qui représentent majoritairement des ordinateurs- les objets dans l'IoT représentent tout équipement électronique ayant une capacité de calcul et de mémorisation, qu'il s'agisse d'un capteur très limité en performances et en consommation d'énergie, ou d'un grand data-center alimenté, avec des capacités ultra-puissantes. À cause de cette diversité d'objets, il est difficile de concevoir un protocole de sécurité robuste et au même temps adapté à ces objets variés. En plus, le faite que la tendance dans l'IoT est d'utiliser les technologies de communication sans fil rend le système IoT encore plus vulnérable et plus exposé à toute sorte de cyberattaque. Afin de sécuriser les systèmes IoT, et d'assurer les propriétés vu ci-dessus. Il faut concevoir un protocole basé sur des algorithmes robustes, mais aux même temps légers et flexibles. Ce protocole doit être adapté aux différents types d'objet, du plus puissant au plus faible, sans qu'il y aie une dégradation en terme de performance sécuritaire.

Chapitre 3

Technologies de communication de l'IoT et leurs mécanismes de sécurité

3.1 Introduction

Selon la portée, le débit, la consommation d'énergie, et le domaine d'application, on peut classifier ces technologies en 4 grandes catégories (voir Figure 3.1) :



FIGURE 3.1 – Catégories des technologies de communication

Réseaux étendus sans fil (Wireless Wide Area Network (WWAN)) : ces réseaux sont considérés comme étant les réseaux les plus étendus. Ils représentent généralement les réseaux à liaisons sans fil à faible consommation énergétique (*Low Power Wide Area Network* (LPWAN)) tel que LoRaWAN [124] et Sigfox [18] (ayant un débit théorique de 0.3 K$bits$/S jusqu'à 50 K$bits$/S), et les réseaux cellulaires tel que GSM, UMTS, et LTE (voir la section 3.2.1.2 page 40). Ils peuvent avoir un débit théorique de 2 M$bits$/S (cas de la *2G*) jusqu'à 100 M$bits$ (cas de la *4G**). Les WWANs incluent aussi les réseaux satellitaires tel que le système mondial de positionnement (*Global Positioning System* (GPS)) [65].

Réseaux métropolitains sans fil (Wireless Métropolitain Area Network (WMAN)) : sont basés sur la norme IEEE 802.16. Cette catégorie peut avoir une portée de 4 à 10 Km et offre un débit utile de 1 à 70 M$bits$/S. La technologie WMAN la plus connue est *WiMax* (voir la section 3.2.2 page 47).

Réseaux locaux sans fil (Wireless Local Area Network, WLAN) : ces réseaux ont une porté d'une centaines de mètres, soit l'équivalent d'un local d'entreprise. En théorie, il peuvent fournir un débit de plus de 50 Mbits/s. Les technologies les plus connus sont *Wi-Fi* (voir

la section 3.2.3.1 page 50) et *HIgh PERformance radio LAN* (HiperLAN) [50], qui suivent la norme IEEE 802.11.

Réseaux personnels sans fil (Wireless Personal Area Network, WPAN) : concernent les réseaux sans fil à faible portée, de l'ordre de quelques dizaines de mètres. Tout comme la porté qui varie d'une technologie WPAN à une autre, le débit varie aussi. Ce dernier peut être à 250 K$bits$/S (*ZigBee*) jusqu'à 1 M$bits$/S (cas du *Bluetooth*). Ces technologies suivent la famille IEEE 802.15, les plus connues celles de la sous norme IEEE 802.15.1 (*Bluetooth*), et celles qui sont utilisées dans le domaine des réseaux de capteurs sans fil (WSN pour Wirless Sensor Networks) qui suivent principalement la sous norme IEEE 802.15.4 tel que *ZigBee*, *OCARI*, *ISA100*, *6LoWPAN*, etc. Ces technologies sont connues par leurs optimalités en énergie et résistances aux interférences dans les zones industrielles.

Les réseaux sans fils représentent des mécanismes de transport de données entres objets, et entre objets et réseaux filaires classiques. Ces technologies servent à recevoir et transmettre des informations à l'aide d'ondes électromagnétiques. Les réseaux sans fil permettent aux périphériques d'être déplacés à différents degrés de liberté tout en conservant la communication entre eux. Ils offrent également une plus grande flexibilité que les réseaux câblés et réduisent considérablement le temps et les ressources nécessaires pour mettre en place de nouveaux réseaux et facilite la création, la modification ou la démolition des réseaux [65].

Ces technologies sont très variées, elles se diffèrent selon leurs portés, leurs débits, et leurs cas d'usage. Dans ce qui suit, on va citer les technologies les plus connues en commençant par les plus étendus jusqu'aux réseaux personnels. On s'intéresse principalement aux réseaux WPAN, plus précisément aux réseaux de capteurs sans fils (*Wireless Sensor Networks* WSN) qui représente un sous domaine de l'IoT utilisant principalement des objets à faibles puissance de calcul et mémorisation, et contraints en énergie. Les technologies de communication des WSNs utilisent la norme IEEE 802.15.4.

L'objectif de ce chapitre est de fournir un état de l'art sur différentes technologies de communication et domaines utilisés par l'IoT et expliquer leurs architectures et mode de fonctionnement. On va traiter principalement les aspects de sécurités, notamment les mécanismes d'authentification et de gestion des clés.

3.2 Réseaux sans fil et leurs mécanisme de sécurité

3.2.1 Réseaux étendus sans fil (WWAN)

Les réseaux étendus permettent de couvrir des très grandes zones (à échelle de plusieurs kilomètres), et englobent différentes technologies telles que les liaisons sans fil à faible consommation énergétique (*Low-Power Wide-Area Network* (LPWAN)) (ex. *LoRaWAN*), les technologies cellulaires (ex. *1G*, *2G*, *3G*, *4G*, etc), et les réseaux satellitaires (ex. GPS).

3.2.1.1 LoRaWAN

Long Range (LoRa) est une technique de modulation permettant de fournir un signal à très longue porté. Cette technique de modulation est basée sur des techniques à spectre étalé et une à variation de *Chirp Spread Spectrum* (CSS) [47] avec des mécanismes de correction d'erreur directe *Forward Error Correction* (FEC) intégrées. Afin de diffuser un signal, LoRa utilise la totalité d'un canal de la bande passante, ce qui rend le signal plus

fiable et robuste contre le bruit. LoRa représente la couche physique. Elle peut être utilisée par différents protocoles de niveaux suppérieurs, et déployée sous différentes topologies (mesh, étoile, etc).

LoRaWAN est le protocole de la sous couche *Media Access Control* (MAC)qui est standardisée et normalisée pour les réseaux étendus à faible puissance (LPWAN) grâce à la LoRa Alliance [124]. Cette dernière vise aussi à rendre *LoRaWAN* interopérable avec d'autres technologies de communication. *LoRaWAN* est entièrement bidirectionnel. Elle possède une architecture totalement adaptée à l'IoT, lui permettant de localiser facilement les objets mobiles. Elle est déployée pour des réseaux nationaux par des grands opérateurs de télécommunications (ex. Orange). Les réseaux *LoRaWAN* sont généralement présentés par une topologie en étoile d'étoiles laquelle des passerelles relient des terminaux (ex. capteurs, ordinateurs, etc) à un serveur réseau central, qui est relié à son tour à un serveur d'applications. La communication entre passerelles et terminaux est faite en utilisant la technologie Long Range modulation technique (LoRa^{TM}) à un saut, ou *Frequency Shift Keying* modulation technique (FSK) [124]. Tandis que la communication entre passerelles et serveur de réseaux, et entre ce dernier et le serveur d'application, est réalisée via le standard IP. La Figure 3.2 montre l'architecture utilisée et déployée dans *LoRaWAN*.

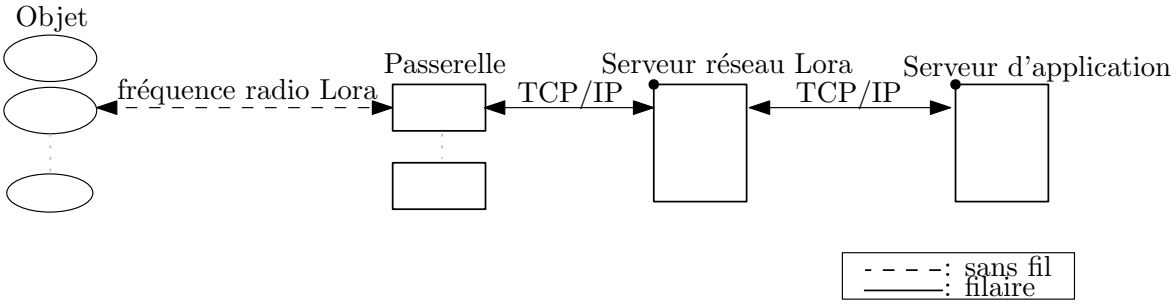


FIGURE 3.2 – L'architecture de *LoRaWAN*

Afin d'optimiser le compromis entre la latence du réseau et la durée de vie de la batterie, le protocole LoRaWAN utilise 3 classes (A,B, et C) d'objets. Chaque objet implémente au moins les fonctionnalités de la classe A. Les classe B et C sont optionnelles, mais restent compatibles avec la classe A (voir [124] pour plus de détails).

La sécurité dans LoRaWAN

La politique de sécurité de *LoRaWAN* assure les mécanismes de base qui sont l'authentification des objets, la confidentialité et l'intégrité des données. Cette politique définie également des techniques de partage de clés.

Authentification des objets et partage des clés :

D'après la spécification de *LoRaWAN* [124], (1) la première méthode d'authentification et de partage de clés s'appelle la *Over The Air Activation (OTAA)*. Chaque objet est munie d'une clé d'application (AppKey), qui représente une clé symétrique unique de 128 bits pré-partagée avec le serveur réseau de *LoRaWAN*. Afin qu'un objet puisse être associé à un réseau *LoRaWAN*, il doit d'abord envoyer une *requête d'association* (join request) au réseau. Cette requête qui contient l'identifiant unique d'application (AppEUI), l'identifiant unique de l'objet (DevEUI), et un nonce aléatoire (pour éviter les attaques par cryptanalyse) de l'objet (DevNonce) d'une taille de 2 octets, doit être complétée par un code d'intégrité de message (*Message Integrity Code* (MIC)). Le MIC (4 octets) représente la signature du message en utilisant AppKey. Le MIC est calculé comme décrit dans l'Algorithme 1

Algorithm 1: Calcul du MIC

```
Function Concat () // concatène les données
Function Aes128_cmac () // assure l'intégrité de données
Function MIC () // retourne les 4 premiers bytes du MAC
begin
    _data = Concat (MHDR, AppEUI, DevEUI, DevNonce)
    _MAC = Aes128_cmac (AppKey, _data)
    _MIC = MIC (MAC)
```

Où $_data$ représente les données à traîter, _MAC est la variable qui contient le code d'authentification de message (*Message Authentication Code* (MAC)) décrit dans la RFC 4493 [148], _MIC est la variable qui contient le MIC, et MHDR est l'entête du MAC (MAC Header) permettant de déterminer le type du message. A la réception de la requête, le serveur vérifie le MIC en utilisant AppKey. Si l'authentification réussit, alors le serveur génère une clé de session d'application (application session key (AppSKey)) et une clé de session réseau (network session key (NwkSKey)). Ces nouvelles clés sont calculées comme expliqué dans l'Algorithme 2.

Algorithm 2: Génération de clés

```
Function Concat () // concatène les données
Function Aes128_encrypt () // assure la confidentialité de données
begin
    _data1 = Concat (0x01, AppNonce, NetID, DevNonce, pad16)
    _data2 = Concat (0x02, AppNonce, NetID, DevNonce, pad16)
    NwkSKey = Aes128_encrypt (AppKey, _data1)
    AppSKey = Aes128_encrypt (AppKey, _data2)
```

Où $_data1$ et $_data2$ représentent les données à traiter, $Aes128_encrypt$ est une fonction de chiffrement symétrique définie dans [64], $AppNonce$ est un nombre aléatoire du réseau, $NetID$ est l'identifiant réseau (network identifier), et $pad16$ représente des données de bourrage.

Ensuite le serveur envoie une acceptation d'association (join accept) chiffré par AppKey (l'algorithme de chiffrement est décrit dans 3.2.1.1 page ci-contre). Cette réponse comprend l'$AppNonce$, le $NetID$, l'adresse de l'objet (DevAddr), des données de configuration (RFU, $RxDelay$), et les canaux à utiliser ($CFList$). Ce message est associé à un deuxième MIC généré par l'Algorithme 3.

Algorithm 3: Calcul du deuxième MIC

```
begin
    _data = Concat (MHDR, AppNonce, NetID, DevAddr, RFU, RxDelay, CFList)
    _MAC = Aes128_cmac (AppKey, _data)
    _MIC = MIC (MAC)
```

Enfin, l'objet vérifie le MIC de la réponse reçue de la part du serveur, déchiffre le message avec AppKey, puis recalcule et installe les clés de session à partir de AppKey et des données reçues.

(2) La deuxième méthode s'appelle l'activation par personnalisation (*Activation By Personalisation* (ABP)). Cette dernière se diffère de OTAA par le faite que les objets sont

déjà pré-configurés avec le $DevAddr$ et les deux clés de session ($NwkSKey$ et $AppSKey$) qui lui sont unique. Cette personnalisation peut être faite dans l'environnement de fabrication des objets par exemple. Par conséquent les objets peuvent commencer à communiquer avec le serveur réseau directement.

Dans les deux méthodes, le faite de pré-configurer deux entités communicantes avec un secret (ex. AppKey), et de l'utiliser pour échanger des signatures (ex. MIC) prouve l'identité de ces entités. Par conséquent, l'authentification mutuelle entre ces entités est assurée.

Confidentialité et intégrité de données :
Une fois que l'objet soit associé au réseau de *LoRaWAN*, tous les messages échangés doivent être chiffrés (pour assurer la confidentialité) et signés (pour assurer l'intégrité) en utilisant les clés de session (connus uniquement par le serveur réseau et l'objet concerné). Le chiffrement de message est établis via le standard AES128 [64] avec le mode d'opération à compteur (CounTeR (CTR)) [102]. L'Algorithme 4 décrit l'opération de chiffrement :

Algorithm 4: Algorithme de chiffrement de *LoRaWAN*

```
Function ceil () // arrondissement d'un nombre
Function len () // longueur d'une donnée
Function encrypt () // chiffrement de données avec aes128_encrypt
begin
    k ← ceil(len(FRMPayload)/16)
    for (i=0; i ≤ k; i++) do
        A_i = (0x01 || (0x00 × 4) || Dir || DevAddr || FCntUp ou FCntDown || 0x00 || i)
        S_i = encrypt (Key, A_i )
        S = S_1 || S_2 ||..||S_k
```

Où || est l'opérateur de concaténation, key représente $NwkSKey$ ou $AppSKey$, A représente des champs de données en clair, $FRMPayload$ est la charge utile de la donnée à chiffrer, Dir est le champ de direction (0 pour les trames de liaison montante et 1 pour les trames de liaison descendante), $FCntUp$ et $FCntDown$ sont respectivement des compteurs d'envoi et de réception qui sont maintenus par l'objet et le serveur. Ces compteurs ne doivent jamais se répéter afin d'éviter les attaques par rejeu. Et enfin S c'est le résultat après chiffrement. S est la séquence de clés (keystream) qui inclut les valeurs $FCntUp$ ou $FCntDown$, ce qui permet de générer des séquence de clés différente pendant toute la durée de vie de l'objet.

Enfin, afin d'obtenir des données chiffrées, on fait le $\oplus$ (ou exclusif) entre $FRMPayload$ et S. L'opération de déchiffrement s'établit par un $\oplus$ entre les données chiffrées et S.

Une partie des données (MAC payload) du message doit être signer afin d'empêcher la modification du message chiffré ou d'autres valeurs telles que $DevAddr$, $FCntUp$ ou $FCntDown$. Cette signature est le MIC, qui est calculé comme décrit par L'Algorithme 5 page suivante.

Où FHDR (Frame header) est l'entête de la trame et $FPort$ (Port field) représente la valeur du port de communication.

Le schéma dans la Figure 3.3 page suivante résume les différents mécanismes de sécurité déployés par *LoRaWAN*. Le partage de clés se fait grâce à la méthode OTAA (présenté dans l'encadrement) ou grâce à la méthode ABP. L'expression "$\{\}_{key}$", signifie chiffré en utilisant la clé 'key.

Algorithm 5: Algorithme de signature de *LoRaWAN*

```
Function sign () // signature de données avec aes128_cmac
begin
    Msg = MHDR || FHDR || FPort || FRMPayload
    B0 = (0x49 || 4 × 0x00 || Dir || DevAddr || FCntUp or FCntDown || 0x00 || len(msg))
    MAC = sign (NwkSKey, B0 || msg)
    MIC = MAC[0..3]
```

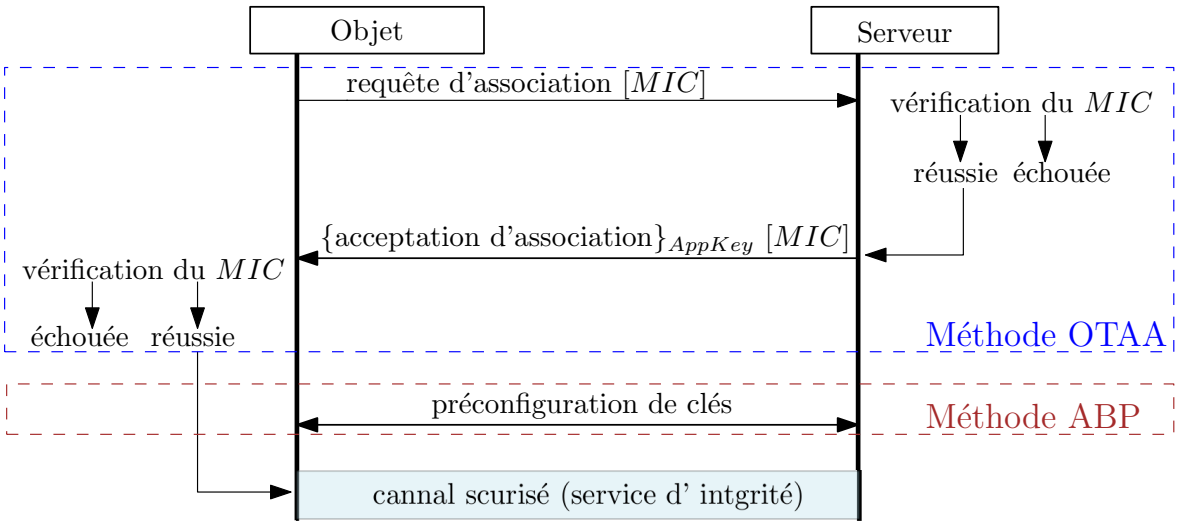


FIGURE 3.3 – Le protocole de sécurité de *LoRaWAN*

3.2.1.2 Les technologies cellulaires

Elles représentent les systèmes radio mobiles. Ces technologies sont passées par 4 générations et la $5^{ème}$ est en cours de développement. La première (*1G*) est apparue vers les années 1970. Elle était basée sur un système de communication mobile analogique. La *1G* a exploité deux grandes inventions techniques : le microprocesseur et le transport de données entre les téléphones mobiles et la station de base. Cette technologie était très coûteuse, et elle utilisait des appareils très volumineux. Ce système analogique était remplacée dès l'apparition d'un autre système numérique plus performant (la $2^{ème}$ génération, ou la *2G*). La *2G* a été inventé à la fin des années 1980, elle permet de transporter les signaux vocaux et échanger des données numériquement. Cette technologie assure une meilleure qualité ainsi qu'une plus grande capacité à moindre coût pour l'utilisateur. Le *Global System for Mobile Communication* (GSM) représente le standard de la *2G*. Cette génération représente une évolution vers une offre enrichie et diversifiée de services tel que le démarrage du marché des messages courts *Short Message Service* (SMS). C'était également les débuts des services d'Internet sur mobile en créant le *Wireless Application Protocol* (WAP). Après, à la fin des années 90, la troisième génération (*3G*) est sortie -labellisé IMT 2000 par l'*Union Internationale des Télécommunications* (UIT) [134]- elle offre des services de communications plus performants pour le transport de la voix et le transfert de données. Le standard de la *3G* s'appelle l'*Universal Mobile Telecommunications System* (UMTS). Comparé à la *2G*, l'UMTS est caractérisé par son haut débit, il offre des services en plus tel que le paiement mobile, la localisation, le multimédia (visiophonie et messages multimédia), l'itinérance internationale, etc. En décembre 2010, l'UIT a accordé aux normes Long Term Evolution (LTE) et *Worldwide Interoperability for Microwave Access* (*WiMAX*) la possibilité commerciale d'être considérées comme des technologies *4G*.

On expliquera la technologie *WiMAX* dans la partie 3.2.2 page 47. Issu du 3GPP (3rd Generation Partnership Project), qui est une instance de coordination entre instituts de normalisation télécoms comme l'*European Telecommunications Standards Institute* (ETSI), l'*Association of Radio Industries and Businesses/Telecommunication Technology Committee* (ARIB/TTC, au Japon), la *China Communications Standards Association* (CCSA), et la *Alliance for Telecommunications Industry Solutions* (ATIS), le LTE a pour objectif d'avoir un très haut débit (aux alentours de 40 Mbits), une taille des cellules de (1) 5 km avec performances optimales, (2) 30 km avec performances raisonnables, et (3) 100 km avec performances acceptables. Elle vise également à permettre une co-existence avec les standards actuels, autrement dit assurer que les clients passent d'un standard à un autre d'une manière totalement transparente, sans interruption de la communication ni intervention manuelle. Et pour finir, [14] explique que la $5^{ème}$ génération (5G) devra être un changement de paradigme qui inclut des largeurs de bande massives, des densités extrêmes de stations de base et d'appareils, et d'un nombre sans précédent d'antennes. Contrairement aux quatre générations précédentes, la 5G sera également très intégratrice, c'est à dire, elle permettra de lier toute nouvelle interface et spectre 5G avec le LTE et le *Wi-Fi* pour offrir une couverture universelle à haut débit et une expérience utilisateur transparente.

La sécurité dans les réseaux LTE

Dans cette partie, on s'intéresse uniquement au LTE, car à partir de l'année 2012, ce dernier est devenu la technologie la plus déployée. D'après [23], plus de 85% des abonnées utilisent cette technologie. La spécification du *3rd Generation Partnership Project* (3GPP) TS36.300 [122] montre une architecture simplifiée du LTE (voir Figure 3.4).

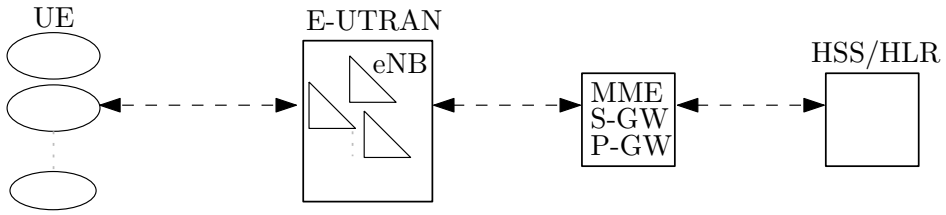


FIGURE 3.4 – L'architecture du LTE

— UE : terminal mobile,

— E-UTRAN : réseau d'accès radio terrestre universel évolué,

— eNB : station de base de la E-UTRAN,

— MME : entité de gestion de la mobilité,

— S-GW : passerelle de service,

— P-GW : passerelle réseau de données par paquets,

— HLR : enregistreur de localisation géographique des abonnés,

— HSS : sur-ensemble du HLR intégrant des nouveaux protocoles de cœur de réseau (Diameter et SIP) propres aux réseaux *4G*.

Les UEs communiquent avec les eNBs dans la E-UTRAN, qui à leurs tour communiquent avec les MMEs -en passant par des passerelles si nécessaire- qui sont connectées aux HSS / HLR. Selon [170] le LTE assure les services principaux de la sécurité, qui sont l'authentification des objets, la confidentialité et l'intégrité de données.

Authentification et échange des clés :

Afin d'assurer une authentification mutuelle entre un terminal (ex. carte *Subscriber Identity Module* (SIM)) et un HLR ou HSS à travers les eNBs et les passerelles, et sécuriser l'accès aux réseaux mobiles, le LTE opte pour le protocole *3GPP Authentication and Key Agreement* (3GPP AKA). En plus de l'authentification, ce dernier sert aussi comme mécanisme d'échange de clés. La partie *Authentification* du protocole AKA permet de vérifier l'identité de l'usager tandis que la partie *Key Agreement* permet la génération des clés de sessions qui assurent la confidentialité et l'intégrité des données échangées sur le trafic réseau d'un utilisateur. AKA représente un protocole qui utilise un mécanisme de challenge/réponse basé sur des clés cryptographiques symétriques. Le schéma sur la Figure 3.5 explique le protocole AKA.

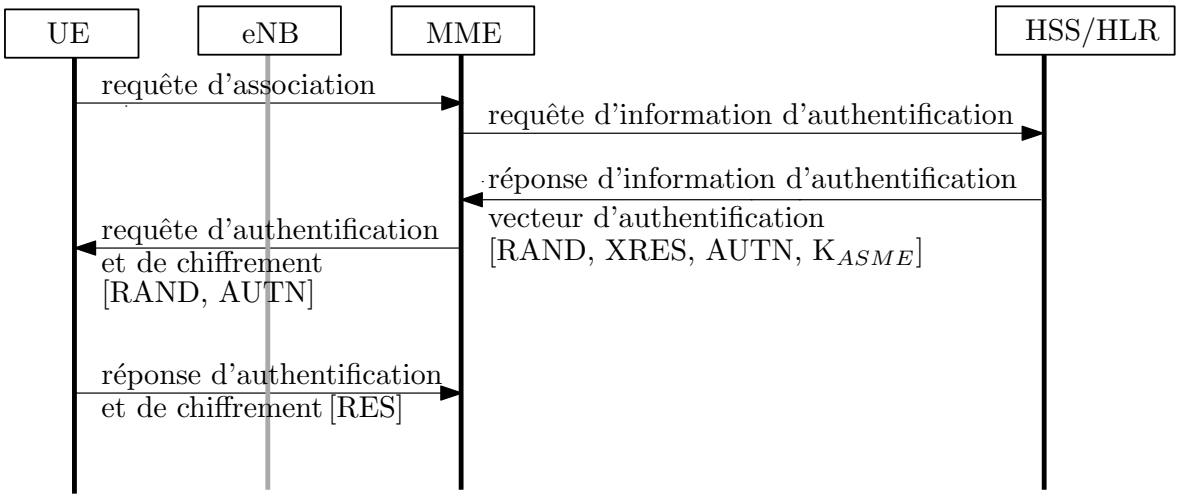


FIGURE 3.5 – Une présentation simplifiée du protocole d'authentification et d'échange de clés AKA

L'authentification est basée sur une clé symétrique secrète et unique K, pré-partagée entre l'UE et le HLR. D'abord l'UE envoie une requête d'association à l'HLR par l'intermédiaire d'un MME, qui lui à son tour transforme la requête d'association en une requête d'information d'authentification (requête du vecteur d'authentification) et l'envoie à l'HLR. Ces vecteurs sont :

— RAND (challenge) : un nombre aléatoire d'une taille de 128 bits généré par le HLR, qui sert en tant que paramètre d'entré pour la génération des autres paramètres ;

— XRES : la réponse attendue (par rapport au challenge), elle sera comparée avec la valeur de RES généré par l'UE ;

— AUTN : un jeton qui permet l'authentification du réseau auprès de l'UE. Ce jeton comprend un numéro de séquence (SQN) chiffrée (SQN $\oplus$ K), et un *Message Authentication Code* (MAC) généré à partir d'une fonction de hachage en utilisant SQN et K, ce qui permet l'authentification du HLR auprès de l'UE ;

— K_{ASME} : est la clé dérivé de CK et IK qui sont dérivées de K. CK et IK représentent respectivement une clé de chiffrement et une clé d'intégrité entre l'UE et le HLR. Le Tableau 3.1 page ci-contre résume et montre la hiérarchie de tous les clés générés pour assurer les services de confidentialité et d'intégrité entre les différentes entités dans l'architecture réseau du LTE.

Après, à la réception du quadruplé (RAND, AUTN, XRES et K_{ASME}), le MME garde XRES et K_{ASME} (utilisée pour générer d'autres clés voir Tableau 3.1 page suivante), transmet RAND et AUTN à l'UE (via une requête d'authentification et de chiffrement), et attend une réponse RES de ce dernier. l'UE vérifie l'AUTN reçu, puis calcule le RES (un résultat

-	UE	eNB	MME	HLR/HSS
UE	-	K_{Penc}, K_{RRCint}, K_{RRCenc} (dérivées de K_{eNB})	K_{NASint}, K_{NASenc}, K_{eNB} (dérivées de K_{ASME}), K_{ASME}	CK, IK (dérivées de K), K

TABLEAU 3.1 – Les clés de confidentialités et d'intégrités utilisées entre les différentes entités LTE

généré à partir de K et du challenge) et l'envoie au MME. L'UE est authentifié si RES et XRES sont identiques. Une fois authentifié, l'UE génère CK et IK et des clés dérivées afin d'assurer un échange de données sécurisé. Le processus de génération des paramètres et des clés est résumé dans la Figure 3.6.

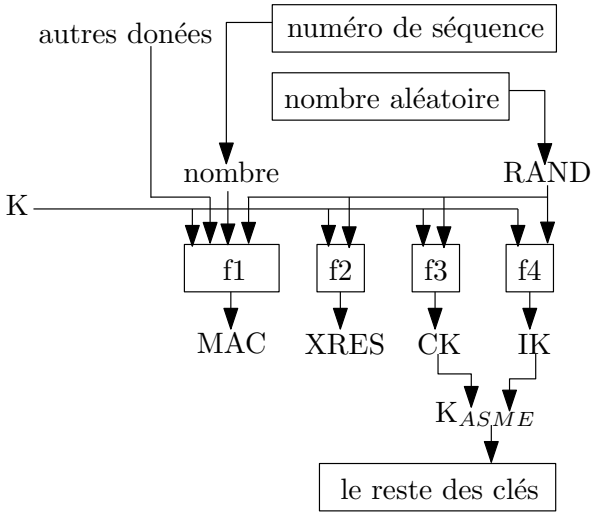


FIGURE 3.6 – Génération de paramètres et de clés

Voir [7] afin d'avoir les détails sur le fonctionnement des fonctions (f1, f2, f3, f3, f4).

Confidentialité et intégrité de données :

Une fois que l'authentification mutuelle et l'association sont établies, la confidentialité et l'intégrité de données sont assurées en utilisant les clés générés. Le LTE utilise un standard de confidentialité appelé *Evolved Packet System Encryption Algorithm* (128-EEA3) et un autre pour l'intégrité appelé *Evolved Packet System Integrity Algorithm* (128-EIA3) [60]. Les deux standards sont basés sur le flux de chiffrement *SNOW 3G* [5]. Il existe une version qui utilise le standard AES [60] (on expliquera l'AES dans la section 7.1.1 page 91).

L'algorithme de confidentialité 128-EEA3 fournit un chiffrement par flux utilisé pour chiffrer/déchiffrer des blocs de données via une clé symétrique (ex. CK). Le bloc de données peut avoir une longueur comprise entre 1 et 2^{32} $bits$ [6]. Le standard 128-EEA3 permet de générer un flux de clé (keystream) de L mots. Chacun des mots est écrit sur 32 $bits$, par conséquent on obtient une chaîne binaire Z = L × (Z[0] ‖ Z[1] .. ‖ Z[31]), où Z[0] est le bit de poids fort (*Most significant bit* (MSB)) du premier mot généré, Z[31] est le bit de poids faible (Least Significant Bit (LSB)), et ‖ est l'opérateur qui exprime la concaténation. Afin de chiffrer/déchiffrer une donnée d'une taille LENGTH $bits$ (LENGTH peut être n'importe quelle valeur ≥ 0), il faut que L soit égale à ceil(LENGTH/32), où ceil() est une fonction qui retourne la valeur entière supérieure la plus élevée en arrondissant la valeur si nécessaire (ex. ceil (4.3) = 5, ceil (39) = 39)).

Les opérations qui permettent le chiffrement et le déchiffrement sont identiques, elles sont réalisées en faisant des ⊕ (ou exclusif) d'une donnée avec le keystream Z généré (voir Algorithme 6 page suivante et Algorithme 7 page suivante respectivement).

Algorithm 6: L'opération de chiffrement du standard 128-EEA3

begin
  // $C = M \oplus Z$:
  $M = M[0] \| M[1] \| ... M[LENGTH-1]$
  $Z = Z[0] \| Z[1] \| ... Z[LENGTH-1]$
  for $(i = 1, i \leq LENGTH, i++)$ **do**
    $C[i] = M[i] \oplus Z[i]$
    $C \leftarrow C \| C[i]$

Algorithm 7: L'opération de déchiffrement du standard 128-EEA3

begin
  // $M = C \oplus Z$:
  $C = C[0] \| C[1] \| ... C[LENGTH-1]$
  $Z = Z[0] \| Z[1] \| ... Z[LENGTH-1]$
  for $(i = 1, i \leq LENGTH, i++)$ **do**
    $M[i] = C[i] \oplus Z[i]$
    $M \leftarrow M \| M[i]$

M est la donnée en clair en bits avec une taille LENGTH. C est la donnée chiffrée d'une taille LENGTH.

L'algorithme qui assure l'intégrité de données est définie dans le standard 128-EIA3. Ce dernier est une fonction qui permet de générer des MACs en utilisant un keystream Z basé sur une clé symétrique d'intégrité (ex. IK) [6]. Au niveau de ce standard le MAC représente une donnée de 32-bits qui est générée comme décrit dans l'Algorithme 8.

Algorithm 8: Mécanisme simplifié de génération du MAC

begin
  $T \leftarrow 0$, `// Initialisation du MAC (32 bits)`
  for *(i=0 ; i ≤ L ; i++)* **do**
    for *(j=0 ; j ≤ (LENGTH-1) ; j++)* **do**
      if *(M[j] = 1)* **then**
        $T = T \oplus i^{ème} Z$
  MAC = le $L^{ème}$ T

Afin de vérifier l'intégrité, on recalcule le MAC à partir de la donnée et on fait une comparaison, si la valeur du MAC n'a pas changé, alors ça prouve que la donnée n'a pas été modifiée.

3.2.1.3 Les technologies satellitaires

Un satellite de communication est un satellite artificiel qui relaie et amplifie les signaux de radiocommunication via un transpondeur. Le satellite permet la création d'un canal de communication entre un émetteur et un récepteur à différents endroits sur terre. Les satellites de communication sont utilisés dans différents domaines tels que la diffusion des programme de télévision, la téléphonie, la radio, la géolocalisation, Internet et les applications militaires. Les communications sans fil terrestres utilisent des ondes électromagnétiques pour transporter des signaux. Ces ondes nécessitent une ligne de visée, par

conséquent, ils sont obstruées par la courbure de la terre. Le but de la communication par satellites est de relayer le signal autour de la courbe de la terre, ce qui permet une communication entre des points largement séparés. Les satellites de communication utilisent une large gamme de fréquences radio et micro-ondes. Afin d'éviter les interférences de signaux, les organisations internationales ont des réglementations sur les gammes de fréquences (bandes de fréquences), pour lesquelles certaines organisations sont autorisées à utiliser. Cette attribution de bandes minimise le risque d'interférence de signal [89].

Le 6 avril 1965, *Intelsat* [54] a lancé le premier satellite de télécommunication commercial, appelé *Intelsat I*. Ce satellite permettait le transfert des images et de la voix, donc assurer des applications de téléphonie, la diffusion des programmes télévisés, et la communication vers des avions et des navires. Au début, la recherche et le développement des technologies de communication spatiales était restreint aux organismes internationaux comme *Intelsat* et *Inmarsat* [22]. Après, au début des années 2000, grâce aux progrès techniques et la forte croissance d'activité sur ce secteur, beaucoup de regroupements spatiaux comme *Globalstar Group* [46], *Iridium* [106] et SpaceX [142] sont apparues. Aujourd'hui, les services satellitaires ne concerne plus uniquement les grands organismes, mais également les entreprises de communication tel que *SigFox* [101]. L'objectif de ces entreprises est d'adapter la technologie satellitaire à l'augmentation du nombre d'objets intelligents, et d'assurer une connexion fiables entre ces derniers, même ceux qui se trouvent dans les zones les plus éloignées du globe.

On peut classifier les satellites en deux catégories, les satellites à orbite géostationnaire (*Geostationary Earth Orbit (GEO)*) et les satellites à orbite terrestre basse (*Low Earth Orbit (LEO)*).

La première catégorie représente le système de communication par satellite traditionnel qui est placé dans l'orbite géostationnaire de la terre. Comme la distance GEO est éloignée de la terre, il y a un problème de délai de communication du signal. GEO n'est donc pas adapté à un système de communication personnel. La deuxième catégorie concerne les systèmes de communication par satellite LEO. Cette technologie est plus récente, en général, le système de communication par satellite LEO constitue un système de communication par satellite mobile global utilisant exactement 66 satellites LEO [79]. L'orbite du satellite LEO est plus proche de la terre que le satellite GEO. Les systèmes à satellites LEO présentent trois avantages principaux qui sont : (1) l'atténuation de la communication du signal est faible, (2) le délai de communication du signal est court et (3) les canaux de communication de données sont larges mais plus étroits que le GEO.

La sécurité dans les réseaux satellitaires

Il existe plusieurs travaux qui visent à sécuriser les réseaux de communication satellitaire. [84] étudie le service de la confidentialité dans un réseau satellitaire bidirectionnel composé de deux utilisateurs mobiles qui souhaitent échanger des messages via un satellite multi-faisceaux. D'autres travaux [140] proposent l'utilisation du protocole *Satellite Secure Sockets Layer* (SSL) [59] qui représente l'utilisation du protocole SSL dans les réseaux satellitaires afin d'assurer l'authentification des utilisateurs, la confidentialité et l'intégrité de données.

[79] propose un protocole de sécurité intéressant, désigné principalement aux systèmes de communication par satellite LEOs. La Figure 3.7 page suivante montre une architecture simplifiée du réseau.

Cette architecture comprend des satellites LEO (SatLEO), des passerelles, des utilisateurs mobiles (UM) et un centre de contrôle de réseau (*Network Control Center* (NCC)). Le satellite gère les communications entre un utilisateur mobile et une passerelle, un uti-

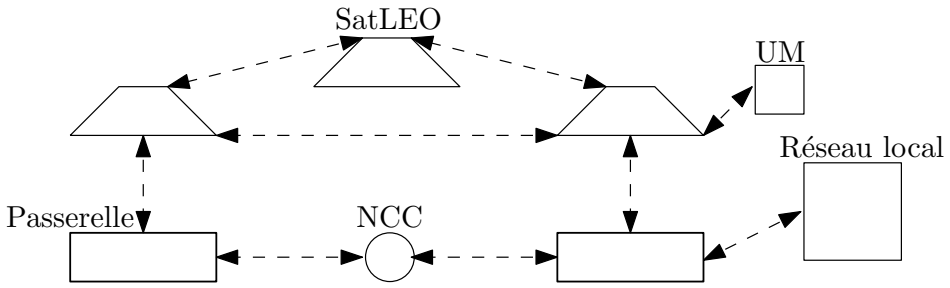


FIGURE 3.7 – Un système de communication satellitaire (LEO)

lisateur mobile et d'autres utilisateurs mobiles, une passerelle et d'autres passerelles, une passerelle et le NCC, ou un satellite et d'autres satellites. Les passerelles représentent des points relais pour la communication entre le NCC et les satellites. En outre, elles sont reliées aux réseaux locaux pour produire un système de communication diversifié.

Enregistrement et authentification :
Afin de pouvoir communiquer, un utilisateur mobile doit d'abord (1) s'enregistrer, puis -lors de son association- (2) s'authentifier. Dans la phase d'enregistrement, la passerelle attribue une identité permanante U_{id}, une clé secrete K_{md}, et une identité temporaire T_{id} à l'utilisateur, ensuite envoie une copie de ces informations, associée par l'identifiant du LEO approprié (LEO_{id}) au NCC. Ce dernier stocke, d'une manière sécurisée, le quadruplet U_{id}, K_{md}, T_{id} et LEO_{id} de chaque utilisateur.

Avant chaque communication, un utilisateur doit être authentifié. L'opération d'authentification est décrite par la Figure 3.8 comme suit :

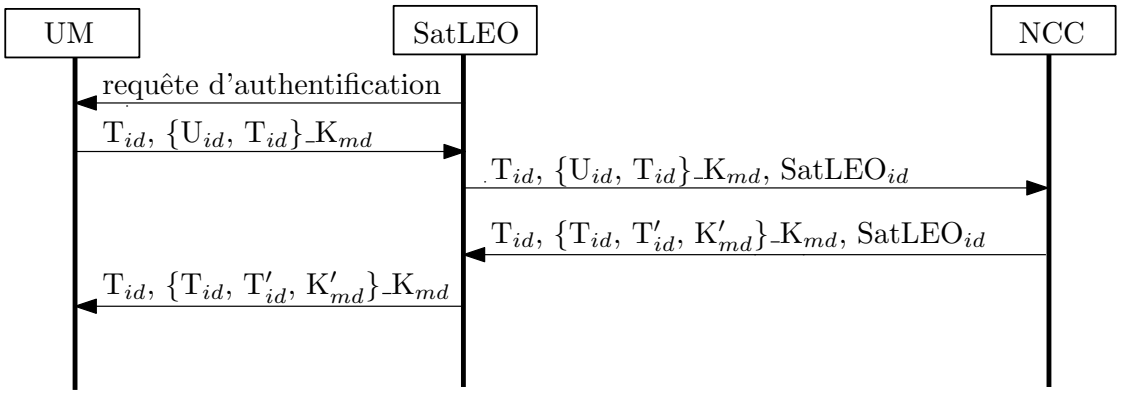


FIGURE 3.8 – Mécanisme d'authentification d'un système satellitaire (LEO)

L'expression "{}_*key*", signifie chiffré avec la clé *key*. Dans ce mécanisme, le chiffrement est établis par le *Data Encryption Standard (DES)* [149]. D'abord, le satellite LEO (SatLEO) envoie une requête d'authentification à l'utilisateur. En recevant la requête, l'utilisateur crée et envoie un message qui contient le chiffré de ses identifiants (U_{id} et T_{id}) en utilisant K_{md} associé par le T_{id} en clair. Ensuite, ce message est concaténé par le LEO_{id} puis transféré au NCC par le satellite. Le NCC vérifie la validité du LEO_{id} puis utilise le T_{id} comme index pour chercher -dans sa base de données- la clé de session K_{md} (prépartagée avec l'utilisateur). Avec cette clé, il déchiffre le message et obtient l'U_{id} et le T_{id}. Après, il vérifie la validité de l'U_{id}, et s'assure que les deux T_{id} (celui qui est envoyé directement en clair avec le déchiffré) sont identiques. Si les vérifications de données réussissent, alors le NCC génère aléatoirement un nouveau T'_{id} et une nouvelle clé de session K'_{id} pour l'utilisateur et met à jour sa base de donnée avec ces informations. Le faite que

l'opération de déchiffrement est réussie, prouve que l'utilisateur est légitime et ça permet l'authentification de ce dernier, car la clé K_{md} qui est secrète est n'est partagée qu'entre le NCC et l'utilisateur. Le NCC envoie à son tour une réponse qui contient l'ancien T_{id} en clair avec une partie chiffrée (par K_{md}) composée du triplet (T_{id}, T'_{id} et du K'_{id}), le message est associé également par le LEO_{id}. Enfin, le LEO valide son identifiant et retransmet le reste du message à l'utilisateur. Ce dernier, déchiffre le message avec l'ancienne K_{md}, vérifie la valeur du T_{id}, et enfin, stocke (K'_{md} et T'_{id}) pour les utiliser dans une prochaine authentification. Le fait d'avoir un T'_{id} unique protège les utilisateurs contre les attaques de rejeu, et l'unicité de K_{md} par session protège cette dernière contre les attaques de cryptanalyse.

Confidentialité de données :
Dans [79], on explique pas le mécanisme de protection de données utilisé, mais ils utilisent le *Data Encryption Standard (DES)* pour d'autre opérations telque l'authentification. DES est un algorithme de chiffrement symétrique (chiffrement par bloc), basé sur des opérations de permutation, de substitution et l'opérateur ou exclusif ($\oplus$).

3.2.2 Réseaux métropolitains sans fil (WMAN)

Les réseaux métropolitains sans fil permettent la couverture de très grandes zones géographiques (à échelle de plusieurs kilomètres), tout en assurant des très hauts débits. Ils sont basés sur la norme IEEE 802.16, et la technologie la plus connue est le *WiMAX*.

3.2.2.1 WiMax

WiMAX est une famille de standards de communication sans fil basée sur l'ensemble des normes IEEE 802.16 (ex. IEEE 802.16a, IEEE 802.16e). Cette famille définie plusieurs options de (1) couches physiques (PHY) et de (2) sous couches de contrôle d'accès au support (*Media Access Control* (MAC) de la couche liaison de données [1]. Cette technologie a été créé par le *WiMAX Forum* [161]. Ce dernier représente un consortium créé en 2001 qui inclus plus de 100 membres dont des fournisseurs majeurs tels que *AT&T* [119], *Fujitsu* [165] et *Intel* [62].

WiMAX peut être déployée en une topologie réseau maillée, mais généralement, il utilise une topologie appelée point à multi-points (*point to multipoint*) [125]. Cette dernière est composée de stations de base centrales, chacune couvre une zone d'un rayon inférieur à 3.5 km, et relie plusieurs utilisateurs (voir Figure 3.9). Une ou plusieurs stations de base peuvent être connectés à Internet.

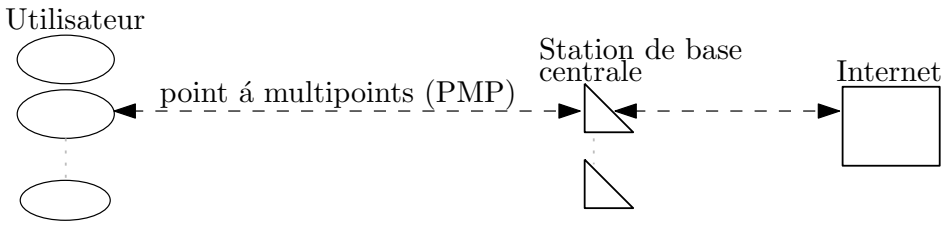


FIGURE 3.9 – La topologie point à multi-points du *WiMAX*

Le *WiMAX* est classé en deux catégories : (1) le *WiMAX* mobile (IEEE 802.16e) et (2) le *WiMAX* fixe (IEEE 802.16, IEEE 802.16c, IEEE 802.16a, IEEE 802.16f, IEEE 802.16m). Dans

1. La couche liaison de données et composée de deux sous couches, la sous couche liaison de données (*Logical Link Control* (LLC)) et la sous couche MAC.

ce qui suit, nous nous intéressons uniquement à la première catégorie, car elle supporte la mobilité des objets, permet un passage transparent d'un relais à un autre, et elle est plus adaptée que la deuxième catégorie pour être utilisée comme une technologie de communication IoT.

Au niveau de la couche physique, le *WiMAX* mobile utilise une modulation à accès multiple par répartition orthogonale de fréquence échelonnable (*Scalable Orthogonal Frequency-Division Multiple Access* (SOFDMA)). Cette technique permet le partage des ressources radio en temps et en bande passante, et adapte une qualité de service (*Quality of Service* (QoS)) pour chaque utilisateur [125]. Le *WiMAX* supporte également la technologie à entrées/sorties multiples (*Multiple Input, Multiple Output* (MIMO)) [2].

Au niveau de la sous couche MAC, dans les systèmes sans fil qui utilisent des méthodes d'accès ne gèrant pas la QoS tel que la méthode d'accès multiples de détection de porteuse avec évitement de collision (*Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA))[169], le trafic de chaque utilisateur peut être à tout moment perturbé par d'autres utilisateurs. Ce qui pose un problème à certaines applications sensibles (ex. application de monitoring à temps réel). La sous couche MAC du *WiMAX* mobile est conçue spécialement pour résoudre ce problème grâce à (1) sa communication orientée connexion. En effet, avant chaque communication, les utilisateurs doivent établir une connexion de bout en bout, par conséquent cela permet un transfert fiable de données. En outre, cette sous couche déploie (2) un algorithme d'ordonnancement d'accès au réseau. Cet algorithme attribue dynamiquement des ressources à n'importe quel utilisateur voulant communiquer. En revanche, selon la priorité des services, ces ressources radio peuvent être augmentées ou diminuées. Par conséquent, ceci engendre une optimisation de la bande passante, et une stabilité au réseau.

La sécurité dans le *WiMAX* mobile (IEEE 802.16e)

Le standard IEEE 802.16e utilise une méthode d'authentification mutuelle appelée (*Pairwise Master Key (PKMv2)*), qui peut déployer (1) l'*Extensible Authentication Protocol* (EAP) [8] ou (2) le protocole *Rivest Shamir Adelman* (RSA) [138] avec l'utilisation des certificats numériques X509 [76]. Ce standard opte pour une variété d'algorithmes cryptographiques robustes. Il permet également d'assurer les services de confidentialité, d'intégrité de données, et protège contre les attaques de rejeu.

Authentification mutuelle et gestion des clés :

D'abord la station de base (SB) authentifie l'utilisateur et vérifie sa légitimité. Et ensuite l'utilisateur authentifie aussi à son tour la SB pour s'assurer qu'il ne se connecte pas à un faux réseau (ex. un réseau créé par une station malveillante). La SB envoie une clé d'autorisation (*Authorization Key* (AK)) à l'utilisateur avec un ensemble d'informations de sécurité (*Security Association* (SA)) qui définie les profiles de sécurité supportés.

Cette authentification mutuelle est réalisée grâce à l'un des deux méthodes :

La première, représente une authentification basée sur l'algorithme cryptographique asymétrique RSA [139]. Cette méthode est basée sur une pair de clés (privée/publique). Une entité communicante voulant s'authentifier, doit signer une donnée (généralement un hash) avec sa clé privée, qui est secrète et n'est connue que par l'entité qui la possède. RSA assure que la vérification de la signature ne peut être faite qu'avec la clé publique associée, et qu'il ne peut y avoir qu'une seul clé publique associée à la privée et vice versa. Par conséquent, cette opération prouve que c'est le possesseur de la clé privée qui a fait la

2. MIMO : est une technique qui utilise plusieurs antennes à la fois pour l'émission et pour la destination. Ces antennes sont combinées afin de minimiser le taux d'erreurs de transmissions.

signature. Cependant cette clé publique ne prouve pas l'identité de son possesseur, d'où interviennent les certificats X509. Un certificat numérique est une structure de données contenant des informations sur la date de validité, l'émetteur du certificat, et des informations sur l'identité de l'entité concernée. Ce certificat contient également la clé publique de l'entité. Toutes les informations du certificat sont hachées (pour protéger leurs intégrités) et signées par la clé privé d'une autorité de certification, laquelle les entités en question font confiance. En d'autres termes, le certificat permet une association fiable de la clé publique et l'identité.

La deuxième, est une authentification basée sur le protocole EAP. Ce protocole -décrit dans [8]- ne spécifie aucun algorithme cryptographique, mais utilise plutôt des politiques et outils de sécurité différents (ex. mot de passe, jeton, etc). Par exemple [35] propose le *EAP Password Authenticated Exchange*, qui est une extension du EAP basée sur des clés pré-partagées. La politique de sécurité doit respecter l'ensemble des critères définit dans la section 2.2 de [151]. Ces critères concernent les exigences de génération de clés, la robustesse des clés utilisées, les exigences d'authentification mutuelle, les mesures de protection contre les attaques du type homme du milieu (*Man In The Midle* (MITM)) et les attaques de rejeu, etc.

Après l'authentification, le standard exige un mécanisme de gestion de clé d'autorisation (*Authorisation Key (AK) Management*).

D'abord lors d'une réception d'une requête d'autorisation, venant d'un utilisateur qui n'est pas encore autorisé, la SB génère une nouvelle AK, puis l'associe à une valeur qui indique la durée de vie de l'AK, et envoie ces données à l'utilisateur via une réponse d'autorisation. La clé AK représente une clé symétrique permettant l'échange et la génération des clés de chiffrement. AK reste active jusqu'à ce que son délais expire (elle reste valide jusqu'à la fin de la durée de grâce). Si un utilisateur ne parvient pas à redemander une autorisation avant l'expiration de sa AK actuelle, la SB considère cet utilisateur comme étant non autorisé, et ne détient plus alors d'AK active pour celui-ci. L'utilisateur est responsable de redemander une autorisation auprès de la SB et de maintenir sa AK active. Pour chaque demande de renouvellement de clé (Key Request), la SB régénère une nouvelle AK, lui attribut une durée de vie qui est égale à la durée restante pour l'ancienne AK plus la durée de vie de base, et envoie la réponse à l'utilisateur. Cette opération se répète à chaque fois que l'AK s'approche à s'expirer. Tant que la période d'uneAK n'est pas encore expirée, la SB peut supporter l'échange d'un même utilisateur utilisant deux AK valides. Les échanges de ses messages sont sécurisés par (1) RSA (chiffrement/signature asymétrique) ou par (2) par EAP (expliqués précédemment).

La Figure 3.10 page suivante résume les mécanismes d'autorisation.

Confidentialité et intégrité de données :
Les services de confidentialité et d'intégrité de données sont établis au niveau de la sous couche MAC grâce au standard de chiffrement AES avec le mode d'opération CCM (expliqué dans 7.1.3 page 94).

En effet, le chiffrement des paquets est réalisé au niveau de la sous couche MAC via l'algorithme de chiffrement AES-CCM, en utilisant une des clés dérivées de AK (une clé pour le mode unicast, une autre pour le mode boradcast, etc). Ces clés ont une taille de 128 $bits$. Avant de chiffrer le paquet, on associe au texte en clair un numéro de séquence de 4 $octets$. Le numéro de séquence est incrémenté après chaque transfert de paquet afin de se protéger contre les attaques de rejeu. Chaque numéro de séquence reçu plus d'une seul fois doit être rejeté. L'AES-CCM permet de générer aussi un MAC, calculé à partir du texte en clair, pour protéger l'intégrité des paquets.

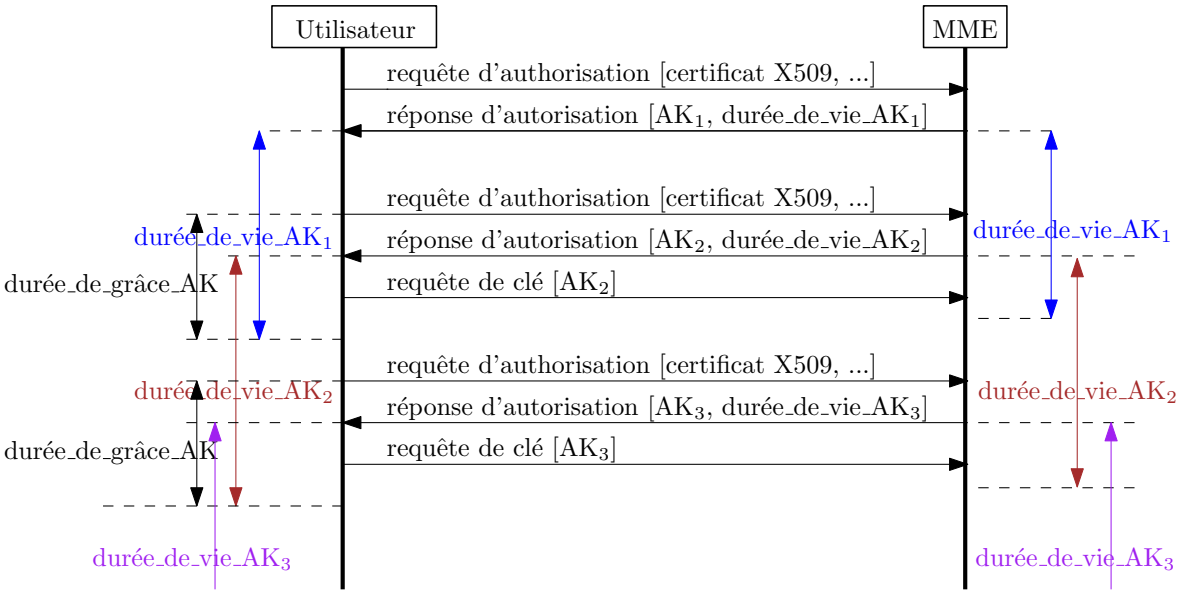


FIGURE 3.10 – L'opération d'autorisation de *WiMAX*

3.2.3 Réseaux locaux sans fil (WLAN)

Les réseaux locaux sans fils représentent des réseaux sur lesquels un objet peut se connecter à un réseau local (LAN) via des ondes radio sans fil. Les WLAN -qui suivent la norme IEEE 802.11- ont une porté équivalente à un réseau d'un bâtiment ou d'un campus (une centaines de mètres). Le *Wi-Fi* est sans doute la technologie WLAN la plus utilisée par les entreprises et les particuliers. Dans ce qui suit on va décrire cette technologie, ainsi se focaliser sur protocoles de sécurité que cette technologie déploie.

3.2.3.1 Wi-Fi

Craig J.Mathias [112] [111] -directeur de *Farpoint Group (un cabinet de conseil spécialisé dans le domaine des réseaux mobiles et réseaux sans fil)*- estime que le *Wi-Fi* sera la technologie la plus adaptée à l'Internet des Objets, et que c'est cette technologie qui va diriger la révolution de ce paradigme. Ceci s'explique par le fait que *Wi-Fi* offre plusieurs avantages, qui sont :

L'exploitation d'une infrastructure existante : aujourd'hui, la plupart des organisations possèdent une infrastructure *Wi-Fi*. Cette technologie offre des avantages majeurs en terme de capacité, de couverture et de facilité d'utilisation. L'ajout de plus en plus des clients et d'applications est devenu une activité quotidienne. Étant donné que les applications IoT sont souvent optimales (peu de transmissions et quantité limitée de données), une charge supplémentaire de données IoT n'influence pas sur les performances du *Wi-Fi*, et dans la plupart des cas, elle ne pose aucun problème. A noter également que de nombreuses applications d'Internet des objets utilisées par les entreprises seront tout simplement de nouvelles applications fonctionnant sur smartphones et autres appareils qui utilisent déjà les réseaux *Wi-Fi*.

Un format compact : cette technologie peut être déployée dans des équipements minuscules (puce, module, etc).

L'évolutivité : l'élargissement de la zone de couverture et l'amélioration de la capacité du

Wi-Fi seront nécessaires après un certain temps, mais à court terme, le standard IEEE 802.11ac [130] et la diversité de modules *Wi-Fi* répondent déjà largement à ces besoins.

L'établissement rapide de la connexion : les travaux qui s'effectuent actuellement par l'IEEE 802.11ai [129] Working Group, vise à améliorer le temps de configuration et de l'établissement de connexion.

Une communication basée IP : comme l'IoT exige toujours l'utilisation des identifiants uniques, alors le *Wi-Fi* utilise déjà l'adressage IP, y compris l'IPv6, et tous les protocoles associés, dans ce contexte, selon [112] « *la technologie Wi-Fi est parfaite pour l'Internet des objets* ».

Wi-Fi est un ensemble de protocoles de communication sans fil géré par les normes du groupe IEEE 802.11. Il fournit un réseau qui permet de relier plusieurs objets via des ondes radio. La marque déposée *Wi-Fi* correspond initialement au nom donné à la certification délivrée par la *Wi-Fi* Alliance (anciennement *Wireless Ethernet Compatibility Alliance (WECA)*) [13], organisme ayant pour mission de certifier les matériels conformes à la norme 802.11, ainsi assurer l'interopérabilité entre ces derniers. La portée d'un réseau *Wi-Fi* peut couvrir des zones de plusieurs dizaines de mètres.

Architecture en couches

La norme IEEE 802.11 s'attache à définir les couches basses du modèle OSI [168], pour une liaison sans fil utilisant des ondes électromagnétiques. D'après [154] le premier standard *Wi-Fi* a défini trois couches physiques, correspondant à trois types de produits :

— *IEEE 802.11 Frequency Hopping Spread Spectrum (FHSS) :* utilise la technique d'étalement de spectre basé sur le saut de fréquence ;

— *IEEE 802.11 Direct Sequence Spread Spectrum (DSSS) :* utilise aussi la technique d'étalement de spectre mais sur une séquence directe ;

— *IEEE 802.11 InfraRed (IR) :* de type infrarouge.

Les réseaux IEEE 802.11 FHSS et IEEE 802.11 DSSS sont des réseaux radio émettant sur la bande ISM [3]. Ces trois types de *couche physique* ne sont pas compatibles entre eux, ainsi, une carte IEEE 802.11 FHSS ne peut pas communiquer avec une carte IEEE 802.11 DSSS, et réciproquement. De même, pour une carte IEEE 802.11 IR, qui n'est pas compatible avec les réseaux IEEE 802.11 FHSS ou avec les réseaux IEEE 802.11 DSSS. Pour palier à ce problème, de nombreuses améliorations ont été apportées au standard d'origine, ce qui a fait apparaître cinq nouvelles couches physiques (appelées également normes 802.11 physiques) qui sont la IEEE 802.11b, IEEE 802.11a, IEEE 802.11g, IEEE 802.11n, IEEE 802.11ac.

IEEE 802.11b ou Wi-Fi : utilise la même bande ISM que IEEE 802.11 mais avec des débits pouvant atteindre 11 Mbit/S. IEEE 802.11b est en réalité une amélioration de IEEE 802.11 DSSS. Par conséquent, il est compatible avec IEEE 802.11 DSSS.

IEEE 802.11a ou Wi-Fi5 : utilise une nouvelle bande, appelée bande U-NII, située autour de 5 GHz. Le débit de IEEE 802.11a peut atteindre 54 Mbit/S, en revanche le standard est incompatible avec 802.11 DSSS, FHSS et 802.11b.

3. En 1985 les Etats-Unis ont libéré trois bandes de fréquence à destination de l'Industrie, de la Science et de la Médecine. Ces bandes sont : 902-928 MHz, 2.400-2.4835 GHz, 5.725-5.850 GHz

<table>
<tr><td rowspan="2">OSI couche 1
liaison
de données</td><td colspan="8">802.11 LLC</td></tr>
<tr><td colspan="8">802.11 MAC</td></tr>
<tr><td>OSI couche 2
physique</td><td>FHSS</td><td>DSSS</td><td>IR</td><td>Wi-Fi
802.11b</td><td>Wi-Fi
802.11g</td><td>Wi-Fi-5
802.11a</td><td>802.11n</td><td>802.11ac</td></tr>
</table>

FIGURE 3.11 – Modèle en couches de l'IEEE 802.11

IEEE 802.11g : utilise la bande ISM, avec un débit pouvant atteindre 20 Mbit/S. Ce standard utilise la forme d'onde OFDM de 802.11a. Mais, contrairement à l'IEEE 802.11a, l'IEEE 802.11g, il est compatible avec les standards 802.11 DSSS et IEEE 802.11b.

IEEE 802.11n : le 802.11n a été conçu pour pouvoir utiliser les bandes de fréquences de 2,4 GHz ou 5 GHz. Le débit théorique atteint les 450 Mbit/S (débit réel jusqu'à 200Mbit/S dans un rayon de 100 mètres) grâce aux technologies MIMO (voir note 2 page 48) et *Orthogonal Frequency Division Multiplexing* (OFDM). Ce standard peut combiner jusqu'à 2 canaux de 20 MHz non superposés, ce qui permet, en théorie d'atteindre une capacité totale théorique de 600 Mbit/S dans la bande des 5 GHz.

IEEE 802.11ac : elle permet une connexion sans fil haut débit dans la bande de fréquences inférieures à 6 GHz (communément appelée bande des 5 GHz). Le 802.11ac offre jusqu'à 1300 Mbit/S de débit théorique, en utilisant des canaux de 80 MHz, soit jusqu'à 7 Gbit/S de débit global dans la bande des 5 GHz (de 5170 MHz à 5835 MHz).

La norme IEEE 802.11 définit la première et seconde couches basses du modèle OSI, autrement dit, la couche physique et la sous couche MAC. La Figure 3.11 illustre l'architecture du modèle proposé par le groupe de travail 802.11 comparée à celle du modèle OSI. En plus des standards vu en haut qui concernent le débit du réseau, on trouve d'autres standards 802.11 qui servent à assurer d'autres fonctionnalités tel que l'interopérabilité (ex. IEEE 802.11f), l'amélioration de la qualité de service (IEEE 802.11e), la sécurité (IEEE 802.11i), etc.

Interconnexion des équipements Wi-Fi

Pour connecter les éléments d'un réseau *Wi-Fi*, il existe généralement deux modes, le mode dit *Infrastructure*, et le mode *Ad-Hoc*. Le premier est basé sur un point d'accès (PA) permettant aux périphériques sans fil de se connecter entre eux et/ou à Internet. Des stations se trouvant à portée de radio avec un PA forment un ensemble de services de base (*Basic Service Set* (BSS)). Chaque BBS est muni d'un identifiant (BSSID) qui est sous forme d'une séquence de 6 $octets$ correspondant à l'adresse MAC du PA.

Le deuxième mode fonctionnent d'une manière totalement distribué, il permet la communication entre deux périphériques sans qu'il y ait une infrastructure. Les stations se trouvant à portée de radio forment ce qu'on appelle un ensemble indépendant de services de base (*Independant Basic Service Set* (IBSS)).

On peut composer un réseau à plusieurs BSS -grâce à un système de distribution connecté à leurs points d'accès- pour former un ensemble étendu de services (*Extended Service Set (ESS)*). Un ESS est identifié par un ESSID (ou SSID) qui est une séquence de 32 caractères, qui représente le nom du réseau. Un ESS peut également inclure un IBSS.

La Figure 3.12 page ci-contre résume les différents modes d'interconnexion *Wi-Fi*.

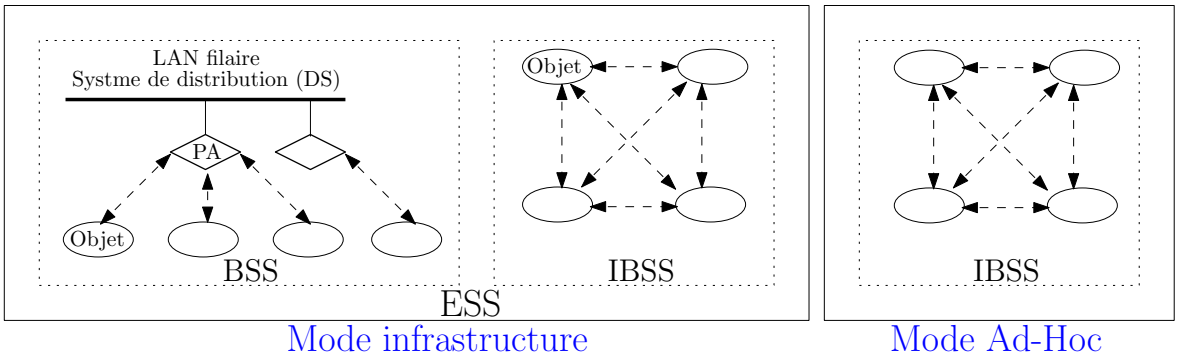


FIGURE 3.12 – Modes d'interconnexion *Wi-Fi*

La sécurité dans le Wi-Fi

D'après [158], les réseaux *Wi-Fi* sont relativement simples à pirater. Un hacker peut facilement exploiter ses vulnérabilités pour accéder aux réseaux filaires considérés comme étant protégés. Les mécanismes et protocoles de sécurité les plus répondus pour *Wi-Fi* sont le *Wired Equivalent Privacy* (WEP) et le *Wi-Fi Protected Access* (WPA). Le standard WEP est connu par sa faiblesse. Selon [25], les mots de passe utilisés par ce protocole peuvent être craqués en quelques minutes en utilisant uniquement un ordinateur portable de base et quelques applications largement disponibles. Ce qui a incité la création des standard WPA1 et WPA2 (WPA version 1 et 2). Le WPA1 était une alternative rapide pour améliorer la sécurité du WEP. Le standard actuel est le WPA2. Dans ce qui suit on va expliquer les standards WEP, WPA1 et WPA2.

Le standard WEP :

Le WEP est un protocole utilisé pour sécuriser les trames IEEE 802.11, il utilise l'algorithme de chiffrement par flux *Rivest Cipher 4* (RC4) [56] pour assurer la confidentialité, et l'algorithme *Cyclic Redundancy Check* (CRC) [146] pour assurer l'intégrité. Selon [27], le WEP repose sur une clé secrète pré-partagée entre les parties communicantes. La sécurisation de la trame de données se fait comme suit :

D'abord, pour assurer l'intégrité en utilisant l'algorithme CRC, on calcule la somme de contrôle $c(\mathrm{M})$ (4 $octets$) du message M, puis on concatène les deux pour avoir le texte en clair (plaintext) $\mathrm{P} = (\mathrm{M}, c(\mathrm{M}))$. Ensuite pour assurer la confidentialité, on chiffre le texte en clair P obtenu dans l'étape précédente avec l'algorithme RC4. D'abord, on choisit un vecteur d'initialisation IV. Ensuite, le RC4 génère ce qu'on appelle une clé de flux (*keystream*) en fonction du vecteur d'initialisation IV et d'une clé K. Cette *keystream* est représentée par $\mathrm{RC4}(\mathrm{IV},\mathrm{K})$. Après on calcule le ou-exclusif ($\oplus$) entre le texte en clair P et la *keystream* $\mathrm{RC4}(\mathrm{IV},\mathrm{K})$ pour obtenir le texte chiffré (ciphertext) $\mathrm{C} = \mathrm{P} \oplus \mathrm{RC4}(\mathrm{IV},\mathrm{K})$. Enfin, on transmet le vecteur d'initialisation IV et le texte chiffré C via la voie de transmission radio.

Remarque : les trames de contrôle, et les trames de gestion sont toujours en texte clair [25].

Pour le déchiffrement des trames, le destinataire inverse juste le processus du chiffrement. D'abord il re-génère la *keystream* $\mathrm{RC4}(\mathrm{IV},\mathrm{K})$, puis établit son $\oplus$ avec le texte chiffré C pour obtenir le texte initial P :

$$\mathrm{C} \oplus \mathrm{RC4}(\mathrm{IV},\mathrm{K}) = (\mathrm{P} \oplus \mathrm{RC4}(\mathrm{IV},\mathrm{K})) \oplus \mathrm{RC4}(\mathrm{IV},\mathrm{K}) = \mathrm{P}$$

Ensuite, afin de vérifier l'intégrité de la trame, il sépare le message M de la somme de

contrôle $c(M)$, recalcule une somme de contrôle $c(M')$ et la compare avec $c(M)$. Au final, uniquement les trames ayant des sommes de contrôle valides qui seront acceptés par le receveur.

Le WEP 64 bits utilise une clé de chiffrement de 40 $bits$ à laquelle est concaténé un vecteur d'initialisation de 24 $bits$. La clé et le vecteur d'initialisation forment ainsi une clé RC4 de 64 $bits$. Au moment où la norme WEP a été rédigée, les restrictions imposées par le gouvernement des États-Unis d'Amérique sur l'export des moyens cryptographiques limitaient la taille des clés. Une fois ces restrictions retirées, les principaux fabricants étendirent le WEP à 128 $bits$ en utilisant une clé de 104 $bits$ [30]. En effet, une clé WEP de 128 $bits$ est saisie comme une suite de 13 caractères ASCII ou 26 caractères hexadécimaux. Chaque doublet hexadécimal représente 8 $bits$ de la clé WEP ($8 \times 13 = 104\,bits$). En ajoutant le vecteur d'initialisation de 24 $bits$, on obtient une clé WEP de 128 $bits$. Aujourd'hui on peut également utiliser des clés WEP de 256 $bits$ et de 512 $bits$ (24 $bits$ réservés au vecteur d'initialisation et le reste pour la clé de chiffrement).

L'une des faiblesse majeure du WEP, c'est qu'il n'inclut pas un protocole de gestion de clés, ce qui fait que si les clés partagées ne soient pas renouvelées régulièrement par les utilisateurs ou par les administrateurs, alors il est possible qu'elles seront piratées (ex. en utilisant des attaques par cryptanalyse). Si un attaquant arrive à avoir le contrôle d'une clé légitime, il peut compromettre le réseau en entier.

Le vecteur d'initialisation est envoyé en texte clair, et a une taille de 24 $bits$ (trop petite), ceci permet de reproduire la même *keystream* en un peu de temps (entre 5 et 7 heurs dans un réseau actif). Ce qui fait que le cassage des *keystream* devient une tache simple pour les hackers (en moins de 17 millions de combinaisons).

En revanche, [157] estime que bien que le WEP n'était pas complètement publié et n'avait pas un bon design technique, il est considéré comme un succès, car il assurait une protection significative pour le *Wi-Fi* lorsqu'il n'y avait aucune autres alternatives.

Le standard WPA v1 et v2 :

Pour corriger les faiblesses du WEP, l'IEEE 802.11 *Task Group I* (TGi) [4] créa le WPA (version 1 et 2). Ce dernier opère sous deux modes : le premier est appelé WPA-PSK (ou WPA Personnel). Il est basé sur les clés pré-partagées, et conçu pour les petits réseaux (réseau d'une maison, d'un bureau, etc) et ne nécessite pas un serveur d'authentification tel que *Remote Authentication Dial-In User Service* RADIUS ou *Authentication, Authorization, Accounting/Auditing* (AAA)[136]. Quant au deuxième -qui s'appelle WPA-802.1X [12] (ou WPA Entreprise)- est destiné aux réseaux d'entreprises et demande l'utilisation d'un serveur d'authentification. WPA-802.1X est plus complexe à installer que WPA-PSK mais il assure plus de sécurité [110]. Les deux mode (WPA-PSK et WPA-802.1X) sont disponibles dans les deux versions du standard WPA.

WPA1 :

Le WPA1 est basé sur un protocole appelé *Temporal Key Integrity Protocol* (TKIP), qui représente un ensemble de mécanismes de sécurité permettant d'assurer la confidentialité et l'intégrité [127]. A partir d'une clé principale, pré-partagée ou gérée par le serveur d'authentification, le WPA1 génère deux types de clés : (1) la première nommée mK (64 $bits$) permet d'assurer l'intégrité de données en générant/vérifiant le *Message Integrity Code* (MIC). (2) La deuxième, appelée K (128 $bits$), permet d'assurer la confidentialité de données [128]. Le WPA1 comprend également un compteur de séquence (un numéro de

4. TGi : a comme objectif l'amélioration de la norme IEEE 802.11 *Medium Access Control* afin d'améliorer la sécurité.

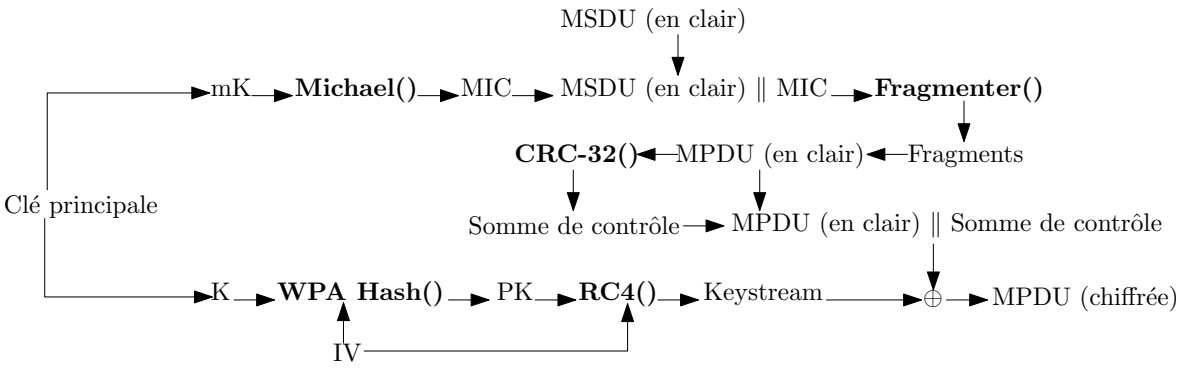


FIGURE 3.13 – Processus d'envoie des messages sécurisés en WPA1

séquence 48 $bits$) pour les vecteurs d'initialisation afin d'éviter les attaques de rejeu [110]. Dans ce qui suit, on va expliquer le mécanisme d'échange sécurisé de messages dans le WPA1.

L'envoie d'un message : lors de l'envoie d'un message, et pour assurer son intégrité, l'expéditeur calcule un MIC (64 $bits$) à partir de mK et une *Mac Service Data Unit* (MSDU [5]) en utilisant l'algorithme *Michael* [6] [31]. La MSDU est concaténée au MIC(3 page 38), le résultat est fragmenté sous forme de *Mac Protocol Data Unit* (MPDU [7]). Ensuite, une somme de contrôle (CRC) de 32 $bits$ est calculée pour chaque MPDU (en utilisant le CRC32 [42]) afin d'assurer son intégrité.

Pour assurer la confidentialité de données, on applique le chiffrement WPA1 sur tous les MPDUs associés à leurs sommes de contrôles. D'abord, on génère ce qu'on appelle un paquet de clé PK à partir d'un vecteur d'initialisation IV de 48 $bits$, de la clé K, et d'une adresse MAC, en utilisant une fonction de hachage spécifique WPA $hash()$. L'IV doit être différent pour chaque MPDU. Comme dans le cas du WEP, le WPA1 utilise l'algorithme de chiffrement par flux RC4. RC4 génère une séquence pseudo aléatoire appelée *keystream* Z à partir de l'IV et de PK, sachant que $Z = Z_1, Z_2, .., Z_L$, où Z_i est une valeur en octets, et $_L$ est la longueur du texte en clair P ($P = P_1, P_2, .., P_L$). Ensuite, on calcule le ou-exclusif ($\oplus$) entre Z et P afin d'obtenir le texte chiffré C ($C = C_1, C_2, .., C_L$). On peut résumer le chiffrement WPA1 comme suit :

$$C = (MPDU \parallel CRC32(MPDU)) \oplus RC4(PK, IV)$$

Les MPDUs chiffrées sont associées à leurs IVs en clair, et le résultat est envoyé au destinataire. Le processus de préparation du message d'envoi du WPA1 est détaillé dans la Figure 3.13. $\parallel$ signifie l'opération de concaténation.

La réception d'un message : le destinataire reçoit une MPDU chiffrée concaténée à un IV en clair. L'IV est comparé avec la valeur de l'IV de la dernière MPDU reçue et acceptée. Si le nouvel IV est inférieur ou égal à l'ancien, alors, la MPDU sera supprimée. Lors de la phase de déchiffrement en WPA1, le receveur génère une *keystream* Z' (identique à Z) à partir de l'IV reçu et PK. Le texte en clair P est obtenu en calculant le $\oplus$ entre C et Z' ($P = C \oplus Z'$). Le destinataire calcule la somme de contrôle à partir de la MPDU reçue, puis la compare avec la somme de contrôle reçue. Si la vérification échoue, alors la MPDU

5. MSDU : paquet de données envoyé par la sous couche liaison de données à la sous couche MAC.
6. Michael est un algorithme adopté par la TGi afin de calculer des hashs, il utilise des clés de 64 $bits$. On peut le comparer avec le HMAC-SHA1 [110]
7. MPDU : paquet de données envoyé par la sous couche MAC à la couche physique

est rejetée, et aucun message d'erreur n'est envoyé à l'expéditeur. Une fois que toutes les MPDUs sont obtenues, elles sont ré-assemblées pour former la MSDU. Le destinataire calcule le MIC à partir de la MSDU reçue et de la clé *m*K en utilisant l'algorithme *Michael*. Si le MIC calculé et celui qui reçu se diffèrent, alors, toutes les MPDUs correspondantes à cette MSDU seront supprimées, et le destinataire envoie un message d'erreur `a MIC failure report frame` à l'expéditeur. En WPA1 la clé *m*K doit être changée si plus de deux messages d'erreurs ayant un rapport au MIC sont envoyés à l'expéditeur dans moins d'une minute. Quand la MSDU est acceptée, la valeur de l'IV est mis à jour, et remplacé par l'IV*s* le plus grand trouvé dans les MPDUs. Le processus de la réception du WPA1 est détaillée dans la Figure 3.14.

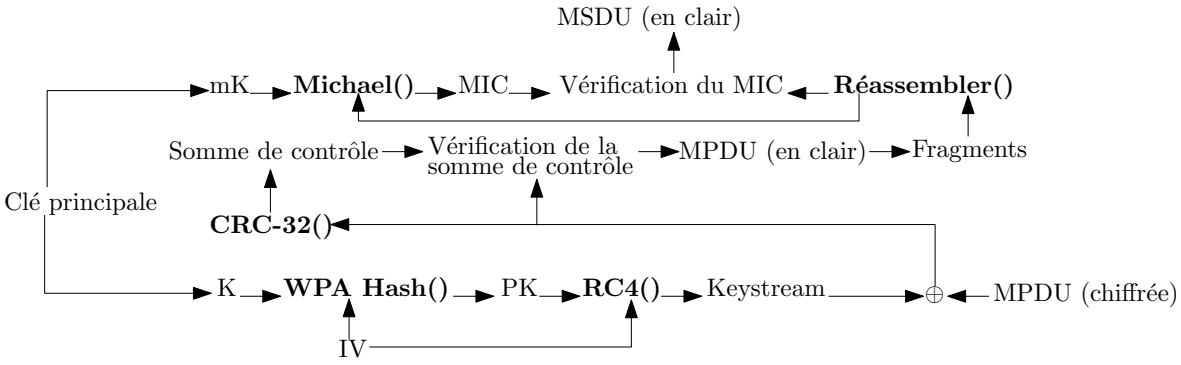


FIGURE 3.14 – Processus de réception des messages sécurisés en WPA1

WPA2 :

Le WPA2 est le successeur du WPA1 qui a été ratifiée en juin 2004 [16].

WPA2 permet d'établir une connexion sécurisée entre deux entités communicantes ou plus grâce à un protocole appelé *Counter Mode Cipher CBC-MAC Protocol* (CCMP) -voir 3.2.3.1 page 54- défini dans le standard IEEE 802.11i. Il représente une alternative considérée comme plus sûre que le TKIP. Ce standard est basé sur AES. Ce dernier, comparé au RC4 qui est utilisé par le WEP et le WPA1, assure une sécurité robuste, avec des performances meilleures. Afin de créer un canal de communication sécurisé, le WPA2 nécessite la réalisation de 4 phases [15]. (1) Une entité voulant communiquer doit d'abord négocier la politique de sécurité (méthode d'authentification, gestion de trafic, etc) avec le point d'accès. (2) Une fois qu'ils se mettent d'accord sur les paramètres supportés par les deux parties, ils doivent s'authentifier mutuellement et (3) générer des clés de session en utilisant deux types de négociations, qui sont le *4 way Handshake* et le *Group Key Handshake*.

Le 4 Way Handshake effectue les tâches suivantes :

— La génération des clés transitoires par paire (*Pairwise Transient Key* (PTK)) à partir de la clé principale (ou maîtresse) par paire (*Pairwise Master Key* (PMK)) -qui est dérivée à son tour de la clé principale (*Master key* (MK)) dans le mode *WPA Entreprise*, ou à partir de la PSK dans le mode *WPA Personnel*. Les PTK sont utilisées pour générer d'autres clés unicast et/ou des clés transitoires par groupe (*Group Temporal Key* (GTK), utilisée pour dériver des clés broadcast et multicast) ;

— A partir d'une PTK, le 4-Way Handshake permet de générer des clés de confirmation de 128 $bits$ (*Key Confirmation Key* (KCK), utilisées pour vérifier l'intégrité des trames pendant la négociation) et des clés de chiffrement de 128 $bits$ (*Key Encryption Key* (KEK), utilisées pour chiffrer les trames pendant cette phase. Et à partir de

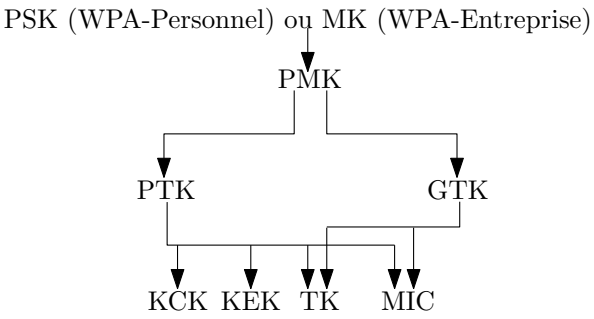


FIGURE 3.15 – Hiérarchie des clés en WPA2

PTK ou GTK, il permet de générer des clés temporaires de 128 $bits$ (*Temporal Key* (TK), utilisées pour sécuriser les données du trafic, et générer des MIC). La Figure 3.15 résume le processus de génération de l'ensemble des clés utilisées ;

— Le 4 Way Handshake confirme une suite de chiffrement (*ciphersuite*[8]) sélectionnée parmi une liste de suites appelée *ciphersuites*[9].

Le *Group Key Handshake* est utilisé pour renouveler le GTK.

(4) Enfin, toutes ces clés sont utilisées par le protocole CCMP, afin d'assurer la confidentialité et l'intégrité de données[15].

Afin d'assurer la confidentialité et l'intégrité de données, le protocole CCMP applique l'algorithme AES CCM [87]. CCM est un mode d'opération[10] qui permet d'assurer à la fois les service de confidentialité et d'intégrité. Pour plus de détails sur AES CCM voir section 7.1.1 page 91 et section 7.1.3 page 94.

Le WPA2 introduit en outre des mécanismes de protection de trafic de contrôle permettant d'éviter un certain nombre d'attaques de déni de service[11] (Denial of service (Dos)) ou déni de service distribué (Distributed Denial of service (DDos)) auxquels tous les réseaux sont actuellement vulnérables.

3.2.4 Réseaux personnels sans fil (WPAN)

Les réseaux personnels sans fil représentent les réseaux sans fil à faible portée (une dizaine de mètres) avec un débit qui peut atteindre 1 M$bits$/S. Les technologies de communication les plus prometteuses, utilisées pour créer des WPAN, sont celles définies dans les standards IEEE 802.15.1 [99] et IEEE 802.15.4 [29]. *Bluetooth*, défini par le standard IEEE 802.15.1, est basé sur un système radio sans fil conçu pour les appareils à courte portée pour remplacer les câbles des périphériques d'ordinateurs, tel que les souris, claviers, manettes et imprimantes, etc. En revanche, le standard IEEE 802.15.4, qui connais un grand succès grâce à son optimalité, sa fiabilité et sa robustesse contre les interférences, se focalise principalement sur la spécification des couches basses (physique et liaison de données) nécessaires pour la création des réseaux de capteurs sans fil (WSN). Les WSNs sont utilisés afin de gérer, en temps réel, un environnement donné. Un WSN permet de récolter des données via des capteurs, les traiter, puis agir si c'est nécessaire

8. Ciphersuite : est une chaîne de caractère définissant un algorithme d'authentification et d'échange de clés, un algorithme de chiffrement symétrique, et une fonction de hachage
9. Ciphersuites : est une liste comprenant un ensemble de *ciphersuite*
10. Mode d'opération : est la manière de traiter les blocs de texte clairs et chiffrés au sein d'un algorithme de chiffrement par bloc.
11. Une attaque de déni de service est une attaque ayant pour objectif de rendre un service non disponible

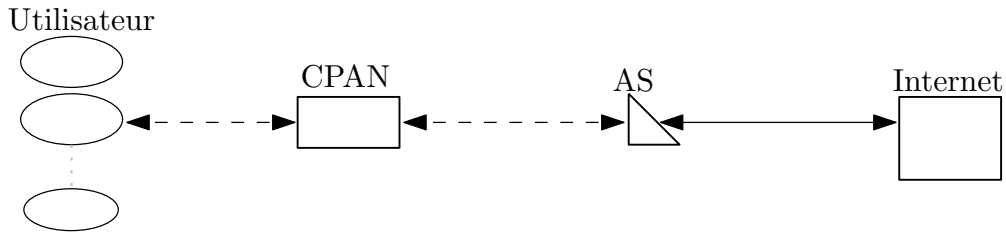


FIGURE 3.16 – Une architecture d'un réseau *6LoWPAN*

en utilisant des actionneurs. Les technologies de communication WSN utilisent généralement les bandes ISM (voir note 3 page 51).

Dans ce qui suit, on s'intéresse uniquement aux technologies IEEE 802.15.4, car ce standard est conçu pour supporter des objets caractérisés par leur faible débit binaire, faible coût, et limitation en terme de capacité de calcul, de mémorisation et en énergie. D'abord on commence par étudier le *6LoWPAN* puis le *ZigBee*, et on termine avec une nouvelle technologie innovante qui s'appelle *OCARI*, qu'on a exploité pour valider nos travaux de recherche et y a déployé nos solutions en terme de sécurité.

3.2.4.1 6LoWPAN

D'après [95], 6LoWPAN est une spécification d'un réseau personnel sans fil à faible puissance. Il peut être déployé avec une topologie en mode étoile ou maillage. Il est basé sur le protocole IPv6, ce qui lui permet d'avoir plusieurs avantages, tel que la possibilité d'utiliser des infrastructures et technologies IP existantes qui sont testées et approuvées. On outre, les objets basée IP, peuvent être connectés facilement à d'autres réseaux IP, sans avoir besoin d'entités intermédiaires comme les passerelles (pour plus de détails voir [95]).

La sécurité dans 6LoWPAN

Comme dans le cas de la plupart des technologies IEEE 802.15.4, afin d'assurer la confidentialité et l'intégrité de données, 6LoWPAN utilise le standard AES-CCM* [12], en revanche il ne définie pas une méthode spécifique pour l'authentification, ni pour la gestion des clés [95]. Un travail intéressant était proposé par [95] définit une méthode d'authentification qui utilise le protocole *Extensible Authentication Protocol Generalized Pre-Shared Key* (EAP-GPSK), qui est basé sur la cryptographie symétrique.

Authentification et mécanisme d'échange des clés :

La Figure 3.16 montre une architecture réseau du *6LoWPAN* définie dans [95]. L'architecture réseau est composé de (1) un Device qui communique avec (2) un coordinateur de réseau personnel *Personal Area Network Coordinator* (CPAN). Le CPAN peut être un routeur, une passerelle, ou n'importe quel autre équipement ayant la possibilité de router les paquets de données entre différents réseaux. Cette architecture contient également (3) un serveur d'authentification *Authentication Server* (AS) qui sert à déployer une ou plusieurs fonctionnalité d'Authentification, Autorisation, Comptabilité/Audit (*Authentication, Authorization, Accounting/Auditing* (AAA). La Figure 3.17 décrit le mécanisme d'authentification mutuelle proposé pour le *6LoWPAN*. D'abord (1) le Device envoie un message de démarrage du protocole d'authentification extensible (message de démarrage EAP) au

12. Le CCM* est une extension du mode d'opération CCM qui met des restrictions sur les tailles de quelques paramètres afin d'éviter certaines failles de sécurité.

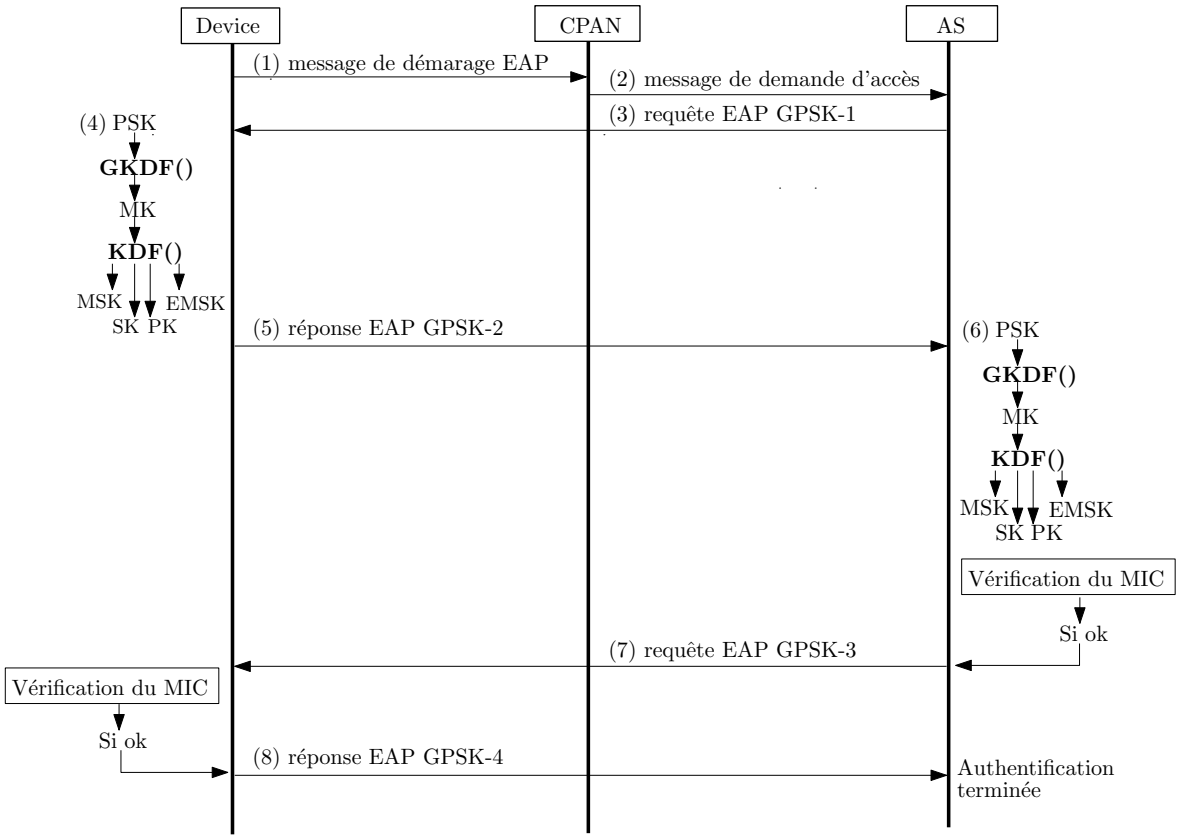


FIGURE 3.17 – Mécanisme d'authentification mutuelle EAP-GPSK (proposé pour *6LoWPAN*)

CPAN. (2) Une fois que le message reçu, le CPAN envoie un message de demande d'accès à l'AS. (3) Ce dernier renvoie à son tour une requête de clé pré-partagé générale EAP (requête EAP GPSK-1) au Device via le CPAN. Ce message contient, un nombre aléatoire de 16 *octets*, l'identifiant de l'AS, qui est une adresse IPv6 de 16 *octets*, et une liste de ciphersuites (voir note 8 page 57) contenant la liste des algorithmes supportés par le AS. (4) Le Device génère aussi un nombre aléatoire, puis à l'aide d'une fonction de génération de clé générale (generalized key derivation function, GKDF) choisie parmis les fonctions proposées dans la liste des ciphersuites, il génère une clé principale (MK) de 16 *octets* en utilisant une clé pré-partagée (PSK), l'identifiant de l'AS, le nombre aléatoire de l'AS, son identifiant, et son nombre aléatoire. Ensuite, à partir de MK, et en utilisant la fonction de génération de clé *Key Derivation Function* (KDF), le Device génère une clé de session principale *Master Session Key* (MSK), une clé principale de session étendue *Extended Master Session Key* (EMSK), une clé de session *Session Key* (SK), et une clé de protection de de donnée *Protected data encryption Key* (PK). (5) Le device envoie une réponse EAP GPSK-2 à l'AS. Ce message contient la ciphersuite sélectionnée, l'identifiant du Device, le nombre aléatoire du Device, une charge utile (PD) et un MIC. L'algorithm du calcul du MIC dépond de la ciphersuite choisi (ex. HMAC-SHA256[126]). Le MIC est calculé afin de protéger l'intégrité des données envoyées. (6) En recevant la réponse du Device, l'AS génère MSK, EMSK, SK, et PK, puis vérifie le MIC. Si ce dernier est valide, alors la première authentification se termine avec succès. (7) Après, une requête EAP GPSK-3 est envoyée de l'AS vers le Device. Cette requête contient un nouveau nombre aléatoire généré par AS, le nombre aléatoire du Device, optionnellement un PD, et un nouveau MIC. (8) Une fois que le Device valide le MIC et authentifie l'AS, il envoie une réponse GPSK-4 contenant, optionnellement un PD, et le MIC du message, et cette opération d'authentification mutuelle se termine une fois que ce message est reçu.

Confidentialité et intégrité de données :
Afin de protéger les données échangées, [95] recommande l'utilisation du standard AES-CCM*, qui est un algorithme qui assure à la fois les services d'intégrité et confidentialité. On étudiera ce standard en détails dans le chapitre 7 page 91.

3.2.4.2 ZigBee

Le *Zigbee* est une technologie WPAN à faible débit et à faible consommation de ressources (énergie, calcul, et mémorisation) qui peut être déployé avec une topologie en mode étoile ou maillée. Il était inventé par la *ZigBee Alliance*[13] [131], qui générait les premières spécifications en 2004. Les deux couches basses de cette technologie sont basées sur le standard IEEE 802.15.4, qui lui permet d'exploiter les bandes de fréquences ISM (voir note 3 page 51) pour l'échange de données. En outre, il possède d'autres bandes normalisées qui sont la 915 Mhz aux *États Unis* et la 868 Mhz en *Europe* [147]. Afin de pouvoir accéder au canal de communication, au niveau de la sous-couche MAC, les objets *ZigBee* utilisent le mécanisme *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) [55]. Ce dernier permet d'optimiser la consommation d'énergie en limitant le nombre de transmission des messages perdus dû aux collisions. La *ZigBee Alliance* définit également la couche réseau qui se charge (1) du routage des messages et de (2) la jointure et dis-jointure des objets au réseau, et (3) assure les services de découverte et maintenance [147]. Et enfin, la couche applicative fournit divers services dont l'authentification mutuelle.

La sécurité dans ZigBee
La sécurité est déployée au niveau de la couche application et la couche réseau. Chaque couche est responsable de sécuriser l'échange de ses données.

Mécanisme d'échange de clés et authentification mutuelle :
Échange de clés :
Le protocole de sécurité du *ZigBee* est basé sur (1) une clé de liaison de donnée de 16 $octets$ (*link key*), partagée entre 2 objets, et utilisée uniquement au niveau de la couche application afin de sécuriser les communications en mono-diffusion (unicast). Et une clé de réseau de 16 $octets$ (*network key*), partagée entre tous les objets du réseau, utilisée à la fois au niveau de la couche application et au niveau de la couche réseau afin de protéger la communication en mode diffusion (broadcast). D'après [131], un objet doit acquérir une *link key* via la pré-installation, l'établissement de clé ou le transport de clé. Un objet doit acquérir une *network key* via la pré-installation ou le transport de clé.

— La pré-installation : elle peut être réalisée au moment de la fabrication de l'objet par l'usine ;

— L'établissement de clé : la sous couche support d'application (APS[14]) permet de générer et partager la *link key* entre deux dispositifs *ZigBee*. Ce service est précédé par une étape d'approvisionnement de confiance. Les informations de confiance (une

13. *ZigBee Alliance* représente une communauté industrielle fondée en Août 2001. Elle regroupe plusieurs grandes entreprises et organisations telque *HUAWEI, Comcast, ARM, Amazon lab*, etc.

14. APS : est une sous couche qui représente une interface entre la couche réseau et la couche application. Elle fournit un ensemble de services qui sont utilisés à la fois par le *ZigBee device object* (ZDO) et par les objets applicatifs définis par le fabricant.

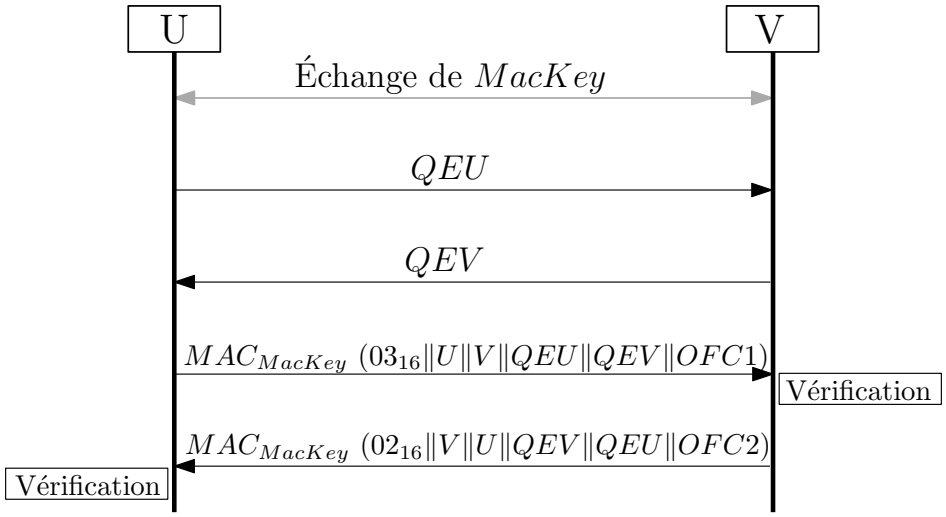


FIGURE 3.18 – Mécanisme d'authentification mutuelle du *ZigBee*

clé principale (*master key*)), qui représentent un point de départ pour l'établissement de la *link key*. Ensuite, le protocole d'établissement de clé doit suivre 3 étapes, qui sont (1) l'échange des informations éphémères, (2) l'utilisation de ces informations pour la génération de la *link key*, et (3) la confirmation que la clé a bien été calculée ;

— Le transport de clés : ce service fournit des moyens sécurisés et non sécurisés pour transporter une clé vers un ou plusieurs objets. Le moyen sécurisé permet de transporter une clé d'une source (centre de confiance) vers un objet et de la protéger par des moyens cryptographiques. Le moyen non sécurisé sert à transférer une clé en clair.

Authentification mutuelle :
D'après la *ZigBee Alliance* [131], le mécanisme d'authentification mutuelle entre 2 objets (U et V) est basée sur une clé partagée et quelques informations échangées. L'opération d'authentification, décrite par la Figure 3.18, suit les étapes suivantes :

Une fois que les clés sont partagées, à l'étape (1) et (3) l'initiateur U (ex. device) et le répondeur V (ex. CPAN) échangent des challenges [15] (QEU et QEV) d'une taille de 16 $octets$. Ensuite, dans (2) et (4) chaque entité calcule un $hash$ à partir d'une fonction de hachage [16] (SHA1 [53]), en utilisant comme données d'entrée les 2 challenges, son propre compteur de trame sortante *Outgoing Frame Counter* (OFC), son identifiant, l'identifiant du correspondant, une valeur (03_{16}, 02_{16}) et une MAC clé ($MacKey$). La $MacKey$ est la clé réseau (*network key*) qui est une clé de diffusion partagée entre tous les objets du même groupe. Après, dans (5) et (7) les deux communicants échangent leurs $hash$ associés de leurs OFCs respectifs (($hash_1$, OFC_1) pour U et ($hash_2$, OFC_2) pour V). (6) U est authentifié si V valide $hash_1$ ($hash_1 = hash'_1$). Et enfin, (8) U exécute exactement la même opération de vérification, et si $hash_2$ est égale à $hash'_2$ (calculé par U), alors l'authentification de V est réussie. Ainsi, l'authentification mutuelle des deux objets est établie.

Confidentialité et intégrité de données :
Les services de confidentialité et d'intégrité sont assurés grâce au chiffrement authenti-

15. Un challenge : est une séquence aléatoire d'$octets$
16. Une fonction de hachage : est une fonction irréversible, avec qui, à partir du résultat, on peut pas revenir au données initiales.

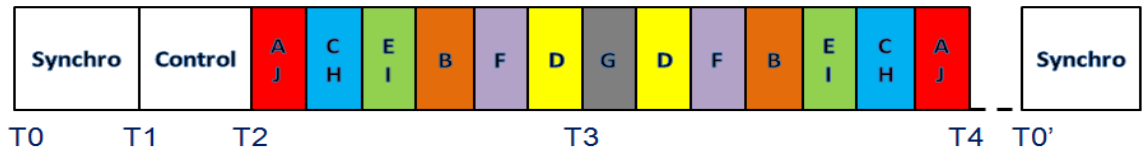


FIGURE 3.19 – Cycles d'*OCARI*

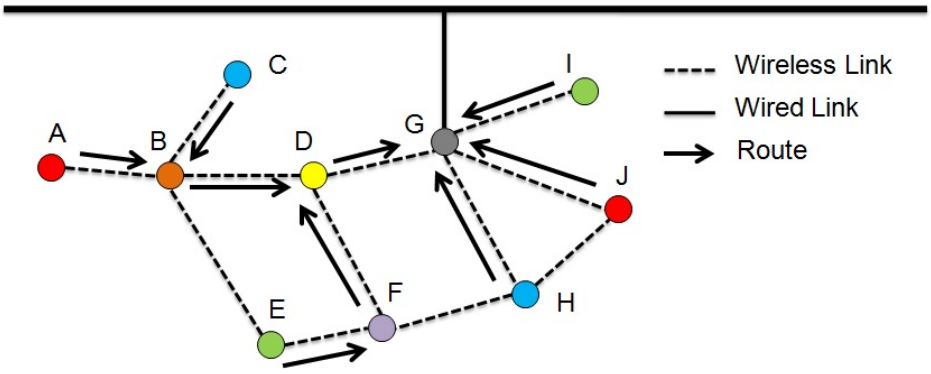


FIGURE 3.20 – Topolologie d'un réseau *OCARI*

fié déployé au niveau de la couche application et la couche réseau. En effet, les messages sont doublement protégés au niveau des deux couches séparément, en utilisant le standard AES-CCM*. Une description complète du fonctionnement d'AES-CCM* est présentée dans le chapitre 7 page 91.

3.2.4.3 *OCARI*

Optimization of Communication for Ad hoc Reliable Industrial networks (*OCARI*) est une technologie de communication WSN. Elle est désignée spécialement pour les environnements industriels afin de pouvoir supporter des applications tel que la supervision des centrales nucléaires, la maintenance conditionnelle des équipements et des machines, etc. D'après [10], l'objectif d'*OCARI* est d'améliorer le *ZigBee* en ajoutant (1) le support d'une sous-couche MAC déterministe pour des communications limitées par le temps, (2) le support d'une stratégie de routage optimale et économique en terme d'énergie pour maximiser la durée de vie du réseau, et (3) le support de la mobilité de certains objets spéciaux.

La couche physique d'*OCARI* est basée sur le standard IEEE 802.15.4. Cette couche est connue par sa fiabilité dans les environnements industriels et sa robustesse contre les interférences. En revanche, *OCARI* développe sa propre sous-couche MAC, libellée MaCARI, afin de fournir un accès déterministe au médium.

La Figure 3.19 montre le cycle de fonctionnement d'*OCARI* correspondant à la topologie illustrée par la Figure 3.20.

Pour l'échange des messages de contrôle, MaCARI utilise la méthode CSMA/CA [55] combinée avec la méthode *Time Division Multiple Access* (TDMA)[166]. Quant aux messages de données, elle utilise un algorithme de coloration d'objets à trois sauts avec réutilisation spatiale des intervalles de temps.

Comme décrit dans la Figure 3.20, MaCARI utilise un arbre de routage pour collecter des données selon un calendrier déterministe afin d'assurer une faible consommation d'énergie. La couche réseau appelée OPERA utilise un protocole de routage appelé EOLSR. Ce dernier représente une extension du protocole de routage *Optimized Link*

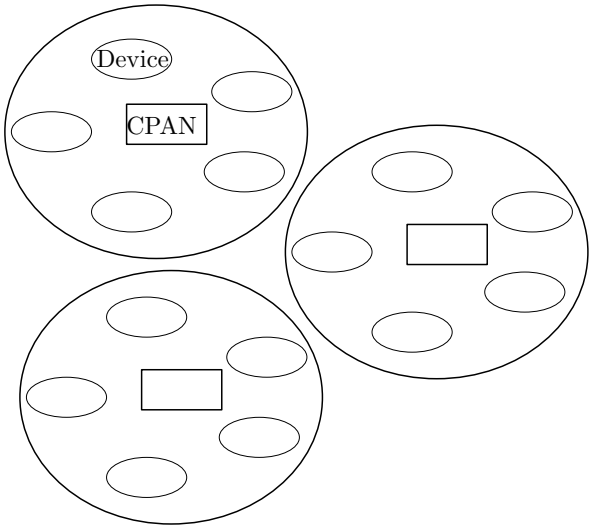


FIGURE 3.21 – Une architecture simplifiée d'un réseau *OCARI*

State Routing (OLSR) [36]. EOLSR prend en compte l'énergie résiduelle des objets. En effet, il sélectionne le meilleur itinéraire qui consomme le minimum d'énergie lors de la transmission/réception des paquets. Pour économiser plus d'énergie, la couche réseau utilise un autre protocole appelé OSERENA. Ce protocole permet d'attribuer une couleur à chaque objet. Les objets transmettent leurs messages de données dans un intervalle de temps correspondant à leurs couleurs.

La Figure 3.21 représente une architecture simplifiée d'un réseau *OCARI*. Elle est composée d'un ensemble de groupe d'objets ayant chacun un identifiant unique (UI) d'une taille de 8 $Octets$. Chaque groupe est géré par un *Personal Area Network Coordinator* (CPAN). Contrairement au reste des objets (devices), qui sont généralement des capteurs ou des actionneurs ayant des capacités limitées en termes de calcul, de mémorisation et en énergie, les CPANs sont des équippements sans contrainte hardware. Afin de se joindre à un groupe, un device doit envoyer une *requête d'association* au CPAN, et reçoit une *réponse d'association*.

La sécurité dans *OCARI*

En raison de la nouveauté d'*OCARI*, la version actuelle du protocole ne définie pas des services de sécurité. Dans ce travail, nous proposons un protocole de sécurité robuste, léger, rapide et économique en énergie, qui est désigné spécialement pour être déployé sur des objets ayant des capacités limités. On a implémenté ce protocole de sécurité sur la plateforme d'*OCARI*, et on l'a déployé sur des vraies capteurs. La partie III page 77 explique en détail notre notre approche, qui est conçue pour sécuriser *OCARI* et l'Internet des Objets d'une manière générale.

3.3 Discussion

Dans ce chapitre nous avons présenté les différentes technologies de communications sans fil utilisées dans le cadre de l'Internet des Objets. Nous nous sommes focalisés principalement sur leurs approches de sécurité, et plus précisément sur leurs mécanismes d'authentification et gestion de clés. Toutes ces approches ont été proposés pour des technologies de communication ayant des caractéristiques similaires. Nous avons établi cette étude sur les solutions déjà existantes afin d'en proposer une plus complète et plus adap-

tée aux exigences de l'IoT. Pour conclure ce chapitre, dans cette section, nous allons faire une comparaison entre ces différents protocoles de sécurité afin de pouvoir définir un modèle de sécurité performant.

Technologie		Authentification mutuelle	Confidentialité	Intégrité	Protection contre			Gestion de clés	Flexibilité	La confidentialité persistante	Resilience	Nombre de message (authentification & gestion de clés)
					Rejeu	*Cryptanalyse*	*Dos/DDos*					
LoRaWAN (voir 3.2.1.1 page 36)		✓	✓	✓	✗	✓	✗	✓	✗	✓	✓	2
Réseau LTE (voir 3.2.1.2 page 40)		✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	5
Réseau satellitaire (voir 3.2.1.3 page 44)		✓	✓	✗	✓	✗	✗	✓	✗	✓	✓	5
WiMAX mobile (voir 3.2.2.1 page 47)	avec certificats X509	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	"$\geqslant$ 2" + 3
	avec EAP	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	"$\geqslant$ 4" + 3
Wi-Fi	WEP (voir 3.2.3.1 page 53)	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	4
	WPA1 (voir 3.2.3.1 page 54)	✓	✓	✓	✗	✗	✓	✓	✗	✓	✓	"$\geqslant$ 10"
	WPA2 (voir 3.2.3.1 page 56)	✓	✓	✓	✗	✗	✓	✓	✗	✓	✓	"$\geqslant$ 10"
6LoWPAN (voir 3.2.4.1 page 58)		✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	6
ZigBee (voir 3.2.4.2 page 60)		✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	"$\geqslant$ 4"

TABLEAU 3.2 – Sécurité et technologies de communications sans fil IoT

Les différentes notions de sécurité dans le Tableau 3.2 sont décrites dans le chapitre 2. "✓" et "✗" signifient respectivement que le service est fournie ou pas, mais n'évalue pas sa qualité et sa performance. On considère qu'un protocole est flexible lorsqu'il n'induit pas une intervention manuelle lors d'une association d'un nouveau device. Autrement dit, lorsqu'on rajoute un objet (initiateur) à un réseau, l'authentification et l'association de ce dernier ne doivent pas inclure une mise à jour au niveau de son répondeur. Par exemple, on considère que le protocole de sécurité de *LoRaWAN* comme étant inflexible car lorsqu'on rajoute un nouvel objet A à un group de n utilisateurs partageants leurs clés avec un serveur gérant B, alors on est obligé de mettre à jour B en lui installant la clé $n+1$ de A. Dans la partie III, nous proposons un nouveau mécanisme qui permet de résoudre ce problème. Cette solution est conçue principalement pour les architectures qui utilisent des coordinateurs (ex. *LoRaWAN*, Réseau LTE, 6LoWPAN, *OCARI*, etc). Notre approche assure une grande flexibilité lors d'une association d'un nouveau device au réseau.

Dans ce qui suit A représente l'initiateur d'authentification, B est le répondeur d'authentification, et C un utilisateur malicieux.

Protocole de sécurité de LoRaWAN :
ce protocole permet d'établir un canal qui protège les données échangées, et nécessite

uniquement l'utilisation de 2 messages pour effectuer une opération d'authentification. Cependant, l'utilisation des nonces aléatoires générés par l'initiateur d'authentification au lieu du répondeur d'authentification peut engendrer une faille de sécurité. En effet, cette faille peut être exploitée pour générer une attaque de rejeu qui engendre un déni de service. On peut définir un scénario d'attaque possible comme suit :

[Msg_1 : A → B] A$_{id}$, $nonce_1$, {A$_{id}$, $nonce_1$}sig_{K} # B authentifie A puis génère sk(s)
[Msg_2 : B → A] {B$_{id}$, $nonce_2$}enc_{K}, {B$_{id}$, $nonce_2$}sig_{K} # A authentifie B puis génère sk(s)
[Msg'_1 : C → B] Msg_1 # B authentifie A puis génère sk(s)
[Msg_3 : B → C| A] {B$_{id}$, $nonce_3$}enc_{K}, {B$_{id}$, $nonce_3$}sig_{K}
(A et B ne possèdent plus les mêmes clés de sessions "sk(s)")

Msg_x est le message numéro x, sk(s) est/sont la/les clé(s) de session, l'expression {}sig_{K} représente le MIC calculé à partir des données qui sont à l'intérieur des accolades et qui est signé par la clé $_{\mathrm{K}}$, et {}enc_{K} pour le chiffrement de données. D'abord A envoie Msg_1 pour s'authentifier auprès de B. Si l'authentification réussit, alors B génère des clés de session qui vont être utilisées pour sécuriser les données échangées entre A et B. Á la réception du Msg_2 A déchiffre le message, authentifie B, puis calcule les clés de session.

Lorsque C renvoie à B une copie du Msg_1 via Msg'_1, il provoque une opération d'association. Par conséquent, comme le MIC est valide, B l'accepte et recalcule les clés de session puis envoie Msg_3 à A. C interrompe Msg_3. De ce fait, A et B ne possèdent plus les mêmes clés de session, donc ils ne peuvent plus communiquer.

On peut imaginer un autre scénario où C génère une attaque Dos/DDos en envoyant un grand nombre de Msg'_1, l'opération d'authentification se répétera autant de fois que le nombre de requête envoyées, et même si ces requêtes viennent d'un utilisateur malicieux, elles seront acceptées par B, et ainsi inciter A et B à refaire le processus d'authentification autant de fois que C le veut.

Protocole de sécurité du LTE :
ce protocole permet de résoudre les failles trouvées dans le protocole de sécurité de *LoRaWAN*. En effet, pour l'authentification de A il faut avoir un numéro aléatoire (challenge) généré par B. Quant à l'authentification de B, elle est basée sur un jeton d'authentification. Ce jeton est basé sur un numéro de séquence (SQN) qui le protège contre les attaques de rejeu. Cependant la génération d'un grand nombre de clés (pour éviter les attaques par cryptanalyse) exige d'une quantité non négligeable de mémorisation, et consomme de l'énergie. En outre, d'après des recherches effectuées par [60] le standard d'intégrité de données (128-EIA3) - utilisé pour sécuriser cette technologie - comprend des failles de sécurité.

Protocole de sécurité du réseau satellitaire :
l'authentification mutuelle est basée sur le standard de chiffrement de donnée (Data Encryption Standard, DES) qui est, d'après [86], vulnérable à certaines attaques par cryptanalyse. De plus, l'intégrité des messages n'est pas assurée. Par conséquent, C peut modifier n'importe quel message échangé entre A (UE) et B (NCC) sans que l'une des deux entités légitimes le détecte. Plus grave encore, si le message contenant les nouvelles données (K$'_{md}$, T$'_{id}$) envoyé par le NCC est interrompu à cause d'une perturbation dans le réseau ou par C, alors A et B ne peuvent plus communiquer, car contrairement à B qui a mis à jour ses données (K$'_{md}$, T$'_{id}$), A possède toujours les anciennes données.

Protocole de sécurité du WiMAX mobile :
dans la phase autorisation, la durée de grâce, autorisant temporairement une ancienne clé d'être acceptée lors d'une mise à jour de clés, permet de résoudre partiellement le problème de perturbation de traffic trouvé dans le mécanisme d'échange de clés proposé pour les réseaux satélitaires. Cependant le fait d'avoir une phase d'authorisation (3 messages) en plus d'une phase d'authentification augmente le nombre de messages échangés et les traitements effectués. En plus, la méthode d'authentification basée sur l'algorithme RSA et les certificats X509 n'est pas adaptée aux objets limités, car elle exige beaucoup de ressources.

Procole de sécurité du Wi-Fi (WEP) :
ce protocole est léger est peut être déployé sur des objets limités. Cependant, ce protocole souffre d'un grand nombre de problème de sécurité. [155] définit différents types d'attaques qui ont été fait sur le WEP, des attaques de rejeu, de falsification d'authentification, de cryptanalyse sur les données chiffrées (KoreKs chopchop attack), etc. L'algorithme de chiffrement utilisé (RC4) est vulnérable contre plusieurs attaques tel que celles décrites dans [113] et dans [108]. Le WEP n'inclut pas un mécanisme de gestion de clés et la sécurisation des échanges se base directement sur les clés pré-partagées, et si un attaquant arrive à obtenir les clés secrètes d'un objet [153], alors ce dernier, voir tous le réseau, peut être compromis.

Procole de sécurité du Wi-Fi (WPA version1 et version2) :
ce protocole assure la plupart des services de sécurité. Bien que la version 2 utilise des algorithmes cryptographiques plus robustes que ceux utilisés dans l'ancienne version, cependant le WPA est vulnérable à certaines attaques de cryptanalyse [156] et de falsification [127]. [159] a démontré que WPA est vulnérable de l'attaque par réinstallation de clé baptisée *Krack*. Cette attaque est réalisée en manipulant et en rejouant les messages échangés dans la phase Handshake. Cela réinitialise les paramètres (ex. nonces) associés à la clé. Par conséquent, cette opération crée une faille qui permet de rejouer des messages, de décrypter des informations, et dans certains cas de falsifier des données. Enfin le WPA2 a besoin de beaucoup de ressources matériel (hardware), nécessite la génération de plusieurs clés, et l'échange d'un grand nombre de messages ($\geqslant 10$) pour l'authentification et l'association d'un utilisateur qui peut être très couteux.

Protocole de sécurité du 6LoWPAN :
le protocole proposé pour l'authentification (voir
3.2.4.1 page 58) est léger et n'utilise pas des algorithmes compliqués, cependant d'après [118], il est possible d'effectuer une attaque Dos sur les utilisateurs voulant s'associer au réseau, voici les étapes de l'attaque ($_{sk}$ est la clé de session) :

[Msg_1 : B → A] B_{id}, $nonce_1$
[Msg_2 : A → B] A_{id}, B_{id}, $nonce_1$, $nonce_2$, MIC_{sk}
[Msg'_1 : C → A] B_{id}, $nonce'_1$ # A choisi un nouveau $nonce'_2$ et génère une nouvelle clé de session $_{sk'}$
[Msg_3 : B → A] B_{id}, A_{id}, MIC_{sk}
(Pour calculé le MIC, A utilise $_{sk'}$ par conséquent la vérification du MIC et l'authentification de B échouent).

En outre, le nombre de messages échangés et les clés générées consomment une quantité importante d'énergie.

Protocole de sécurité du ZigBee :
son mécanisme d'authentification est basé sur la méthode du challenge/réponse définie par la RFC1994 [145]. Cette méthode est connue par sa légerté et sa robustesse, et ne nécessite pas beacoup de ressources. Son principe de fonctionnement se résume par le faite qu'une partie présente un défi (challenge) et une autre partie doit fournir une réponse valide (calculée à partir d'un secret partagé et une fonction de hachage) pour être authentifiée. Grâce à ses performances, cette méthode est très adaptée au sysème IoT, cependant le mécanisme de gestion de clé dépolyé par le protocole de sécurité du *ZigBee* [131] limite sa fléxibilité et ses performances, et engendre des failles de sécurité. Dans ce qui suit, nous allons expliquer en détails ces problèmes de sécurité :

— l'architecture *ZigBee* est composée d'un enssemble de groupes de devices. Chaque groupe est géré par un coordinateur. Pour associer un nouveau device à un groupe, il faut établir une opération d'authentification entre le device et le coordinateur (comme décrit dans Figure 3.18 page 61) en utilisant la clé MAC ($MacKey$). Cette clé est la clé réseau (*network key*) active [131]. Autrement dit un nouveau device s'authentifie auprès du coordinateur en utilisant une clé de diffusion partagée avec tous les autres devices du réseau. Ceci représente une grande faille de sécurité, car n'importe quel device interne peut usurper l'identité d'un autre device d'un même group. Pour plus d'explication voici un sécnario d'attaque :

[Msg_1 : C → B] A_{id}, $challenge_1$
[Msg_2 : B → A] B_{id}, $challenge_2$
[Msg'_2 : B → C] B_{id}, $challenge_2$ # C capture une copie du Msg_2
[Msg_3 : C → B] A_{id}, $hash(challenge_1|challenge_2, MacKey)$
(Et l'authentification de l'attaquant C se termine avec succès)

A représente le U sur le schéma de la Figure 3.18 page 61, B est le coordinateur (V), et C est n'importe quel attaquant interne possédant la clé de diffusion ($MacKey$).

D'abord C, en usurpant l'identité de A, envoie une *requête d'association* (Msg_1) à B, ensuite capture une copie du Msg_2 (Msg'_2), et enfin, il calcule un $hash$ avec $MacKey$ et l'envoie à B (Msg_3), et l'authentification de C est réussie.

On peut également imaginer un autre scénario d'attaque, où l'attaquant usurpe l'identité de B, par conséquent A peut être associé à un coordinateur malicieux ;

— comme expliqué dans la section 3.2.4.2 page 60 un objet peut avoir une clé de diffusion via une pré-installation ou un transport de clé. (1) La pré-installation d'une clé de diffusion peut être faite lors de la fabrication d'un objet. Par conséquent un objet appartenant à un groupe ne peut pas être associé à un autre groupe sans l'intervention du fabricant ou d'un administrateur qui doivent renouveler les clés de l'objet, ce qui empêche la mobilité de ce dernier. (2) Le transport de clé de diffusion se fait via une communication établie entre l'objet et une source de confiance appelée *Trust Center* (TC), qui est généralement le coordinateur. Cette communication peut être (a) non chiffrée (en clair), ce qui induit à des attaques de recouvrement de clés [164], ou (b) chiffrée en utilisant une clé mono-diffusion dérivée de la *link key*. Cette dernière doit être partagée entre les deux objets communicants. Ce qui freine la mobilité des objets qui migrent d'un groupe vers un autre ;

— la *link key* est partagée entre deux objets en utilisant la pré-configuration, le transport de clé, ou la méthode d'établissement de clé (voir 3.2.4.2 page 60). La *link key*

est dérivée d'une clé initiale (*master key*) qui est à son tour partagée entre deux objets avec une pré-configuration ou d'un transport non sécurisée « *Alternatively, if the application can tolerate a moment of vulnerability, the master key can be sent via an in-band unsecured key transport* »[131]. Par conséquent, le problème de mobilité des devices (pré-configuration) ou l'envoie en clair des clés initiales (transport non sécurisé) baisse considérablement les performances du protocole de sécurité du *ZigBee*.

En se basant sur ces travaux, nous proposons un nouveau protocole de sécurité basé sur des algorithmes robustes et rapides qui permet de résoudre ces problèmes et failles de sécurité, tout en étant adapté aux contraintes des technologies IoT. Pour assurer une meilleur performance à notre protocole, nous avons opté pour l'utilisation d'un nouveau concept qui a émergé dans le monde de la technologie d'information, ce dernier est intitulé la *blockchain*. Avant d'entamer la présentation de notre approche (partie III page 77), nous allons d'abord expliquer dans le chapitre 4 page ci-contre ce que c'est que la *blockchain*, sa puissance et les avantages qu'elle peut apporter à la sécurité et aux performances de l'IoT.

Chapitre 4

Les Blockchains

4.1 Introduction

Une *blockchain*, ou chaîne de blocs, est définie comme une base de données distribuée (*ledger*) qui conserve un enregistrement permanent et infalsifiable des données transactionnelles. Une *blockchain* est un système totalement décentralisé et basé sur un réseau pair à pair (peer-to-peer). Chaque objet du réseau conserve une copie du *ledger* afin d'éviter d'avoir un point unique de défaillance. Toutes les copies sont mises à jour et validées simultanément. Bien que l'objectif initial de la création de la *blockchain* était la résolution du problème de la dépense multiple en crypto-monnaie (monnaie virtuelle) [123]. Cette technologie peut être explorée dans de nombreux cas d'utilisation et utilisée comme un moyen sécurisé de gestion et protection de toute sorte de données (monétaire ou pas) [116] [49] [144].

Le *ledger* est composé d'un ensemble de blocs. Chaque bloc contient deux parties. La première partie représente le corps du bloc. Il contient les transactions, appelées également faits (*facts*), que la base de données doit enregistrer. Ces *facts* peuvent être des transactions monétaires, des données médicales, des informations industrielles, des logs systèmes, etc. La deuxième partie est l'entête (*header*) du bloc. Ce dernier contient des informations concernant le bloc tel que l'horodatage (*timestamp*), le hach des transactions, etc. Ainsi que le hachage du bloc précédent. De ce fait, l'ensemble des blocs existants forme une chaîne de blocs liés et ordonnés. Plus la chaîne est longue, plus il est difficile de la falsifier. En effet, si un utilisateur malicieux veut modifier ou échanger une transaction sur un bloc, (1) il doit modifier tous les blocs suivants, puisqu'ils sont liés par leurs hachs. (2) Ensuite, il doit changer la version de la chaîne de blocs que chaque objet participant stocke. La Figure 4.1 illustre un exemple simplifié d'une *blockchain*.

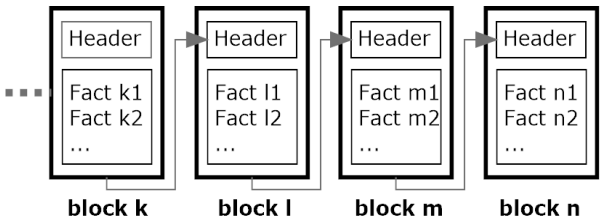


FIGURE 4.1 – Exemple simplifiée d'une *blockchain*

Il existe deux types d'objets participants : (1) des objets qui peuvent uniquement lire les *facts* (mode passif), et (2) des objets qui peuvent lire et écrire des *facts* (mode actif)

appelés mineurs. Afin de rajouter un nouveau bloc à la *blockchain*, il faut suivre les étapes suivantes :

— une transaction est regroupée avec d'autres transactions dans un bloc ;
— les mineurs vérifient que les transactions du bloc respectent les règles définies ;
— les mineurs exécutent un mécanisme de consensus pour valider le bloc ajouté ;
— une récompense est donnée aux mineur/mineurs [1] qui valident le bloc ;
— les transactions vérifiées sont stockées dans la *blockchain*.

Afin de prouver la validation honnête d'un bloc, il existe de nombreux mécanismes de validation. Les plus utilisés sont le mécanisme de *Proof of Work* (PoW) et le mécanisme de *Proof of Stake* (PoS).

4.1.1 Mécanismes de validation de blocs

4.1.1.1 Proof of work

Dans ce mécanisme, un mineur doit effectuer une quantité de travail, qui est souvent un puzzle ou un défi mathématique, difficile à calculer mais facile à vérifier. La difficulté du défi est adaptée, par la *blockchain*, en fonction du temps nécessaire à la validation d'un bloc. Une PoW est exigée pour la validation de chaque bloc. D'une part, elle a l'avantage de protéger l'intégrité des transactions et des blocs, car afin qu'un attaquant puisse modifier un bloc, il doit modifier tous les blocs qui le succèdent et fournir une nouvelle PoW pour chacun de ces bloc, ainsi que la mis à jour de tous les objets par la nouvelle version de la chaîne (falsifiée). Ce qui nécessite une énorme puissance de calcul et d'énergie. D'autre part, la PoW souffre de certaines lacunes qui peuvent avoir des mauvaises conséquences. Sans parler du fait que la PoW consomme une grande quantité d'énergie lors de la résolution du défie mathématique, ce mécanisme peut mener à une potentielle tragédie des ressources d'usage commun (*tragedy of commons*) [21]. En effet, au fil du temps, les récompenses diminueront, ce qui entraînera une diminution du nombre de mineurs, car les seuls frais qui seront gagnés viendront des transactions. Ces frais de transactions vont également diminuer due à la concurrence d'autres systèmes similaires. La diminution du nombre de mineurs rend l'écosystème *blockchain* vulnérable à l'attaque du 51%[100]. Cette dernière se produit lorsqu'un mineurs malicieux (ou un pool de mineurs malicieux) contrôlent 51%, ou plus, de la puissance de calcul du réseau. Ainsi, il peut créer des blocs de transactions frauduleux pour lui-même, ou pour une autre entité, tout en invalidant les transactions des autres utilisateurs dans le réseau. Enfin, dans certains mécanismes de consensus, tel que celui de la chaîne la plus longue (*longest chain*) appliqué dans *Bitcoin* (voir section 4.2 page ci-contre), de nombreux mineurs qui valident des blocs et réalisent la PoW ne sont pas récompensés, car ils n'ont pas assez de puissance pour construire la chaîne la plus longue, ce qui leur cause beaucoup de pertes.

La PoW représente la méthode de validation de bloc la plus adoptée par les systèmes *blockchain* (ex. *Bitcoin*, *Ethereum* (voir section 4.3 page 72) et *Litecoin* [88]).

4.1.1.2 Proof of Stake

Afin de résoudre les lacunes de la PoW, la PoS a été proposée. Dans ce mécanisme il n'y a pas de minage où on consomme beaucoup de ressources. Les mineurs sont appelés for-

1. Dépendamment de la *blockchain* utilisée, il peut y avoir un seul mineur ou plusieurs qui reçois/reçoivent une récompense.

geurs. Un forgeur peut valider des blocs en fonction de la quantité d'argent qu'il possède. Ce qui signifie que plus il possède de monnaies, plus il augmente sa chance de validation.

Si on compare la PoS à un jeu de pari, où chaque forgeur parie sur un bloc. On peut dire qu'une fois que les blocs honnêtes (ne contiennent aucune transaction frauduleuse) sont ajoutés à la chaîne, chaque forgeur touche une récompense relative à son pari. Et contrairement à la PoW où les mineurs malicieux sont pardonnés, dans la PoS un forgeur dont le bloc s'avère malhonnête est pénalisé et le montant du pari qu'il a mis est débité de son solde. Le point faible de la PoS est que les forgeurs qui possèdent beaucoup de monnaies sont ceux qui bénéficient le plus. Il existe plusieurs systèmes *blockchain* qui utilisent la PoS, et d'autres qui replacent la PoW par la PoS (ex. *Nxt* [143], *PPCoin* [90] et *BlackCoin*[160]).

Il existe d'autres mécanismes de validation de blocs tel que la *Delegated Proof-of-Stake* (DPoS) [167], la *Proof of Stake/Time* (PoST) [135], la *Proof of Existence* (PoE) [40], etc.

4.1.2 Catégories de la *blockchain*

La *blockchain* peut être "avec permission" (privée) ou "sans permission" (publique). La première catégorie impose des restrictions aux contributeurs du consensus. Seul ceux de confiances et choisis qui ont le droit de valider des transactions. Elle ne nécessite pas beaucoup de calcul pour atteindre un consensus, ainsi, elle est économique en terme de temps d'exécution et en énergie. Généralement les transactions sont privées et ne sont accessibles que par les objets autorisés. La deuxième catégorie (*blockchain* publique) utilise un nombre illimité d'objets anonymes. En se basant sur la cryptographie, chaque acteur peut communiquer d'une manière sécurisée. Chaque objet est représenté par une pair de clés (publique/privée), et a le droit de lire, d'écrire et de valider des transactions dans la *blockchain*. La blockchain est sûre si 51% des objets (ou plus) sont honnêtes et lorsque le consensus du réseau est atteint. Généralement, les *blockchains* sans permission consomment beaucoup d'énergie et de temps, car elles exigent un montant de calcul pour renforcer la sécurité du système (ex. en utilisant la PoW).

4.2 Bitcoin

***B**itcoin* est un système de payement numérique basé sur une *blockchain* publique. Il permet de créer une crypto-monnaie appelée ***b**itcoin*. Chaque bloc de la *blockchain Bitcoin* contient un hach de ses transactions appelé le *merkle root* [115] stocké au niveau de son entête. Ce dernier contient aussi le hach de l'entête du bloc précédent. Chaque participant au réseau *Bitcoin* peut être un mineur ou pas, et stocke une copie de la *blockchain* courante. Dans l'opération de minage, les transactions sont ordonnées et horodatées, puis stockées dans des blocs. Ensuite, un mécanisme de consensus est exécuté. En effet, afin de valider les transactions, *Bitcoin* utilise des règles qui lui sont propre. Plus précisément, les transactions ont des numéros de version qui indiquent aux objets *Bitcoin* l'ensemble de règles appropriées qui doit être utilisé pour assurer leur validation [3]. Afin de partager la même *blockchain* et éviter les conflits entre mineurs, *Bitcoin* utilise la règle de *la chaîne la plus longue* (*longest chain*). Un conflit se produit quand plusieurs mineurs (en compétition) génèrent des blocs en même temps, et que chacun de ces mineurs considère que son bloc est le bloc légitime qui doit être ajouté à la *blockchain*. Par exemple, si deux mineurs A et B essaient de rajouter le bloc numéro n, alors A génère le

bloc n_A et B génère n_B. Les deux blocs peuvent contenir différents ensembles de transaction, et chaque bloc contient l'adresse de son générateur afin que ce dernier puisse recevoir une récompense. Ensuite, comme les blocs ne sont pas ajoutés et partagés instantanément dans le réseau, chacun suppose que son propre bloc est légitime. Ainsi, il l'ajoute à sa chaîne et commence à construire le bloc suivant $(n+1)$. Si B est plus rapide que A et qu'il arrive à générer le bloc $n+1_B$ avant $n+1_A$, alors, en se basant sur le principe de *la chaîne la plus longue*, A doit prendre la chaîne de B en la considérant comme étant la chaîne valide, et abandonne la sienne (la plus courte), qui va être appelée chaîne orpheline (*orphaned chain*). *Bitcoin* utilise le mécanisme de validation de bloc PoW afin de rendre son système plus résistant aux attaques de modification. Par conséquent, comme décrit au dessus (voir section 4.1.1.1 page 70), pour chaque nouveau bloc, le mineur doit fournir sa PoW, qui est un défi difficile à produire (coûteux et long) mais facile aux autres à le vérifier. Plus précisément, la procédure de minage de *Bitcoin* et comme suit : (1) Chaque mineur crée un bloc contenant un entête (horodatage, *merkle root* des transactions du bloc, le hach du bloc précédent, etc) et un corps (transactions). Ensuite (2) le protocole de *Bitcoin* génère un (target) "t", qui représente une valeur $t \in]0, 2^{256}-1]$. (3) Chaque mineur doit calculer un hach d'un nombre choisi n ($n \in]0, 2^{256}-1]$) concaténé avec le hach de son bloc, d'une manière à ce que la valeur résultante doit être $\leq t$. En d'autres termes, le mineur change la valeur de n continuellement jusqu'à ce qu'il peut satisfaire l'équation $sha256(sha256(block)||n) \leq t$. Une fois que cette équation est satisfaite, le mineur ajoute cette valeur au bloc en tant que proof of work. Lorsqu'un objet construit un bloc et l'envoie sur le réseau, tous les destinataires vérifient les transactions du bloc et sa PoW. Si la majeure partie des objets du réseau s'accordent sur un bloc, alors celui-ci est validé et ajouté à la *blockchain*. Par conséquent tous les autres objets mettent à jour leurs copies de *blockchain*, et le créateur du bloc reçoit sa récompense. Le fonctionnement de la mise à jour de la *blockchain* se produit toutes les 10 minutes. Un mineur d'un bloc orphelin ne reçoit aucune récompense. Théoriquement, les blocs *Bitcoin* peuvent être falsifiés uniquement lorsqu'il y a 51% ou plus d'objets qui sont corrompus. Ce qui est actuellement presque impossible à réaliser. Par exemple, si *Google* exploite la partie de sa puissance de calcul, utilisée par son service *cloud comuting* [162], pour faire du minage, alors ça va représenter environ 0,0019% de l'ensemble des opérations de minage *Bitcoin* dans le monde[2].

4.3 Ethereum

Ethereum est une *blockchain* publique. Elle permet de fournir une crypto-monnaie appelée *Ether (ETH)*. Cette *blockchain* est utilisée pour effectuer des transactions financières ainsi que traiter d'autres types de données. En plus de la validation des blocs, les mineurs *Ethereum* traitent des programmes appelés contrats intélligents (*smart contracts*), c'est pourquoi *Ethereum* représente une plateforme pour les applications décentralisées. Les *smart contracts* sont exécutés par les objets participants en utilisant un système d'exploitation appelé *Ethereum Virtual Machine (EVM)* [4]. Tout comme dans le cas du *Bitcoin*, l'opération de minage consiste en la création et la validation des blocs. La taille d'un bloc *Ethereum* est plus petite que celle dans *Bitcoin*, et l'opération de validation finale d'un bloc et son rajout à la *blockchain* prend uniquement 14 secondes. Le système de rajout de bloc et de récompense est également différent. En effet *Ethereum* utilise un pro-

2. http ://www.zerohedge.com/news/2015-11-19/bitcoins-computing-network-more-powerful-525-googles-and-more-10000-banks

tocole appelé *Ethereum Greedy Heaviest Observed Subtree* (GHOST). Un mineur qui réussit à valider un bloc et l'ajouter à la *blockchain* principale reçoit 5 *ETH*. En plus, selon la complexité de l'exécution du contrat, il reçoit également un montant additionnel payé en *gas*[3] par l'expéditeur de chaque transaction. Lorsqu'un mineur construit un bloc, il l'envoie avec sa PoW via le réseau. Pendant les 14 secondes du consensus, chaque objet reçoit de nombreux blocs. De ce fait, il garde le premier bloc reçu dans sa chaîne principale et considère les autres comme étant *uncles* [63]. Ainsi, la chaîne qui contient le plus de blocs, appelée la chaine la plus lourde (*heaviest chain*) sera gardée en tant que chaîne principale et sera suivie par les différents objets participants. Enfin, *Ethereum* permet à chaque générateur d'*uncle* de recevoir une récompense égale à une partie de la valeur de la récompense du générateur du bloc de la chaine principale. Ce dernier, obtient à son tour des parts des récompenses désignées aux générateurs des *uncles*[37].

Pour la validation des blocs, *Ethereum* utilise un mécanisme basé sur la PoW appelé *Ethash*. Comme expliqué dans [38], les objets participants doivent calculer un cache pseudo-aléatoire de 16 M*o* basé sur un "*seed*" généré à partir des en-têtes des blocs. A partir de quelques éléments de ce cache, les objets génèrent un jeu de donnée de 1 G*o*. Le cache est stocké par des clients légers, tandis que le jeu de données est stocké par les mineurs. En hachant des morceaux du jeu de données choisis aléatoirement, les mineurs essaient de résoudre un défi mathématique. L'opération de vérification nécessite uniquement le cache afin de régénérer les morceaux spécifiques du jeu de données.

Actuellement, il existe une version beta d'*Ethereum* qui utilise un protocole appelé *Casper*. Ce dernier est basé sur la *PoS*.

Ethereum peut être utilisé en tant que *blockchain* privée, par conséquent, les objets participants sont choisis et les mécanismes de validation de bloc (ex. PoW) ne sont plus nécessaires.

4.4 Hyperledger Fabric

Hyperledger Fabric est une *blockchain* open-source privée, créée par la *Linux Foundation*, plus précisément par *IBM*. Contrairement à *Bitcoin* et *Ethereum*, *Hyperledger Fabric* ne fournit pas une monnaie virtuelle. Selon la nature des informations stockées, les transactions peuvent être publiques ou confidentielles. *Hyperledger* utilise le *Practical Byzantine Fault Tolerant* (PBFT) comme mécanisme de consensus. Comme expliqué dans [32], *PBFT* est un mécanisme utilisé dans les réseaux distribués, et qui tolère un certain taux de fautes afin de permettre la continuité des opérations du système. Tous les objets participants se connaissent et sont de confiances, et les objets validateurs sont choisis aléatoirement. *Hyperledger Fabric* permet également le développement des *smart contracts* appelés *chaincodes*.

3. *Gas* représente l'unité utilisée pour l'exécution d'une transaction ou d'un contrat dans *Ethereum*. 1 *gas* vaut 0.01 *microEther*

Troisième partie

Contribution

Dans ce qui suit, nous allons présenter notre travail de recherche et expliquer notre approche. Afin de résoudre les problèmes liés à la sécurité des technologies présentées dans le chapitre 3 page 35 et discuté dans la section 3.3 page 63 et créer un protocole performant et adapté aux systèmes IoT, notre travail est passé par plusieurs versions. Chaque version complète et améliore la précédente, jusqu'à l'obtention d'un protocole final et complet qui fournit les services de sécurité nécessaires pour la protection des technologies IoT. Les résultats obtenus prouvent l'efficacité de notre approche et ses bonnes performances. De ce fait, cette partie est divisée en chapitres, chacun traite une version et explique les services rajoutés et/ou améliorés. Afin d'avoir un cas d'usage bien défini, on a opté pour les technologies de capteurs sans fil (WSN), plus précisément nous avons utilisé la technologie de communication OCARI, expliquée dans la section 3.2.4.3 page 62. Nous avons progressivement conçu, implémenté et testé notre protocole de sécurité, ainsi nous l'avons déployé sur des vrais capteurs.

Dans la version 1 (chapitre 5 page 79), notre protocole assure l'authentification d'un device [4] lors de son association à un réseau de capteur géré par un CPAN. Il assure également l'intégrité des données échangés. La version 2 (chapitre 6 page 87) rajoute, en plus de l'authentification du device, l'authentification du CPAN, ainsi assurer une authentification mutuelle des objets. En outre elle rajoute aussi un mécanisme d'échange pour la clé de diffusion ($key_{broadcast}$). La version 3 (chapitre 7 page 91) améliore le service d'intégrité des données, et rajoute le service de confidentialité. La version 4 (chapitre 8 page 105) consiste en la création d'un nouveau mécanisme d'authentification décentralisé, basé sur les *blockchains* privées, qui améliore et complète la version 3, et qui permet la migration transparente et sécurisée des objets d'un réseau à un autre. La version 5 (chapitre 9 page 115) définie un mécanisme d'authentification décentralisé, basé sur les *blockchains* publique. Ce mécanisme est proposé pour assurer l'authentification de n'importe quel système IoT (ex. maison intelligente, ville intelligente, usine automatisée), mais on peut également l'exploiter pour rajouter beaucoup plus de performances de flexibilité et d'évolutivité à notre protocole de sécurité en améliorant la version 4.

4. Les mots "device", "Device" représentent un objet à capacité limité, contrairement au CPAN qui est un objet qui n'est pas contraint en terme de ressources.

Chapitre 5

Version 1 : Authentification des devices

5.1 Approche

Chaque communication entre deux objets nécessite une phase d'association. Généralement dans les réseaux WSN, afin qu'un objet A puisse être associé à un réseau géré par un objet B, A doit envoyer une *requête d'association* à B, et ce dernier lui renvoie une *réponse d'association*. Cependant n'importe quel objet malicieux C peut prétendre être A et usurper son identité. Pour palier à ce problème, il est primordial de mettre en œuvre un mécanisme d'authentification (voir chapitre 2 page 31). Après avoir étudié les différentes approches décrites dans la partie II page 25 et analysées dans le chapitre 3.3 page 63, nous avons opté pour une approche basée sur le mécanisme de "mot de passe à usage unique" (*One Time Password* (OTP)) qui est défini dans la RFC 2289 [66] et la RFC 4226 [120]. Par définition, un OTP est un mot de passe qui n'est valide qu'une seul fois. Par conséquent, il est très robuste contre les attaques de rejeu et de cryptanalyse (voir section 2 page 33). Il peut être utilisé en mode synchrone ou asynchrone.

Notre approche utilise le mode *asynchrone* qui est basé sur la méthode de challenge/réponse définie par la RFC 1994 [145]. Nous avons choisi ce mode car il n'exige aucune approbation préalable entre les objets communicants. Contrairement au mode *synchrone* qui nécessite un accord entre objets sur certains paramètres comme le "temps" (ex. Time-based One-Time Password algorithm (TOTP) [121]) ou un "compteur" (ex. HMAC-based One-time Password (HOTP) [120]). En plus, beaucoup de technologies sans fil ne supportent pas le temps absolu (absolute time), et due à l'instabilité de la plupart des réseaux sans fil, si un message est perdu, alors les valeurs des compteurs ne sont plus les mêmes, ce qui engendre un grand problème de synchronisation.

De ce fait, nous calculons notre OTP en combinant la méthode de challenge/réponse [145] avec HOTP [120] (défini à la base pour le mode *synchrone*) comme décrit dans l'Algorithme 9. Nous appelons la fonction qui génère l'OTP f_{otp}.

Algorithm 9: Calcul du OTP (f_{otp})

begin
- $hash$ = HMAC-SHA256 (key, $challenge$)
- $index = f_{index}(hash)$
- DBC1 = $4_{bytes}(index, hash)$
- DBC2 = DBC1 & 0x7F
- OTP = DBC2 % 8

D'abord on calcul un $hash$ -qui est une séquence de 32 $octets$- basé sur HMAC-

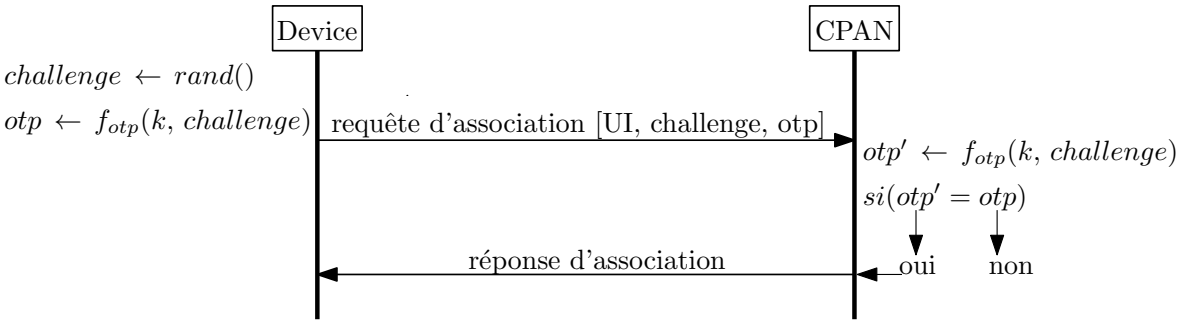


FIGURE 5.1 – Mécanisme d'authentification (cas 1)

SHA256 [126]. Ce hash contient comme données d'entré une clé secrète key (32 $octets$) et un $challenge$ (32 $octets$). Ce dernier représente un nombre aléatoire reçu de la part de l'authentificateur.

Ensuite, via une fonction f_{index}, on peut avoir une valeur qui s'appelle $index$. En effet, f_{index} prend les 4 bits aux poids faibles du dernier $octet$ du $hash$ obtenu, par exemple si le $hash$ est égal à :

`D7|F6|09|E3|51|3F|AA|5C|19|4D|98|2B|A2|EB|C3|C6|84|70|A4|E8|EB|58|B7|DD|56|3A|7E|53|83|AF|69|BC`.

Alors l'$index$ = 12, car le dernier $octet$ est égal à `0xBC`, et les 4 bits aux poids faibles représentent `0x0C` (12 à base 10).

Après, en prenant 4 $octets$ du $hash$ à partir de l'$index$ (ex. `0xA2EBC3C6`), on obtient le "Dynamic Binary Code" 1 (DBC1).

Afin d'éviter certains problèmes liés à l'interopérabilité entre les systèmes, nous traitons le DBC1 comme un entier non signé, big-endian de 31 $bits$, et l'$octet$ au poid fort (`0xA2`) est masqué avec un `0x7F`, ce qui donne DBC2 (ex. `0x22EBC3C6`).

Enfin, pour s'assurer que la taille de l'OTP soit égale à 4 $octets$, nous faisons "DBC2 modulo (%) 8" et on génère l'OTP final (ex. OTP = `0x22EBC3C6`).

5.1.1 Association

Avant chaque échange de données, un objet doit s'associer d'abord au réseau, ce qui nécessite son authentification. Dans ce qui suit, on va montrer les différentes conceptions par lesquelles on est passé avant d'arriver à la version sécurisée et finale par rapport à ce chapitre.

Afin d'éviter des traitements supplémentaires et d'assurer plus de transparence et de flexibilité aux couches hautes du modèle *Open Systems Interconnection* (OSI) [73], nous avons déployé notre mécanisme d'authentification au niveau de la sous-couche MAC.

Comme illustré par la Figure 5.1, pour éviter l'échange de plusieurs messages et économiser le temps et l'énergie nous avons exploité uniquement les deux messages d'association de base (*requête d'association* et *réponse d'association*). D'abord le Device génère un $challenge$ en utilisant la fonction random(). Puis il calcule un otp en utilisant la fonction f_{otp} (voir Algorithme 9 page précédente) avec comme données d'entré le $challenge$ et une clé secrète k partagée avec le CPAN. Ensuite il envoie une *requête d'association* contenant son identifiant unique (UI) associé par les paramètres générés. A la réception de la requête, le CPAN génère otp' -en se basant sur les mêmes paramètres- et le compare avec otp. Si les deux OTP correspondent, alors l'authentification du Device réussie

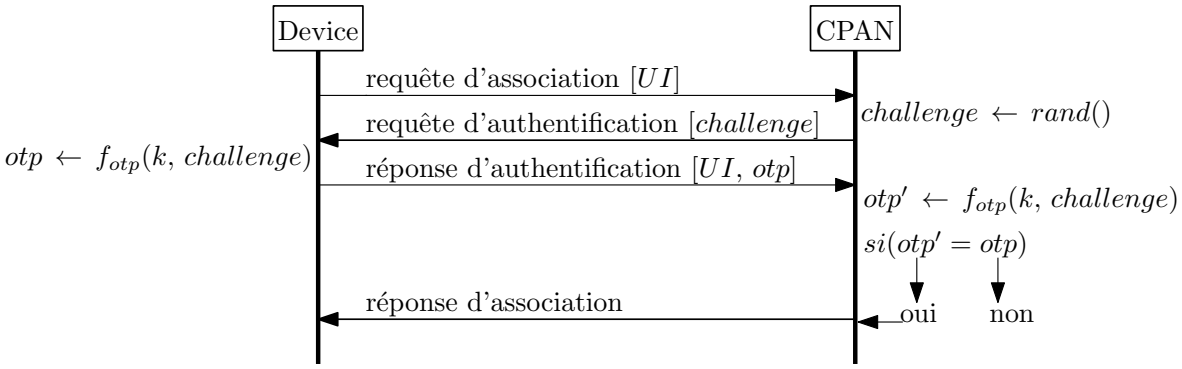


FIGURE 5.2 – Mécanisme d'authentification (cas 2)

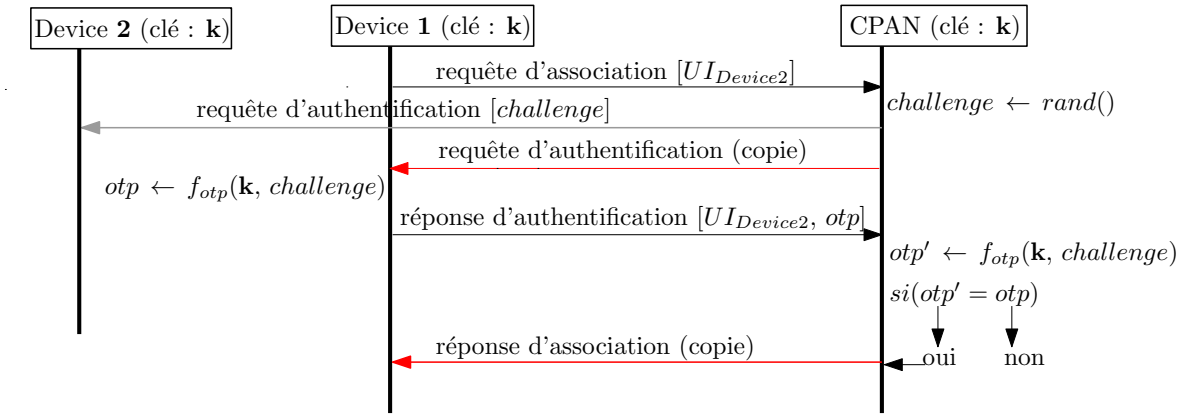


FIGURE 5.3 – Usurpation d'identité par un objet interne

et le CPAN l'associe au réseau, sinon l'authentification échoue et l'opération d'association s'arrête.

Cette solution est très optimale, cependant comme expliqué dans 3.3 page 64 (Protocole de sécurité du LoRaWAN), le fait que c'est l'initiateur de l'authentification qui génère le challenge représente une faille de sécurité.

Pour résoudre ça, on a rajouté deux autres messages *requête d'authentification* et *réponse d'authentification* (voir Figure 5.2).

Dans ce nouveau design, c'est l'authentificateur (CPAN) qui génère le challenge -après avoir reçu une *requête d'association*- et l'envoie au Device, afin que ce dernier puisse calculer otp et l'envoyer au CPAN via une *réponse d'authentification*. Enfin, le CPAN authentifie le Device et l'associe au réseau.

Notre mécanisme de sécurité est basé sur une clé secrète partagée. Cette clé doit être unique et connue uniquement par les deux entités communicantes, sinon si cette clé est partagée entre n objets (où $n \geq 3$), comme ce qui est le cas du protocole de sécurité du *Zigbee* expliqué dans 3.3 page 67 (Protocoles de sécurité du *ZigBee*), ceci représente une faille de sécurité et engendre des attaques internes sur l'authentification.

Comme l'explique Figure 5.3, un attaquant interne (Device 1) a la possibilité d'usurper l'identité de Device 2 car il possède la même clé secrète **k** que Device 2 et le CPAN. Ainsi, Device 1 peut écouter, modifier ou falsifier les messages échangés entre le Device 2 et le CPAN.

Ce problème peut être résolu en installant des pairs de clés uniques entre les devices et le CPAN. Cependant cette solution va poser un problème de flexibilité comme ce qui est le cas de la plupart des approches étudiées dans la section 3.3 page 63.

Donc pour remédier à ça, nous avons conçu un nouveau mécanisme de **"Personnali-**

sation" de clé. La Figure 5.4 résume les étapes du mécanisme.

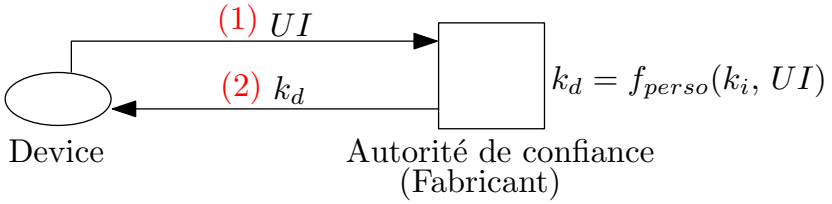


FIGURE 5.4 – Mécanisme de personnalisation de clés

L'autorité de confiance, qui est généralement le fournisseur, doit installer -en mode hors-bande (ex. flashage physique)- une clé initiale k_i dans le CPAN, et en génère des clés dérivées k_d(s) attribuées à chaque dispositif légitime appartenant au même cluster. Ces clés sont personnalisées et uniques car elle sont basées sur l'identifiant unique de chaque device (UI [1]). La dérivation de cette clé est basée sur la fonction $f_{perso}(k_i, \text{UI})$ (voir Équation 5.1).

$$\begin{cases} k_d = f_{perso}(k_i, \text{UI}) \\ f_{perso}(k_i, \text{UI}) = hash_function(k_i, \text{UI}) \end{cases} \tag{5.1}$$

Où $hash_function$ est une fonction irréversible qui génère des clés robustes et qui protège k_i contre les attaques par déduction. Une fois que k_d est créée et configurée dans le device, celui-ci peut être associé au réseau.

En plus de la résolution du problème d'usurpation d'identité par les objets internes, cette solution est optimale et très flexible, car le CPAN n'a pas besoin d'avoir les k_d(s) des objets participants pour les authentifier, mais plutôt, en utilisant k_i il peut obtenir n'importe quelle k_d. Lorsqu'on rajoute un nouveau device x au réseau, il n'est pas nécessaire de mettre à jour le CPAN par la nouvelle clé k_{dx}, car en se basant sur UI_x et k_i, le CPAN peut générer k_{dx}.

Le design du mécanisme d'authentification corrigé est illustré par la Figure 5.5.

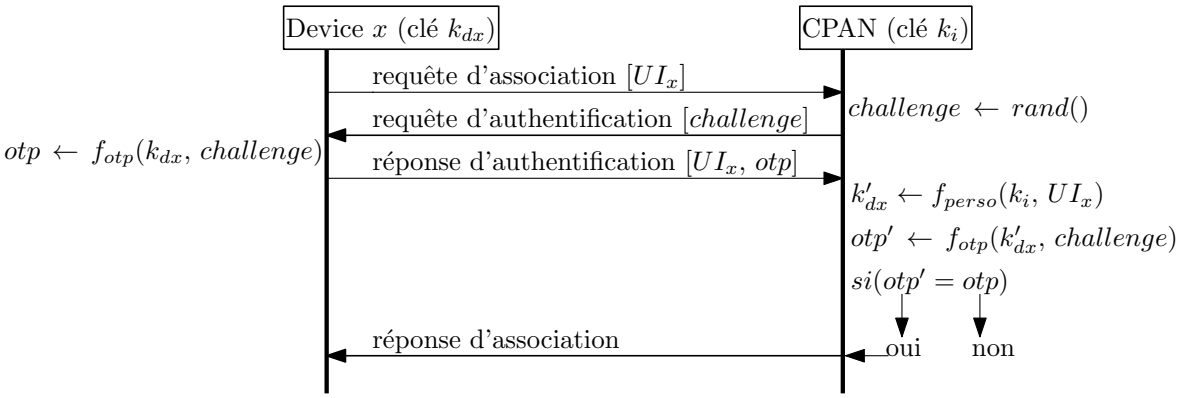


FIGURE 5.5 – Mécanisme d'authentification (design corrigé)

Lorsqu'un Device x ayant une clé dérivée k_{dx} veut être associé à un réseau, il doit d'abord générer une *requête d'association* contenant son identifiant UI_x, le CPAN lui génère et envoie un $challenge$ via une *requête d'authentification*. A la réception de la requête, Device x calcule otp en utilisant le $challenge$ reçu et k_{dx} comme données d'entrée, puis il envoie le résultat -associé à UI_x- au CPAN via une *réponse d'authentification*.

1. Par exemple dans OCARI, UI est une adresse IEEE d'une taille de 8 $octets$.

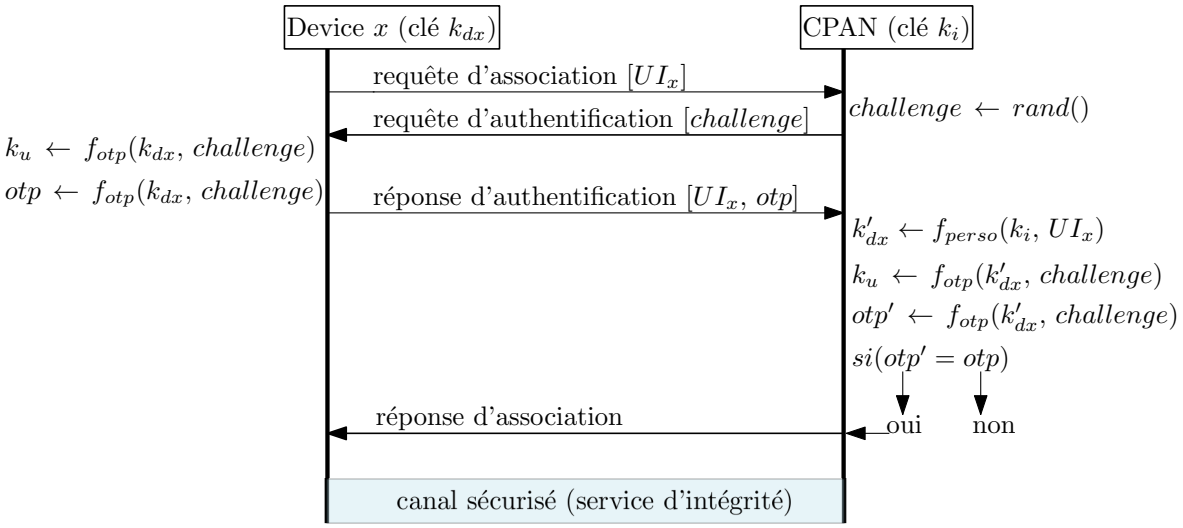


FIGURE 5.6 – Protocole de sécurité (version 1)

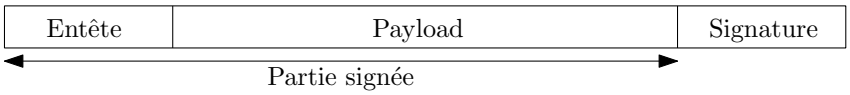


FIGURE 5.7 – Format d'une trame (version 1)

Enfin, le CPAN génère une clé dérivée k'_{dx} correspondante au Device x, et calcule otp' en se basant sur k'_{dx} et le $challenge$. Si la vérification réussit, alors le device sera associé, sinon il sera rejeté.

5.1.2 Canal de communication

En exploitant les 16 premiers $octets$ du hash généré lors du calcul d'otp on génère une clé k_u qui va être utilisée pour assurer l'intégrité des données échangés -en mode unicast- au niveau d'un canal de communication (session). k_u est une clé éphémère et n'est valide que pendant une seul session, ceci protège notre protocole contre les attaques de cryptanalyse comme celle appliquée sur le WEP (voir section 3.3 page 63). La Figure 5.6 montre le protocole final de ce chapitre.

Toute les données échangées son munies d'une signature permettant d'assurer leurs intégrités. Le format des trames à envoyer est présenté par la Figure 5.7. La signature représente les n premiers $octets$ du HMAC-SHA256 de la trame à envoyer (voir Equation 5.2).

$$signature = first(n, \text{HMAC} - \text{SHA256}(k_u, payload)) \tag{5.2}$$

La fonction *first()* permet d'avoir les n premiers $octets$ d'une donnée. Due à la limitation de taille des trames dans les réseaux IoT, n ne doit pas être grand (ex. 4 ou 16 $octets$). De cette façon on assure que la signature n'influence pas négativement sur les performances de l'échange de données.

5.2 Évaluation et résultats

Dans ce chapitre nous nous intéressons uniquement au temps écoulé lors de la phase d'association. Une évaluation plus complète de notre protocole de sécurité est présen-

tée dans le chapitre 7 page 91. Afin d'évaluer les performances de notre protocole de sécurité, nous avons réalisé une implémentation en langage C sur des capteurs (micro-contrôlleurs) *Dresden Elektronik deRFsam3 23M10-R3*, ayant 48 ko de RAM, 256 ko de ROM et un processeur *Cortex-M3*. Pour notre expérience, nous avons réalisé une petite architecture composée de 3 objets qui représentent un CPAN et deux devices.

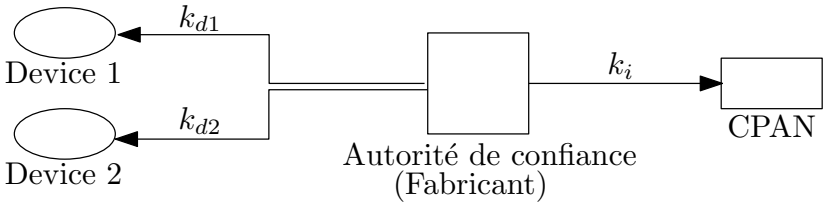


FIGURE 5.8 – Architecture de test

Comme montré dans Figure 5.8, d'abord via une autorité de confiance (ex. le fabricant), on installe une clé initiale k_i au niveau du CPAN et on personnalise les devices (Device 1 et Device 2) avec les clés dérivées k_{d1} et k_{d2} (comme expliqué dans la section 5.1.1 page 80). Lors de la phase d'association, Device 1 demande son association au réseau en communicant directement avec le CPAN, tandis que Device 2 s'associe au CPAN en passant sa requête par Device 1 (à 2 sauts) qui est déjà associé et qui joue le rôle de routeur.

En utilisant *Zolertia z1* sniffer (hardware) et *Wireshark* (software) [58], nous avons mesuré 10 fois le temps d'association de devices à 1 et à 2 sauts par rapport au CPAN, avec et sans authentification.

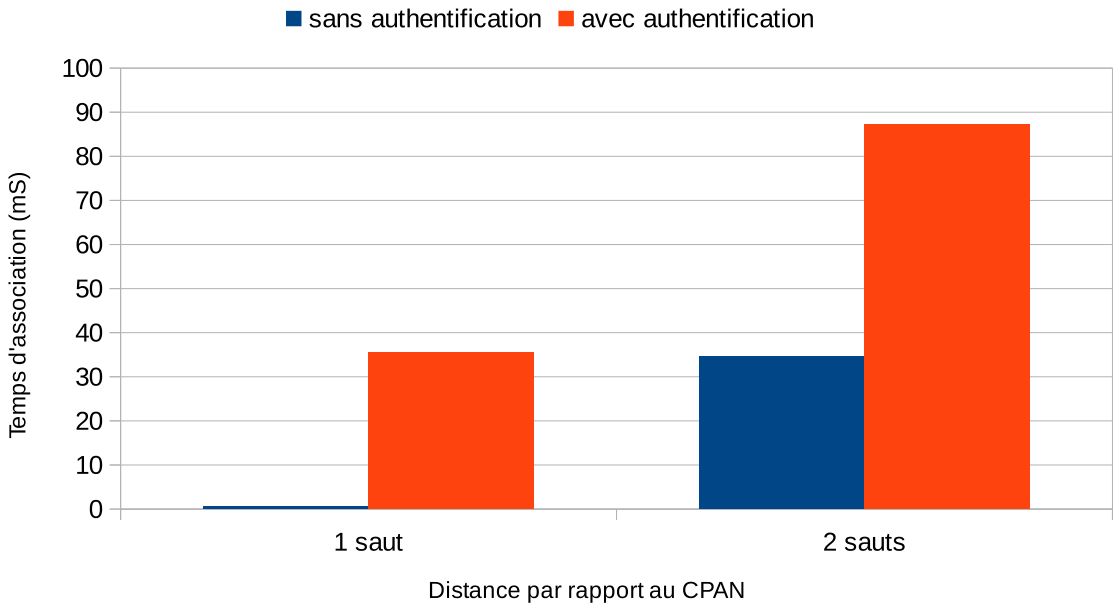


FIGURE 5.9 – Temps d'exécution de l'opération d'association

Comme le montre la Figure 5.9, le temps d'association (la moyenne) est augmenté de 0.524 mS à 35.535 mS pour le device à 1 saut du CPAN. Quant au temps d'association (la moyenne) requis par le device à 2 sauts, il est augmenté de 34.508 mS à 87.194 mS. Cette augmentation est due à l'échange de 2 messages supplémentaires du protocole d'authentification. En effet, le nombre de messages nécessaires au device à 2 sauts pour son asso-

ciation sans authentification (4 messages) est prèsque égal à celui de l'association à 1 saut avec authentification (4 messages).

Par conséquent, l'augmentation du temps causée par l'authentification est due principalement aux messages supplémentaires et de manière insignifiante au temps de traitement.

5.3 Conclusion

Dans ce chapitre nous avons conçu un protocole qui contient un mécanisme d'authentification des devices et un service de protection d'intégrité de données. Le mécanisme d'authentification est léger rapide et robuste. Il permet de protéger les systèmes contre les attaques de rejeu et de cryptanalyse. Le principe de personnalisation assure une gestion de clés d'une manière optimale, protège le réseau contre les attaques internes, et fournit une flexibilité et une transparence par rapport aux rajout des nouveaux objets. Enfin, le service d'intégrité permet de protéger les données contre les attaques de falsification.

Cependant ce protocole est incomplet. Premièrement, il est important que le device puisse authentifier le CPAN avant d'échanger des informations, car il se peut que ce dernier soit malicieux et en créant un faux réseaux il peut corrompre le device. Par conséquent, une authentification mutuelle est nécessaire. Et deuxièmement, le mécanisme d'échange de clé ne permet pas d'échanger la clé de diffusion k_b utilisée pour sécuriser les échanges de données en mode broadcast.

Dans le prochain chapitre, nous allons rajouter des améliorations sur notre protocole afin de le rendre plus complet et plus performant.

Chapitre 6

Version 2 : Authentification mutuelle des objets

6.1 Approche

Dans cette version, nous allons améliorer notre protocole afin de le rendre plus efficace et plus complet. La Figure 6.1 illustre le nouveau design du protocole.

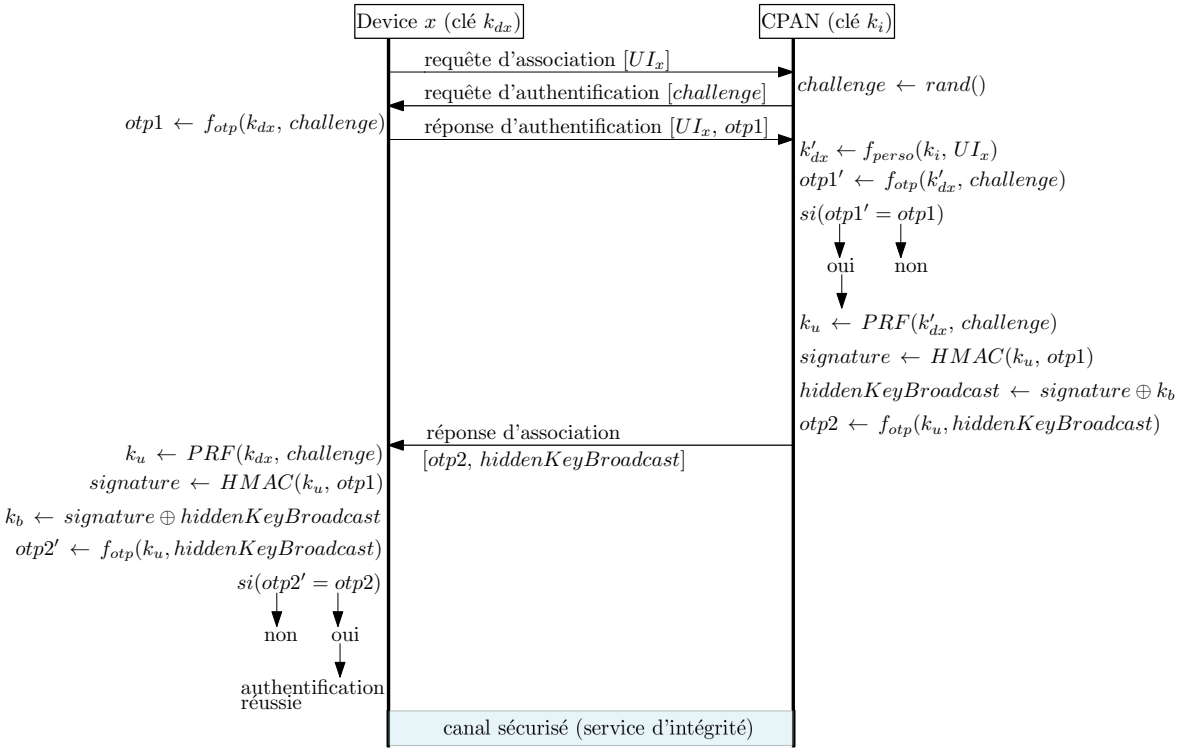


FIGURE 6.1 – Protocole de sécurité (version 2)

D'abord, le device fait une *requête d'association*, reçoit une *requête d'authentification*, génère et envoie $otp1$, et enfin, authentifié par le CPAN s'il est légitime. Ensuite, contrairement à l'ancienne version, la clé k_u est généré en utilisant la *Pseudo Random Function* (PRF) définie dans la RFC 5246 [44] qui permet d'avoir des clés très robustes. La génération des clés ne s'établit qu'après avoir terminé l'opération d'authentification avec succès afin d'éviter de faire des calculs inutilement. Dans cette version nous avons également rajouté un mécanisme d'échange sécurisé de clé de diffusion k_b -appelé *hidden key broad-*

cast- et un calcul d'un deuxième OTP ($otp2$) pour l'authentification du CPAN. Le mécanisme *hidden key broadcast* est réalisé en 2 phases. (1) On génère une valeur nommée $signature$ en calculant un HMAC($k_u, otp1$), ensuite, (2) on $\oplus$ le résultat avec k_b.

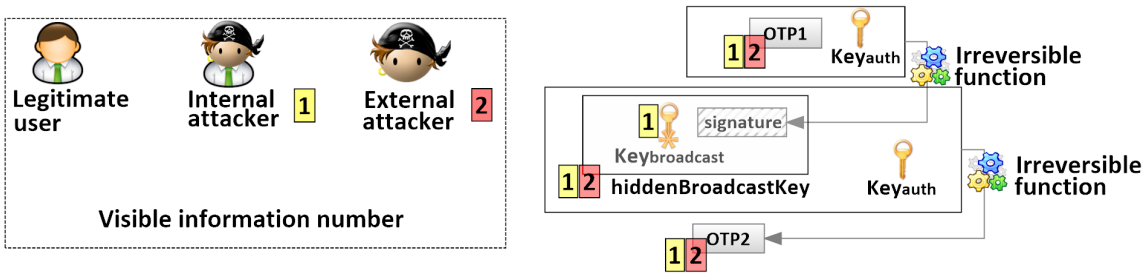


FIGURE 6.2 – Mécanisme d'échange de clé de diffusion (*hidden key broadcast*)

La Figure 6.2 montre les différents types d'utilisateurs et leurs droits d'accès aux informations échangées.

Si un attaquant externe intercepte toutes les informations ($challenge$, $otp1$, $otp2$ (expliqué ci-dessous) et $hiddenKeyBroadcast$), il ne peut obtenir aucune information secrète (clés), car il ne possède pas le couple (k_{dx}, k_b) ni le couple (k_u, k_b).

Pour un attaquant interne qui a k_b en plus de toutes les informations échangées, il ne peut pas non plus obtenir les clés d'autres devices. Autrement dit, lorsqu'un attaquant interne tente d'obtenir k_u d'un autre device, il calcule le $\oplus$ entre k_b et $hiddenKeyBroadcast$ afin d'obtenir la $signature$, et comme cette dernière est générée à partir d'une fonction irréversible (HMAC), même en utilisant $otp1$, l'attaquant ne peut pas obtenir k_u.

$otp2$ est calculé par le CPAN pour faire d'une pierre deux coups. Premièrement, pour protéger l'intégrité de $hiddenKeyBroadcast$. Deuxièmement, pour assurer l'authentification du CPAN. La génération d'$otp2$ nécessite un secret (clé) et un challenge unique. Par conséquent le CPAN utilise k_u en tant que secret, et exploite $hiddenKeyBroadcast$ comme challenge. Ce dernier est unique car il est basé sur une signature unique, qui est basé sur un OTP unique ($otp1$). Ensuite, $otp2$ et $hiddenKeyBroadcast$ sont envoyés au device via une *réponse d'association*. A la réception du message, le device calcule k_u puis $signature$. Il $\oplus$ $signature$ avec $hiddenKeyBroadcast$ afin de récupérer k_b qui doit être examinée (vérifier son intégrité). Pour y parvenir, le device calcule $otp2'$ -basé sur la valeur de $hiddenKeyBroadcast$ reçu et k_u- puis le compare avec $otp2$. Si les deux OTP correspondent, alors (1) ça prouve que $hiddenKeyBroadcast$ n'est pas falsifié, et donc la k_b est correcte, et (2) le CPAN est authentifié (légitime). Sinon si $otp2$ ou $hiddenKeyBroadcast$ ou les deux ne sont pas correctes ou modifiés pendant leur transmission, alors les 2 OTP ne correspondent pas, par conséquent k_b n'est pas acceptée, le CPAN est rejeté, et l'opération d'association échoue. Le fait d'avoir un $otp2$ correcte prouve la validité de l'identité du CPAN. Car ce dernier est calculé en utilisant k_u qui est dérivée de k_{dx}.

Une fois que l'opération d'association se termine, tous les messages échangés entre le device et le CPAN doivent être signés par HMAC-SHA256 (comme expliqué dans la section 5.1.2 page 83) en utilisant k_u pour le mode unicast et k_b pour le mode broadcast.

6.2 Évaluation et résultats

Afin d'évaluer notre nouvelle version de protocole de sécurité, nous avons effectué des tests sur la même architecture et le même hardware utilisé dans le chapitre 5.2 page 83.

Nous avons mesuré le temps écoulé pendant la phase d'association en utilisant *Zolertia z1* sniffer (hardware) et *Wireshark* (software) [58].

Due à la particularité de la sous-couche MAC d'OCARI, nous ne pouvons pas comparer notre temps d'association sécurisée avec d'autres technologies basées sur le même standard (le standard 802.15.4), car elles sont très différentes. Néanmoins, à titre indicatif, nous avons présenté les résultats du temps d'association obtenus lors d'une évaluation du protocole *ZigBee* (expliqué dans la section 3.2.4.2) déployé sur un hardware similaire.

Remarque : dans ce chapitre nous nous intéressons uniquement au temps écoulé lors de la phase d'association. Une évaluation plus complète de notre protocole de sécurité est présentée dans le chapitre 7 page 91.

Nous avons mesuré les délais d'association de notre approche (appliquée sur OCARI) "avec authentification (auth)" et "sans authentification (none)" à "1 saut" et à "2 sauts". Ensuite en se basant sur des résultats sur le temps d'association à "1 saut" du *ZigBee* "avec authentification" obtenus à partir de [17], nous avons réalisé le diagramme à barres illustré par la Figure 6.3.

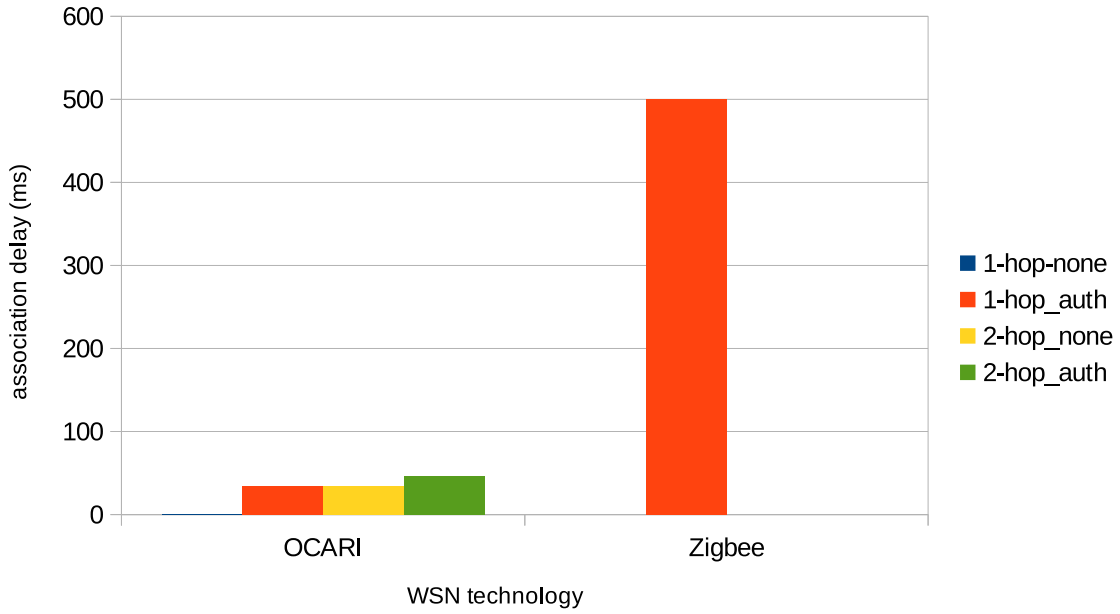


FIGURE 6.3 – Temps d'association de notre approche et celui du protocole *ZigBee*

Comme le montre la Figure 6.3, pour un device à "1 saut" du CPAN, le délai moyen de 10 associations passe de 0.524 mS pour le mode "sans authentification" à 37.504 mS pour le mode "avec authentification". Le temps d'association d'un device à "2 sauts" est augmenté de 34.508 mS "sans authentification" à 45.876 mS "avec authentification".

Le nombre de messages utilisé par le device à 2 sauts pour son association "sans authentification" est presque égal à celui à "1 saut" "avec authentification" (4 messages). La différence entre les deux est seulement de 2.996 mS (37.504 mS - 34.508 mS). Par conséquent, cette augmentation est due principalement aux 2 messages additionnels nécessaires pour l'authentification, et dans une moindre mesure, au temps de traitement supplémentaire.

Enfin, par rapport au protocole *ZigBee*, on peut constater que notre mécanisme d'authentification est optimal.

6.3 Conclusion

Ce protocole améliore les performances de celui expliqué dans le chapitre précédent (voir 5 page 79). Elle assure un service d'authentification mutuelle robuste, flexible et adapté aux objets à faibles ressources. En plus, elle fournit un mécanisme transparent et efficace pour la génération et l'échange des clés. Enfin, au niveau du canal sécurisé, elle permet d'assurer l'intégrité de tous les messages échangés en mode unicast et broadcast.

Bien que cette nouvelle version est plus complète que son ancienne, la confidentialité des données, échangées au niveau du canal sécurisé, n'est pas assurée, et n'importe quel utilisateur malicieux peut écouter le trafic et récupérer des informations qui peuvent être sensibles et confidentielles. Pour résoudre ce problème, dans le prochain chapitre, nous allons intégrer l'utilisation de l'*Advanced Encryption Standard* (voir chapitre 7 page ci-contre) avec les modes d'opérations *Galois Counter Mode (GCM)* et *Counter with CBC-MAC (CCM)*, afin d'assurer à la fois le service de confidentialité et d'intégrité des données.

Chapitre 7

Version 3 : Sécurisation des systèmes IoT

7.1 Approche

Comme expliqué ci-dessus, généralement les systèmes IoT, et en particulier les réseaux WSN sont constitués d'objets contraints (ex. capteurs ou actionneurs) gérés par un objet non contraint (ex. CPAN). Lorsqu'un nouveau device souhaite rejoindre un réseau WSN, une authentification mutuelle entre ce dernier et le gérant du réseau doit d'abord être effectuée. Ensuite, un canal sécurisé symétrique se créé entre les entités communicantes pour protéger les données échangées. Dans cette nouvelle version, afin d'assurer les services d'intégrité et de confidentialité de données, nous avons opté pour l'*Advanced Encryption Standard* (AES). Ce dernier est un algorithme de chiffrement symétrique qui était annoncé par le NIST en 2001 sous le nom de *FIPS PUB 197* [150], et est devenu l'algorithme de chiffrement par bloc le plus utilisé au monde. AES traite les blocs de données à l'aide des clés de chiffrement d'une longueur de 128, 192 ou 256 $bits$.

En tant que mode d'opération [1], nous avons choisi le *Galois Counter Mode* (GCM) et le *Counter with CBC-MAC* (CCM), qui assurent à la fois l'intégrité et la confidentialité des données. L'activation du mode GCM ou CCM est réalisée grâce à un fichier de configuration.

Avec cette amélioration, nous obtenons la version 3 de notre protocole, qui est représenté par la Figure 7.1 page suivante.

Donc, comme expliqué précédemment, d'abord les deux objets s'authentifient mutuellement, génèrent et échangent des clés de session, et enfin créent un canal sécurisé assurant un chiffrement authentifié des données échangées en utilisant AES-GCM/CCM.

Dans ce qui suit nous allons expliquer le fonctionnement de AES avec ses différent mode d'opération.

7.1.1 AES

Le processus de chiffrement AES consiste à chiffrer indépendamment des blocs de données d'une taille de 128 $bits$. Chaque bloc subit tout d'abord un $\oplus$ avec la clé symétrique fournie (k). Pour une clé de 128 $bits$, le résultat obtenu sera chiffré en 10 étapes (tours). D'après la spécification d'AES on aurait eu 12 tours pour une clé de 192 $bits$ et 14 pour une clé de 256 $bits$ [1]. Lors de chaque tour, le bloc passera successivement par une fonction de substitution d'$octets$, une fonction de décalage et une fonction de permutation avant de subir enfin un $\oplus$ avec une clé dérivée (k_x) de la clé symétrique principale

1. En cryptographie, un mode d'opération est la façon de traiter les blocs de données (chiffrés ou en clair) en utilisant un algorithme de chiffrement par bloc.

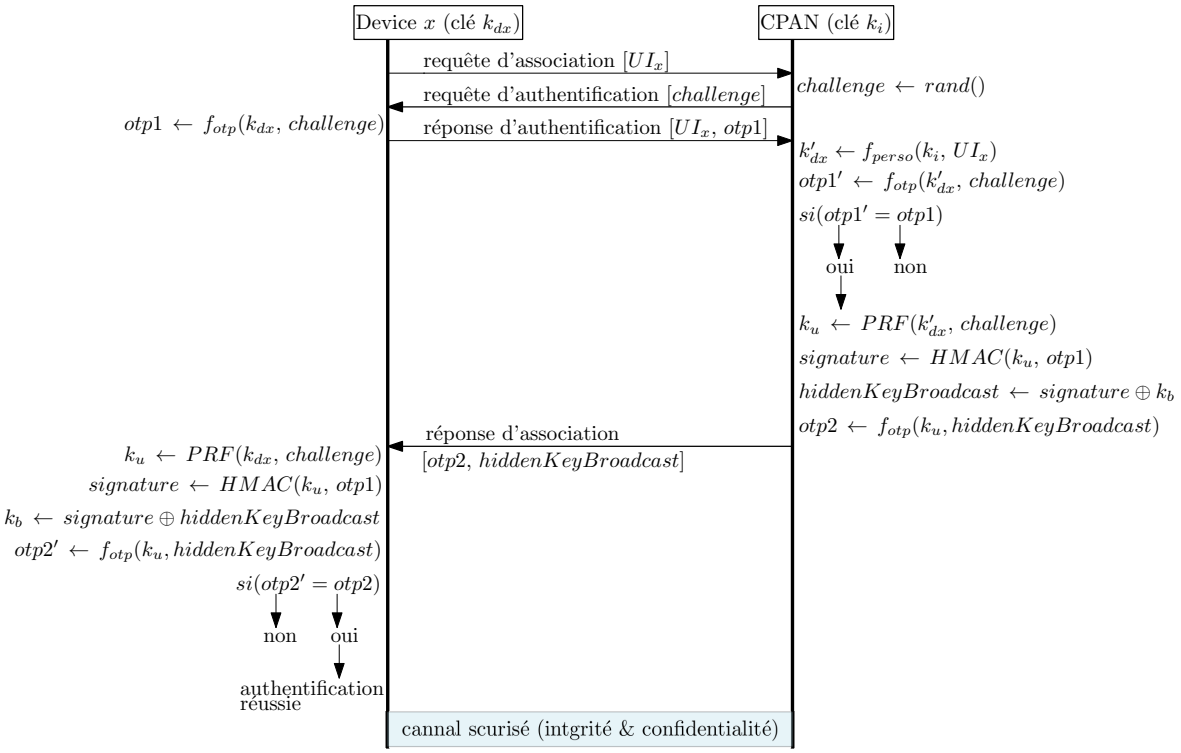


FIGURE 7.1 – Protocole de sécurité (version 3)

(voir Figure 7.2). Le résultat sera alors prêt pour le tour suivant (ou éventuellement la sortie). Le bloc obtenu en sortie (128 $bits$) subira un $\oplus$ avec le bloc de données à chiffrer (128 $bits$). Le résultat sera ainsi le bloc de données chiffré. Le processus continue tant qu'il y a des données à chiffrer.

7.1.2 GCM

GCM est un mode d'opération de chiffrement par bloc qui utilise le hachage universel sur un champ de Galois binaire.

7.1.2.1 Chiffrement authentifié

Une entité voulant échanger une données, elle doit la chiffrer puis l'authentifier (signer). Cette opération utilise la clé généré k_u pour le mode unicast, ou la clé k_b pour le mode broadcast. Elle nécessite comme donnée d'entrée (input) le plaintext $P_1, \cdots, P_n$, une donnée additionnelle authentifiée A (A peut être n'importe quelle donnée, ajoutée pour renforcer l'algorithme de chiffrement) et un vecteur d'initialisation IV. La génération du IV est effectuée en se basant sur k_u et un compteur. Ce dernier est utilisé afin d'éviter les attaques par cryptanalyses. Comme résultat (output), on obtient le ciphertext $C_1, \cdots, C_n$ et un tag d'intégrité T.

L'opération du chiffrement authentifié est décrite par l'Algorithme 10.
Où : E est l'opération de chiffrement, K est la clé secrète (K_u ou K_b), 0^{128} est un bloc de 128 $bits$ composé de 0, H est obtenu en chiffrant 0^{128} par la clé K, Y est un compteur commençant à partir de Y_0. Ce dernier représente la concaténation (||) de IV avec 31 "0" et un "1" ($bits$), MSB_t est le bit de poids fort (en anglais most significant bit) , len(A) est la longueur de A et GHASH() est la fonction de hachage du mode d'opération GCM.

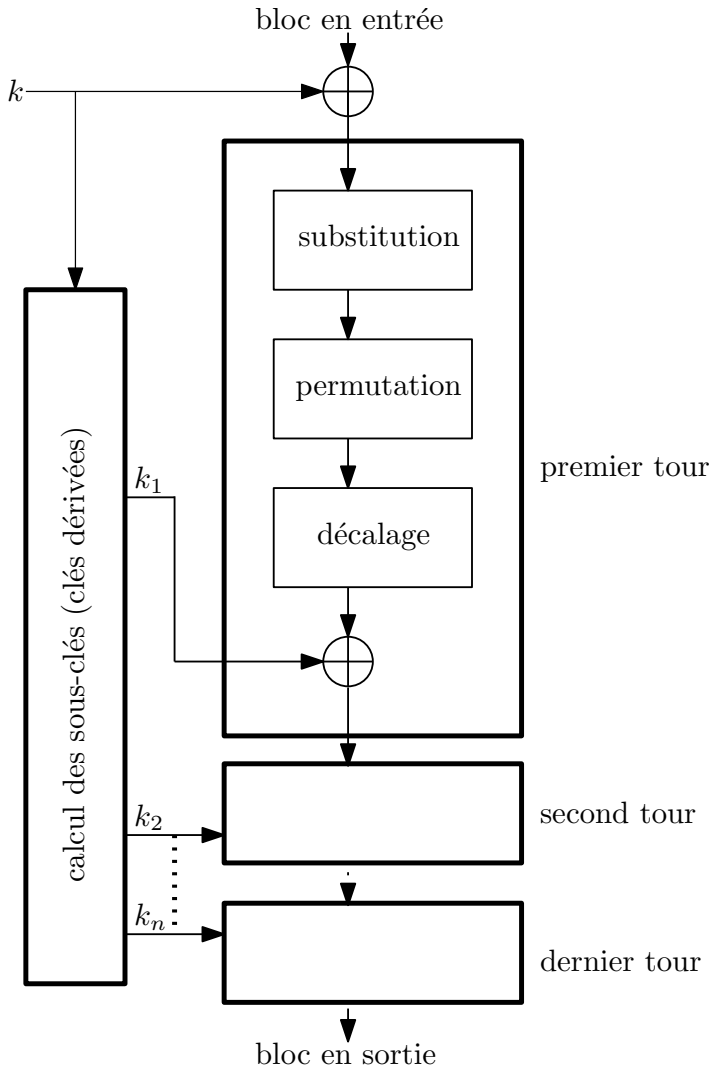


FIGURE 7.2 – Le mécanisme de fonctionnement d'AES

Algorithm 10: chiffrement authentifié du AES-GCM

begin
- $H \leftarrow E(K, 0^{128})$
- $Y_0 \leftarrow IV||0^{31}1$ // on utilise $len(IV)$ de $96\,bits$
- **for** $(i = 1, i \leq n, i++)$ **do**
  - $Y_i = Y_{i-1} + 1$
- **for** $(i = 1, i \leq n, i++)$ **do**
  - $C_i = P_i \oplus E(K, Y_i)$
- $T = MSB_t(GHASH(H, A, C, len(A), len(C)) \oplus E(K, Y_0))$

A la fin de l'opération du chiffrement authentifié, nous concaténons le *ciphertext* résultant avec le tag T. Ce dernier va être utilisé afin de vérifier l'intégrité des données une fois que le paquet soit reçu.

7.1.2.2 Déchiffrement authentifié

Pour le déchiffrement nous avons 4 inputs qui sont : Le ciphertext $C_1, \cdots, C_n$, le tag reçu T, le IV et la données additionnelle authentifié A. En tant que output, nous obtenons le plaintext $P_1, \cdots, P_n$ et un tag T' afin de pouvoir vérifier l'intégrité des données.

L'opération du déchiffrement authentifié est décrite par l'Algorithme 11.

Algorithm 11: déchiffrement authentifié du AES-GCM

begin
- $H \leftarrow E(K, 0^{128})$ $Y_0 \leftarrow IV||0^{31}1$ `// on utilise` $len(IV)$ `de` $96 bits$
- $T' = MSB_t(GHASH(H, A, C, len(A), len(C)) \oplus E(K, Y_0))$
- **for** $(i = 1, i \leq n, i++)$ **do**
  - $Y_i = Y_{i-1} + 1$
- **for** $(i = 1, i \leq n, i++)$ **do**
  - $P_i = C_i \oplus E(K, Y_i)$
- **if** $(T = T')$ **then**
  - `// opération de déchiffrement réussie`
- **else**
  - `// opération de déchiffrement échouée`

L'opération de déchiffrement utilise la même clé utilisée pour l'opération de chiffrement. A la fin du processus de déchiffrement, on compare le T reçu avec T' (calculé par l'entité réceptrice) Si les deux correspondent, alors l'opération de déchiffrement réussie, sinon elle échoue. Pour plus de détails sur le mode GCM, voir [52].

7.1.3 CCM/CCM*

Le Counter mode with CBC-MAC (CCM) [51] est un mode d'opération qui combine le mode counter (CTR) [75] et l'algorithme d'intégrité CBC-MAC [20]. Comme dans le cas du GCM, le CCM assure à la fois le service d'intégrité et le service de confidentialité. Il permet d'authentifier puis chiffrer les données. Le CCM utilise des blocs d'une taille de 128 $bits$. AES-CCM utilise également une clé symétrique unique pour le chiffrement/déchiffrement authentifié.

Remarque : cette section concerne également le mode CCM*, car, ce dernier fonctionne de la même manière que le CCM, mais avec des restrictions sur les tailles de quelques paramètres afin d'éviter certaines failles de sécurité.

7.1.3.1 Chiffrement authentifié

L'opération de chiffrement est décrite par l'Algorithme 12.

Algorithm 12: Chiffrement authentifié du AES-CCM

begin
- $X_1 \leftarrow E(K, P_0)$
- **for** $(i = 1, i \leq n, i++)$ **do**
  - $X_{i+1} \leftarrow E(K, X_i \oplus P_i)$
- $T \leftarrow first_M_bytes(X_{n+1})$
- **for** $(i = 0, i \leq n, i++)$ **do**
  - $S_i = E(K, A_i)$
- **for** $(i = 1, i \leq n, i++)$ **do**
  - $C_i \leftarrow S_i \oplus P_i$
  - $C \leftarrow C||C_i$
- $U \leftarrow first_M_Bytes(S_0) \oplus T$

Où : T est le tag d'intégrité (connu également par Tag), M et L sont, respectivement, le nombre d'$octets$ dans le champ T et le champ de longueur, K est la clé symétrique, P (P_i..P_n) et C (C_i..C_n) représentent, respectivement, le plaintext et le ciphertext, || est l'opérateur de concaténation, A est la donnée additionnelle authentifiée (une donnée utilisée pour renforcer l'algorithme de chiffrement/déchiffrement), N est un nombre aléatoire (nonce), C$ounter$/C$ounter_x$ est le compteur, A_x est la concaténation d'un drapeau (flag) de 1 $octet$ avec N et C$ounter_x$. U une valeur permettant d'avoir T.

AES-CCM nécessite 5 inputs : K, N, P, A et C$ounter$.

L'algorithme CBC-MAC assure l'intégrité des données par le calcul du MAC (T) comme suit :

(1) Le plaintext (P) est dévisé en n blocs désignés $P_1, \cdots, P_n$. (2) Un bloc P_0 est créé par la concaténation de A, N et L. (3) Le bloc P_0 est utilisé en tant que vecteur d'initialisation et est chiffré avec l'algorithme AES (E()) en utilisant la clé K. Le résultat est mis dans X_1. (4) Après avoir calculé X_{n+1}, en prenant les M premiers $octets$ (généralement égal à 16 $octets$) on obtient T.

Le mode CTR assure le chiffrement des données. Cette opération utilise des blocs de flux de clé appelés S (S_i..S_n). Chaque bloc de flux de clé S_x est calculé en chiffrant un bloc A_x par l'algorithme AES en utilisant la clé K.

Une fois que les blocs de flux de clé sont obtenus, on chiffre les données en faisant un $\oplus$ entre P et S. Le bloc de flux de clé S_0 n'est pas utilisé pour le chiffrement, mais à la place, pour calculer U ($U \leftarrow first_M_Bytes(S_0) \oplus T$)

Le résultat final consiste en les données chiffrées C suivis par U.

7.1.3.2 Déchiffrement authentifié

L'opération de déchiffrement est décrite par l'Algorithme 13.

Algorithm 13: Déchiffrement authentifié du AES-CCM

begin
  for $(i = 0, i \leq n, i++)$ **do**
    $S_i = E(K, A_i)$
  // on sépare C et U, puis
  $T \leftarrow first_M_Bytes(S_0) \oplus U$
  for $(i = 1, i \leq n, i++)$ **do**
    $P_i \leftarrow S_i \oplus C_i$
    $P \leftarrow P || P_i$
  // on calcule T′ (comme T dans Algirthme 12), puis
  if $(T = T')$ **then**
    // opération de déchiffrement réussie
  else
    // opération de déchiffrement échouée

Afin de déchiffrer les données, nous avons besoin de 4 inputs, qui sont K, N, A, et les données chiffrées et authentifiées (C||U). D'abord, on recalcule les blocs de flux de clés (S) afin d'obtenir le plaintext P et la valeur T. Ensuite, en utilisant P et A on calcule T′ afin de le comparer avec T. Si T est valide alors l'opération de déchiffrement authentifié est réussie, sinon elle échoue.

7.2 Évaluation et résultats

Dans ce qui suit, nous allons établir une évaluation complète de notre protocole. En premier lieu, nous allons effectuer un modèle de menaces pour pouvoir étudier, analyser et relever les besoins sécuritaires des systèmes IoT. En deuxième lieu, nous allons montrer que notre approche permet de répondre à ces besoins. En troisième lieu, afin de prouver la robustesse et la sûreté de l'approche, nous allons utiliser un langage formel qui permet de valider, théoriquement, un protocole de sécurité (voir 7.2.3 page 98). En quatrième lieu, nous allons étudier les performances du protocole par rapport au réseau. Et enfin, en dernier lieu, nous allons étudier la consommation d'énergie de notre protocole.

7.2.1 Modèle de menaces

Afin de bien évaluer notre système, étudier les besoins en terme de sécurité, et définir les contre-mesures de protection nécessaires, nous avons créé un modèle de menaces, qui représente une procédure d'analyse de protocole de sécurité afin d'identifier ses objectifs et ses vulnérabilités. Pour faire cela, nous avons opté pour le modèle de *Dolev-Yao* [48] qui représente le modèle le plus utilisé. Ce modèle suppose que le réseau est formé par un ensemble d'entités qui échangent des données entre elles (ex. communication entre un device et un CPAN) en utilisant une technologie de communication sans fil non fiable (peut perdre des paquets). Il est également supposé que les entités communiquent dans un environnement non sécurisé. Cet environnement contient des utilisateurs malicieux (attaquants) qui capturent, modifient, interrompent, rejouent, forgent, réordonnent, et retardent des messages. Ces attaques sont expliquées dans la section 2 page 33. Nous supposons que les informations secrètes stockées sont protégées est ne peuvent pas être volées physiquement.

7.2.2 Services de sécurité de notre approche

Un système de sécurité doit satisfaire aux exigences décrites dans le modèle de menace (voir section 7.2.1).

La protection contre les attaques d'usurpation
Les deux entités communicantes s'authentifient mutuellement en utilisant des OTPs. Ces derniers sont basés sur des couples d'informations secrètes (non connues par l'intrus) et par un nombre aléatoire ou pseudo aléatoire unique et valide uniquement pour une seule utilisation ($(k_d, challenge)$ pour otp_1, et $(k_u, hiddenKeyBroadcast)$ pour otp_2). En plus k_d et k_u (dérivé de k_d) sont liés à l'identifiant de l'objet (voir mécanisme de personnalisation dans la section 5.1.1 page 80). Par conséquent, un noeud sans clé personnalisée ne peut ni être authentifié ni usurper l'identité d'un utilisateur légitime.

La protection contre les attaques de rejeu
Pendant la phase d'association, le fait que l'OTP n'est valide que pour une seule utilisation, protège le système contre les utilisateurs malveillants qui essayent de renvoyer (rejouer) les mêmes messages afin d'avoir une utilisation non autorisée du système.

Dans la phase d'échange de données via le canal sécurisé, les paquets de données utilisent des numéros de séquences (compteurs), et comme deux paquets ne peuvent pas avoir le même numéro, aucune attaque de rejeu ne peut être possible.

La protection contre les attaques de l'homme du milieu (MITM)
Notre protocole ne peut pas protéger un système lorsqu'un intrus interrompt le trafic pendant une communication. Cependant, comme expliqué dans le chapitre 6 page 87, dans l'étape d'association, si un intrus obtient tous les messages échangés, il ne peut pas les exploiter pour obtenir des informations secrètes ou pour falsifier un message.

Et au niveau du canal sécurisé, tous les messages échangés sont chiffrés et signés en utilisant le standard AES-GCM ou AES-CCM. Ces derniers sont recommandés par le *National Institute of Standards and Technology (NIST)* dans [52] et [51], due à leurs robustesses, légèretés, rapidités et optimalités en énergie.

La confidentialité persistante (forward secrecy)
Comme expliqué dans la section 2 page 32, la confidentialité persistante est une caractéristique cryptographique qui garantit que la découverte d'une information secrète (ex. clé secrète, clé privée, etc) d'un objet légitime par un utilisateur malicieux ne compromet pas la confidentialité des communications passées. Le fait que la valeur du IV change lors de chaque communication, un utilisateur malicieux qui possède une clé secrète mais qui ne connaît pas le bon IV qui a été utilisé ne peut pas déchiffrer les anciens messages. De plus les clés k_u sont renouvelées lors de l'établissement de chaque nouvelle session. Par conséquent notre protocole assure cette caractéristique.

La protection contre les attaques de cryptanalyse
Pendant l'étape d'association, k_i n'est utilisée que pour les opérations internes du CPAN (génération de k_d), et k_d n'est jamais utilisée sans qu'il y ai un nombre aléatoire ou pseudo-aléatoire et une fonction de hachage (irréversible).

En outre, le canal de communication est basé sur un algorithme de sécurité très robuste (AES-GCM/CCM), des clés de sessions éphémères, et des compteurs (ex. le compteur de l'IV). Par conséquent, ça protège le système contre toute attaque de cryptanalyse qui peut être déployée pour récupérer les clés ou les données échangées.

La protection contre les attaques par force brute
Retrouver les clés ou les données chiffrées par une attaque de brute force est presque impossible. Selon [28], trouver un mot de passe de 12 caractères (96 $bits$) en utilisant un hardware avec des bonne performance peut prendre 2 siècles. La probabilité P de trouver la bonne clé k_i ou k_d (256 $bits$) dès le premier coup est P = $\frac{1}{2^{256}}$, et pour k_u ou k_b (128 $bits$) est P = $\frac{1}{2^{128}}$.

La protection contre les attaques par dénie de service
Notre protocole n'est pas conçu spécifiquement pour protéger contre cette catégorie d'attaques. Cependant, grâce à l'utilisation optimale de la mémoire, il peut éviter certaines attaques visant à saturer la mémoire du CPAN.

Le Tableau 7.1 page suivante résume les caractéristiques de sécurité de notre protocole, où " Brt " signifie une attaque de force brute et " Crpt " signifie une attaque de cryptanalyse.

	Usurpation	Rejeu	MITM			Brt	Crp
			Interruption	Modification	Écoute		
Protection	✓	✓	✗	✓	✓	✓	✓

TABLEAU 7.1 – Les services de sécurité de notre protocole

7.2.3 Validation formelle

Afin de vérifier la robustesse et la sûreté de notre protocole, nous avons réalisé une validation formelle en utilisant un outil/langage de vérification automatique des protocoles de sécurité appelé *Scyther* [39]. Ce dernier a été créés par *Cas Cremers* à l'Université d'*Oxford*. Dans le langage formel de *Scyther*, chaque protocole est définie par des "rôles". Chaque rôle doit être joué par un "agent" et décrit par une séquence d'éventements (`send`, `receive`, etc). Dans ce qui suit, nous présentons la structure de notre code source.

```
la définition des nouveaux types
... etc

protocol  OCARIAuthAndEncProtocol(D,C)
{
function     OneTimePassword                                          ;
function     keyGeneration                                            ;
macro        otp1 = OneTimePassword(k(D,C),challenge)                 ;
// k(D,C) est la clé personnalisée entre D et C
macro        keyAuthAndEnc = keyGeneration(k(D,C), challenge)         ;
macro        otp2 = OneTimePassword(keyAuthAndEnc, hiddenKeyBroadcast);
hashfunction signFunc                                                 ;
macro        signature = signFunc(keyAuthAndEnc,otp1)                 ;
function     HiddeBroadCastKey                                        ;
function     RevealBroadCastKey                                       ;
const        deviceAuthError : String                                 ;
const        cpanAuthError   : String                                 ;
const        securedPacket                                            ;
```

`OCARIAuthAndEncProtocol` représente le nom du protocole, `function` et `macro` sont utilisés, respectivement, pour la définition des fonctions et l'abréviation des formules. Nous avons deux rôles joués par le device (D) et le CPAN (C). Les différentes opérations de transmission de données sont définies par les deux événements `send` et `receive`. `_id` est l'étiquette (label) de l'événement, il représente l'identifiant qui lie un événement `send` avec l'événement `receive` approprié de l'autre rôle. L'événement `match` est utilisé pour modéliser les tests d'égalité.

Les types de l'événement `claim` représentent les objectifs de la validation formelle. Pour valider la confidentialité des transmissions des données, on utilise la "claim" `Secret`. Et pour formaliser la confidentialité de la transmission des clés de session, il est préférable d'utiliser la "claim" `SKR` (Session Key Reveal). Nous avons également utilisé 3 "claim" pour valider l'authentification, qui sont `Alive` (pour l'existence), `Weakagree` (pour un accord faible) et `Niagree` (pour un accord non-injectif).

```
role D {
la déclaration des variables
locales, etc
recv_1(C,D,challenge)              ;
send_2(D,C,otp1)                   ;
recv_3(C,D,(receivedOtp2
            ,hiddenKeyBroadcast));

match(receivedOtp2,otp2)           ;
  // Ok, authentification réussie
  macro broadCastKey =
     RevealBroadCastKey(signature
              ,hiddenKeyBroadcast);
  send_4(D,C,
   ,{securedPacket}keyAuthAndEnc);

claim (D, SKR, keyAuthAndEnc)      ;
claim (D, SKR, broadCastKey)       ;
claim (D, Alive)                   ;
claim (D, Weakagree)               ;
claim (D, Niagree)                 ;
claim (D, Secret, securedPacket)   ;

}
```

```
role C {
la déclaration des variables
locales, etc
send_1(C,D,challenge)              ;
recv_2(D,C,receivedOtp1)           ;

match (receivedOtp1,otp1)          ;
  // Ok, authentification réussie
  macro hiddenKeyBroadcast =
    HiddeBroadCastKey(signature,
                      broadCastKey);
  // pour cacher broadCastKey
  send_3(C,D,(otp2
            ,hiddenKeyBroadcast));
  recv_4(D,C
  ,{securedPacket}keyAuthAndEnc);

claim (C, SKR, keyAuthAndEnc)      ;
claim (C, SKR, broadCastKey)       ;
claim (C, Alive)                   ;
claim (C, Weakagree)               ;
claim (C, Niagree)                 ;
claim (C, Secret, securedPacket);
}} ;
```

[103] explique les trois ”claim” d'authentification. Nous supposons que A est l'initiateur de la communication et B est le répondeur.

- `Alive` : nous considérons qu'un protocole garantit à A l'existence de B si, à chaque fois que A termine une exécution du protocole avec B, alors ce dernier a précédemment exécuté le protocole ;
- `Weakagree` : nous considérons qu'un protocole garantit à A un accord faible avec B si, à chaque fois que A termine une exécution du protocole avec B, alors ce dernier a déjà exécuté le protocole avec A ;
- `Niagree` : on considère qu'un protocole garantit à A un accord non-injectif avec B sur un ensemble d'éléments (paramètres) apparaissant dans la description du protocole si, à chaque fois que A termine une exécution du protocole avec B, alors ce dernier, exécutait le protocole avec A en agissant comme répondeur, et les deux agents (A et B) se mettaient d'accord sur les paramètres.

La Figure 7.3 page suivante montre les résultats obtenus après l'exécution de notre code *Scyther*. Nous avons effectué le nombre maximum d'exécutions (100 exécutions)

Les première, deuxième et troisième colonnes de la Figure 7.3 page suivante représentent respectivement : le nom du protocole, le rôle concerné (D ou C) et un identifiant unique de la ”claim”. La quatrième colonne représente le type de la ”claim” et ses paramètres.

Les deux dernières colonnes (”Status” et ”Comments”) affichent le résultat du processus de vérification (”Fail” ou ”Ok”) ainsi qu'une courte description. Le résultat ”No attack within bounds” doit être interprété par ”*Scyther* n'a trouvé aucune attaque en atteignant la limite du nombre d'exécution” [39].

Comme on peut le constater, la validation prouve la sûreté de notre protocole.

Claim				Status	Comments
OCARIAuthAndEncProtocol	D	OCARIAuthAndEncProtocol,D1	SKR keyGeneration(k(D,C),challenge)	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D2	SKR RevealBroadCastKey(signatureFunction(keyGenera...	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D3	Alive	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D4	Weakagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D5	Niagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,D6	Secret securedPacket	Ok	No attacks within bounds.
	C	OCARIAuthAndEncProtocol,C1	SKR keyGeneration(k(D,C),challenge)	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C2	SKR RevealBroadCastKey(signatureFunction(keyGenera...	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C3	Alive	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C4	Weakagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C5	Niagree	Ok	No attacks within bounds.
		OCARIAuthAndEncProtocol,C6	Secret securedPacket	Ok	No attacks within bounds.

FIGURE 7.3 – Résultats de la validation formelle

7.2.4 Performance par rapport au réseau

La résilience
On peut définir la résilience par la capacité d'un système à surmonter une altération de son environnement. Dans notre cas, si un objet est compromis, cela ne devrait pas influencer l'ensemble du réseau. Comme expliqué dans la section 7.2.2 page 97 et la section 7.2.2 page 97, il est très difficile de récupérer les clés ou les informations chiffrées échangées en effectuant une attaque par cryptanalyse ou par force brute. Par conséquent, un attaquant ne peut prendre le contrôle de l'ensemble du réseau que s'il obtient la k_i avec un accès physique au CPAN. Et dans le cas où nous protégeons (physiquement) seulement le CPAN, si un attaquant prend le contrôle d'un device, il peut envoyer des fausses informations au CPAN, mais il ne peut en aucun cas prendre le contrôle ou infecter d'autres devices, grâce au mécanisme de personnalisation (voir section 5.1.1 page 80).

La flexibilité
La flexibilité représente un point fort dans notre approche. En effet, contrairement à la plupart des approches vues dans le chapitre 3 page 35 et analysées dans la section 3.3 page 63, notre protocole permet (1) de rajouter un nouveau device (équipé par k_d) à un réseau sans avoir besoin de mettre à jour les données du CPAN. Et (2) grâce au mécanisme d'échange de clé de diffusion *hidden key broadcast* le device peut obtenir une clé de diffusion (k_b) pendant sont association au réseau.

L'évolutivité
Notre approche permet l'évolutivité du réseau, car le fait que le CPAN n'a besoin que de stocker sa clé k_i pour authentifier n'importe quel device légitime, rend notre protocole très évolutif. Par conséquent, le grand nombre d'objets n'a aucune influence sur les performances de sécurité.

La tolérance aux fautes D'un point de vue sécuritaire, tous les objets sont indépendants. Ainsi, même si un ou plusieurs devices se désactivent, la communication sécurisée des autres objets reste toujours maintenue.

7.2.5 Consommation d'énergie

L'optimisation de la consommation énergétique est l'un des défis les plus importants dans la conception de notre protocole. En effet, le protocole de sécurité devrait être optimisé afin de maximiser la durée de vie des objets et du réseau. Afin de pouvoir prouver l'efficacité énergétique de notre protocole, d'abord, (1) nous allons effectuer une estimation théorique de l'énergie consommée par un device lors de l'étape d'association de notre protocole. (2) Ensuite, nous allons présenter les résultats obtenus concernant l'énergie consommée par un device qui utilise la méthode d'association (handshake) du protocole *Datagram Transport Layer Security* (DTLS) appliqué sur un système WSN et déployé sur un hardware similaire [92]. D'après la RFC6347 [137], le DTLS est un protocole qui assure la confidentialité et l'intégrité des échanges d'un protocole de communication.

Remarque 1 : due à la particularité de la sous-couche MAC d'OCARI, nous ne pouvons pas comparer notre temps d'association sécurisée avec le DTLS handshake. De ce fait, nous présentons les résultats du DTLS handshake à titre indicatif.

Remarque 2 : les deux solutions proposent *AES-CCM* en tant qu'algorithme de chiffrement authentifiés pour le canal sécurisé. Dans notre cas, nous rajoutant la possibilité d'utiliser le mode GCM également. Ces standards sont fortement recommandés par le NIST [51][52], nous n'allons donc pas étudier l'énergie consommée par le canal sécurisé.

Dans cette étude, nous supposons que la communication est totalement fiable et que l'opération d'association est effectuée sans perte de paquets.

L'énergie peut être calculée en utilisant Équation 7.1.

$$\mathrm{E} = \mathrm{P} \times t \tag{7.1}$$

Où E est l'énergie (en mJ), P est la puissance (en mW), et t est le temps (en S). L'énergie consommée E comprend l'énergie consommée pendant la communication $\mathrm{E_C}$ et l'énergie de traitement $\mathrm{E_P}$. Ainsi, nous devons calculer la puissance de communication $\mathrm{P_C}$ et la puissance de traitement $\mathrm{P_P}$.

Puissance et consommation d'énergie de la communication

$\mathrm{P_C}$ peut être calculé en utilisant l'équation suivante (Équation 7.2) :

$$\begin{aligned}\mathrm{P_C} = \mathrm{N_T}[\mathrm{P}_{te}(\mathrm{T}_{on} + \mathrm{T}_{wu}) + \mathrm{P}_o \times \mathrm{T}_{on}] + \\ \mathrm{N_R}[\mathrm{P}_{re}(\mathrm{R}_{on} + \mathrm{R}_{wu})] + \mathrm{P}_{idle} \times \mathrm{T}_{idle}\end{aligned} \tag{7.2}$$

Où : P_{te} et P_{re} représentent la puissance consommée par l'émetteur et le récepteur, respectivement. P_o est la puissance de sortie de l'émetteur. T_{on} et R_{on} sont les durées d'activité de l'émetteur et du récepteur (lorsqu'ils sont en marche). T_{wu} et R_{wu} sont les durées de démarrage de l'émetteur et du récepteur. $\mathrm{N_T}$ et $\mathrm{N_R}$ représente le nombre de fois que l'émetteur et le récepteur sont en marche. Enfin, P_{idle} et T_{idle} sont la puissance consommée et la durée de temps du device quand il est inactif.

Le device testé est un Dresden Elektronik deRFsam3 23M10-R3, avec 48 ko de RAM, 256 ko de ROM, et un processeur Atmel SAM3S4A Cortex-M3. L'intensité I de $\mathrm{P}_{te} = \mathrm{P}_{re}$ est égal à 18 mA. La tension utilisée U est de 3.99 V. P_o est de 1 mW. P_{idle} est de 0.0012 mW.

$T_{wu} = R_{wu} = 0.032\ mS$.

$$P = UI \tag{7.3}$$

On se basant sur Équation 7.3 :
$P_{te} = P_{re} = 71.820\ mW$.

En utilisant une largeur de bande de canal de 250 $kbits/s$, et en considérant la taille maximale d'un paquet (136 $octets$) par *request / response* nous avons :
$T_{on} = R_{on} = 4.352\ mS$.

T_{idle} est calculé en utilisant Équation 7.4.

$$T_{idle} = T_{fullAssoc} - [N_T \times (T_{wu} + T_{on}) + N_R \times (R_{wu} + R_{on})] \tag{7.4}$$

Où $T_{fullAssoc}$ est le temps écoulé pendant toute l'opération d'association (la moyenne des durées mesurées).

$T_{fullAssoc} = 37.504\ mS$, et le device échange 4 messages ($N_T = 2$, et $N_R = 2$), donc en utilisant Équation 7.4, on obtient :
$T_{idle} = 19.968\ mS$.

Ainsi, en utilisant Équation 7.2, on obtient :
$P_C = 1268.163482\ mW$.

Et en se basant sur Équation 7.1, l'énergie de communication consommée par le device pendant le temps de communication $T_C = 0.034508$ S est :
$E_C = 43.761785\ mJ$.

Puissance et consommation d'énergie du traitement
[104] fournit un bon modèle pour calculer la puissance consommée par le traitement (voir Équation 7.5).

$$P_P \approx N \times C_{load} \times U^2 \times f \tag{7.5}$$

Où N est le nombre de cycles d'horloge par tache (clock cycles per task), C_{load} est la capacité de charge, U est la tension d'alimentation, et f est la fréquence d'horloge.

Tout d'abord, nous devons calculer le N correspondant à toutes les tâches exécutées lors de l'étape d'association Pour calculer otp_1, key_u, $signature$ et otp'_2 nous utilisons HMAC 4 fois pour des données d'entrée (input) de, respectivement, 64 $octets$, 64 $octets$, 20 $octets$ et 32 $octets$, et en se basant sur [114], on obtient $N_{HMAC} = 1476$.

On se référant du manuel du processeur *ARM Cortex-M3* [43], l'opération $\oplus$ et celle de la comparaison (comp), chacune prend 1 $clock\,cycle$ par $octet$. Pour générer k_b on utilise l'opération $\oplus$ sur une données de 16 $octets$ ($N_\oplus = 16$), et on compare une donnée de 4 $octets$ entre otp' et otp ($N_{comp} = 4$).
En conséquence, le total N = 1496 $clock\ cycles$.

Sachant que $C_{load} = 7.5\ pF = 7.5 \times 10^{-12}$ F, U = 3.99 V, et $f = 32\ kHz$ (32000 Hz),et en utilisant Équation 7.5, la puissance consommée par le traitement est :
$P_P = 5.716 \times 10^{-3}$ W = $5.716\ mW$.

Enfin, l'énergie de traitement consommée par le device pendant le temps de traitement T_P = 0.002996 S :
E_P = 0.017125 mJ.

L'énergie totale consommée par notre protocole est E = E_C + E_P, donc :
E = 43.77891 mJ.

DTLS handshake
E′ est l'énergie consommée par un device utilisant l'approche DTLS, qui représente la consommation d'énergie de l'association – la consommation d'énergie du serveur. En se basant sur les résultats obtenu par [92] (avec une clé de 2048 $bits$) :
E′ = 389.1 mJ .

Le Tableau 7.2 résume les résultats obtenus.

Association	**Notre approche**	**Approche basée sur DTLS [92]**
Nombre de messages	4	6
Temps d'exécution	37.504 mS	4000 mS
Total de consommation énergétique du device	43.77891 mJ	389.1 mJ

TABLEAU 7.2 – Résultats d'évaluation

D'après ces résultats, on constate que notre protocole est économique en énergie et consomme beaucoup moins d'énergie que l'approche basée sur le DTLS. En effet, l'approche DTLS utilise des algorithmes asymétriques, qui consomment généralement beaucoup de temps et d'énergie. Notre protocole est optimisé, il ne nécessite pas l'échange d'un grand nombre de messages et est basé sur des algorithmes robustes et légers, ce qui le rend très bien adapté aux exigences de l'IoT.

7.3 Conclusion

Tout comme la version précédente (voir chapitre 6 page 87), celle-ci offre une méthode légère et robuste d'authentification mutuelle et fournit un bon mécanisme d'échange de clé. Elle sert à créer un canal de communication qui protège non seulement l'intégrité des données, mais sa confidentialité également. Nous avons prouvé la sûreté du protocole via un modèle de menaces et une validation formelle. Nous avons montré les performances de notre approche par rapport au réseau. Et enfin, nous avons affirmé son optimalité, et son efficacité en énergie.

Comparé aux autres approches qu'on a étudiées (voir section 3.3 page 63), notre solution assure une flexibilité et une transparence lors de l'association d'un device à un réseau, car elle ne nécessite pas la mise à jour du CPAN, et toute l'opération d'association sécurisée se fait automatiquement. Malgré cette flexibilité, cette version (Version 3) ne permet pas la migration sécurisée d'un device d'un réseau vers un autre réseau.

Dans le chapitre suivant (chapitre 8 page 105), nous allons rajouter un mécanisme d'authentification décentralisé, basé sur les *blockchains* privées, qui permet la migration sécurisée des objets et d'avoir une vue globale sur les différents réseaux.

Chapitre 8

Version 4 (BCTrust) : Système d'authentification décentralisé (*blockchain* privée)

8.1 Approche

La version 3 du protocole (voir chapitre 7 page 91) assure les services de sécurité de base, cependant, la mobilité des devices est absente. En effet, si un device a_x appartenant à un réseau géré par le CPAN A veut migrer vers un autre réseau géré par B, alors a_x ne pourra pas s'authentifier et s'associer au réseau de B car il ne possède pas la clé appropriée (voir Figure 8.1).

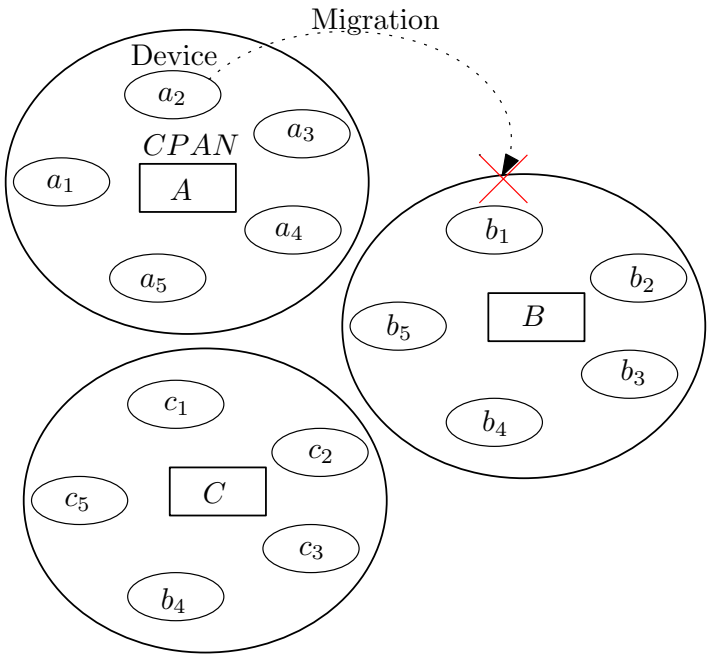


FIGURE 8.1 – Problème de migration sécurisée des devices

L'une des solutions qu'on peut imaginer est de mettre les clés des n réseaux au niveau de chaque device mobile (dans la Figure 8.1 $n = 3$), afin que ce dernier puisse s'authentifier et s'associer à ces n réseaux, en utilisant la clé appropriée. Cependant cette solution n'est pas optimale car elle sature la mémoire des objets. En plus, elle limite la mobilité des devices par un nombre pré-configuré de réseaux possibles, et si un nouveau réseau ap-

paraît, alors il faut mettre à jour le device par la clé correspondante à ce nouveau réseau. Donc pour palier à ce problème, il faut concevoir un système d'authentification décentralisé, transparent et évolutif, qui n'impacte pas négativement les performances des objets ni celles des réseaux. Ce système doit fournir une liberté totale de déplacement d'un device d'un réseau à un autre, tout en étant assurer la sécurité de ces derniers.

De ce fait, nous avons conçu une nouvelle approche appelée *BCTrust* basée sur le principe de « *l'ami de mon ami est mon ami* ». Ce qui veut dire que si un device est authentifié dans un réseau légitime, il devient fiable et accepté par tous les autres réseaux. Ce qui permet d'avoir une vision globale sur tout les réseaux. Pour réaliser ce principe, nous avons opté pour les *blockchains privées* (voir section 4.1.2 page 71), pour les raisons suivantes :

— dans cette catégorie de *blockchains*, les transactions ne sont effectuées que par des objets légitimes et connus par la *blockchain*. Par conséquent, la validation de ses transactions ne nécessite pas l'utilisation d'un mécanisme de validation tel que la *PoW* ou *PoS* (voir section 4.1.1 page 70), ce qui permet (1) un temps de validation très court, et (2) de ne pas consommer beaucoup de ressources ;

— généralement les transactions dans les *blockchains* privées sont gratuites.

Comme expliqué dans le chapitre 4 page 69, la *blockchain* garantit l'intégrité, la fiabilité et la disponibilité des informations stockées. Notre solution est déployée sur la *blockchain Ethereum*, qui fournit l'un des plus grands registres (*ledgers*) au monde, a une grande communauté qui le suit, et représente une plateforme pour les applications décentralisées. *Ethereum* assure des transactions sécurisées, et comme dans le cas de *Bitcoin*, il est testé à grande échelle. Enfin, nous pouvons prendre une copie de sa chaîne actuel et utiliser librement *Ethereum* en local. Nous avons ajouté des restrictions la *blockchain Ethereum*, qui la transforme d'une *blockchain* publique en une privée (en utilisant des modificateurs (*modifiers*) [57]). Dans le *smart contract*, seul un ensemble d'objets de confiance a les droits d'écriture sur la *blockchain*. Ces objets privilégiés sont les CPANs des réseaux. Chaque CPAN a une paire de clé privée/publique, ce qui lui permet de faire des transactions en toute sécurité avec la *blockchain*. La Figure 8.2 explique le principe de fonctionnement de *BCTrust*.

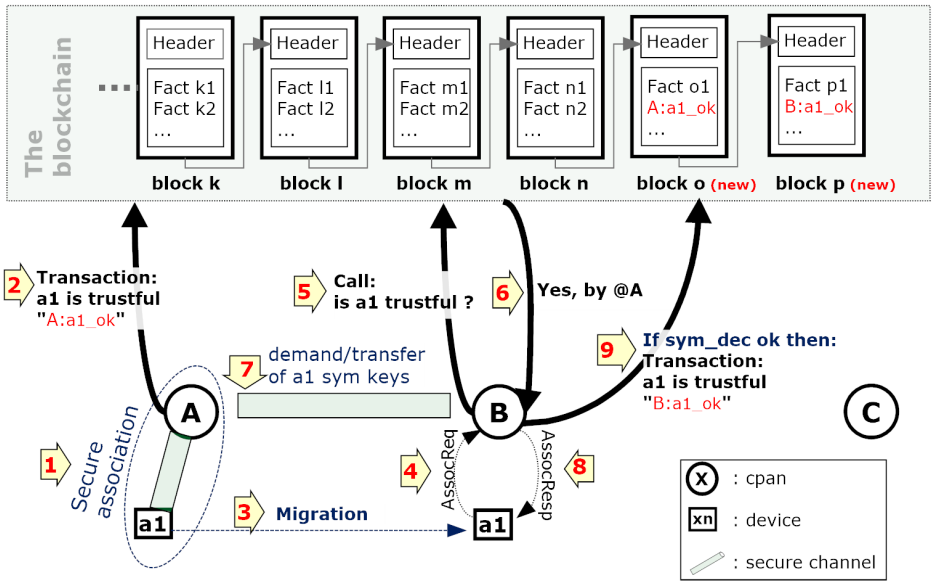


FIGURE 8.2 – Le mécanisme de fonctionnement de *BCTrust*

Nous avons un device a_1 appartenant au réseau géré par le CPAN A. (1) La première association sécurisée de a_1 avec A est s'effectue comme expliqué dans le chapitre 7 page 91 (version 3). (2) Une fois que le device est authentifié, A envoie une transaction, signée par sa clé privée, à la *blockchain* avec comme information "A : a1_ok", ce qui signifie que A est le garant que a_1 est de confiance, et que A possède les clés symétriques de a_1. Cette transaction est stockée dans un nouveau bloc validé par les CPAN participants. (3) Lorsque a_1 migre vers le réseau géré par B, (4) il envoie une *requête d'association* à B. (5) B envoie un *call*[1] à la *blockchain* afin de vérifier si a_1 était authentifié avant ou non. (6) Quand

Algorithm 14: Définition des paramètres et fonctions

```
parameter : X : Object // X est le CPAN
            d : Object // d est le device
            IV : Key   // IV est le vecteur d'initialisation incrémenté
            k_u : Key  // k_u est la clé symétrique (unicast)
            k_b : Key  // k_b est la clé symétrique (broadcast)
            msg : Message
            encMsg : Message
            req : Message
            resp : Message
            authList : AddressList
            authenticator : Address
            channel : Channel
            BC : Blockchain
            const ok : State
            const auth_req : String
Function  : LastAddress(AddressList list)
// retourne l'adresse du dernier CPAN dans la liste
Function  : FirstAssociation()
// une authentification basée sur la version 3
// de notre approche ( 7 page 91), et création de k_u et IV
Function  : SecChannel(Object obj1, Object obj2)
// établissement d'un canal sécurisé entre obj1 et obj2
Function  : SymEncrypt(Message msg, Key k)
// chiffrement symétrique du msg en utilisant k
Function  : SymDecrypt(Message msg, Key k)
// déchiffrement symétrique du msg en utilisant k
Function  : Receive(Message msg, Address from)
// réception de msg de la part de from
Function  : Send(Message msg, Address to)
// envoie de msg à to
```

B est informé par la *blockchain* que a_1 est fiable et qu'il a été authentifié par A, (7) il demande à A -via un canal sécurisé asymétriquement- les informations secrètes de a_1, qui sont la dernière k_u utilisée et le IV incrémenté par le compteur (pour la prochaine opération de chiffrement/déchiffrement). (8) B envoie une *réponse d'association* contenant la clé de diffusion k_b de son réseau. Cette clé est chiffrée avec AES-GCM/CCM en utilisant k_u. Enfin, B associe le device, et (9) envoie la transaction "B : a1_ok" après une opération de chiffrement/déchiffrement réussie avec a_1. Si a_1 veut à nouveau migrer vers le

1. un *call* est une transaction qui ne modifie pas l'état de la blockchain. Elle est utilisée généralement pour récupérer une information de la *blockchain*.

réseau de C, ce dernier demande (*call*) à la *blockchain* des informations à propos de a_1, et le résultat sera ["A : a1_ok","B : a1_ok"], dans ce cas C prend le dernier CPAN (dans cette exemple c'est B) pour obtenir les clés symétriques de a_1. Le fait que la valeur de IV change après chaque opération de chiffrement/déchiffrement protège les données et les clés symétriques (k_u et k_b) contre les attaques de rejeu et de cryptanalyse. Il est également possible de mettre à jours les clés k_u et k_b. Cette opération peut être réalisée par le CPAN qui -après une durée d'expiration des clés- génère, chiffre (avec la k_u courante), et envoie des nouveaux clés.

Le mécanisme de migration est résumé par les Algorithmes 14, 15, et 16.

Algorithm 15: Mécanisme de migration, côté device

```
begin
    d← this
    Send (req, X.address)
    resp ← Receive (msg, X.address)
    if (resp = auth_req) then
        FirstAssociation ()
    else if (SymDecrypt (resp, k_u) = ok) then
        // récupération de k_b
        // association réussie
    else
        // association échouée
```

Algorithm 16: Mécanisme de migration, côté CPAN

```
begin
    X ← this
    req ← Receive (msg, d.address) // réception d'une requête d'association
    authList ← BC.SendCall (d.address) // retourne la liste des
        authentificateurs du device
    authenticator ← LastAddress (authList)
    if (authenticator = NULL) then
        FirstAssociation ()
    else if (authenticator = X.address) then
        Send (SymEncrypt (resp[k_b]), d.address) // réponse d'association
            contenant k_b
        // association réussie
    else
        channel ← SecChannel (X, authenticator)
        k_u ← channel.getKey (d.address)
        IV ← channel.getIV (d.address)
        Send (SymEncrypt (resp[k_b]), d.address) // réponse d'association
            contenant k_b
        // association réussie
    wait() // attente pour recevoir d'autres messages
    encMsg ← Receive (msg, d.address) // encrypted msg
    if (SymDecrypt (encMsg, k_u) = ok) then
        BC.SendTransaction (X, d.address, ok)
```

8.2 Évaluation et résultats

8.2.1 Cadre expérimental

Notre approche est basée sur la *blockchain*, donc elle est renforcée par tous les avantages que cette dernière offre :

1. le déni de service et le déni de service distribué (dos/ddos) représentent la catégorie d'attaque réseau la plus déployée et la plus dangereuse qui vise à empêcher la disponibilité d'un service En raison de la nature décentralisée et dupliquée de la blockchain, il est très difficile, voir impossible, de cibler ce type d'architecture avec une tel catégorie d'attaque ;
2. l'utilisation de *Elliptic Curve Digital Signature Algorithm* (ECDSA) assure des transactions rapides et robustes [83] ;
3. grâce au mécanisme de consensus, les données stockées ne peuvent pas être falsifiées et les opérations établies ne peuvent pas être répudiées ;
4. notre approche fournit une vue globale sur tous les objets communicants. Ainsi, l'opération de migration est réalisée de manière sûre, transparente, avec une grande flexibilité, et sans impliquer les administrateurs humains.

Pour évaluer notre approche, avons utilisé une architecture réseau composée d'un device a_1 et deux CPANs A et B. Nous sommes conscients qu'un vrai réseau IoT contient un plus grand nombre d'objets. Cependant, le but de notre étude est de donner une preuve de concept de notre approche. Plus précisément, nous visons à prouver (1) la robustesse de notre système et sa flexibilité concernant la migration des devices et (2) en évaluant le temps d'exécution et l'énergie consommée, on peut vérifier l'optimalité de l'approche et son adaptation aux objets IoT. C'est pourquoi, l'utilisation d'une telle architecture est plus que suffisante.

Nous avons effectué cette expérience comme suit :

— nous avons utilisé *TestRPC*[2] [2] pour déployer une *blockchain Ethereum* en local munie d'un client qui interagit avec ;

— nous avons implémenté le code source du programme des objets en langage C ;

— pour que les objets puissent interagir avec la *blockchain* via *TestRPC*, nous avons créé une interface qui permet aux programmes écrits en C de communiquer avec le client *TestRPC* en mode d'appel de procédure à distance, et l'inciter à interagir avec la *blockchain*. Autrement dit, notre interface encode/décode les données entre les programmes écrits en C et *Ethereum*. Cette interface utilise *JSON RPC* pour la communication, où *JSON* est un format de données textuel standard et *RPC* un système d'appel de procédure à distance ;

— après avoir préparé les outils et les programmes nécessaires, nous avons créé notre *smart contract* en langage Solidity [57].

La Figure 8.3 page suivante illustre les différentes couches de l'architecture de *BCTrust*.

Lorsque on démarre *BCTrust*, on compile le contrat puis on le stocke dans la *blockchain* via une transaction. Si cette dernière est valide, la *blockchain* renvoie une adresse spéciale appelée l'adresse du contrat. Cette adresse est utilisée pour localiser le *smart contract* dans la *blockchain* afin de pouvoir interagir avec lui.

2. TestRPC : est un outil utilisé généralement pour tester et développer les applications décentralisées basées sur *Ethereum*.

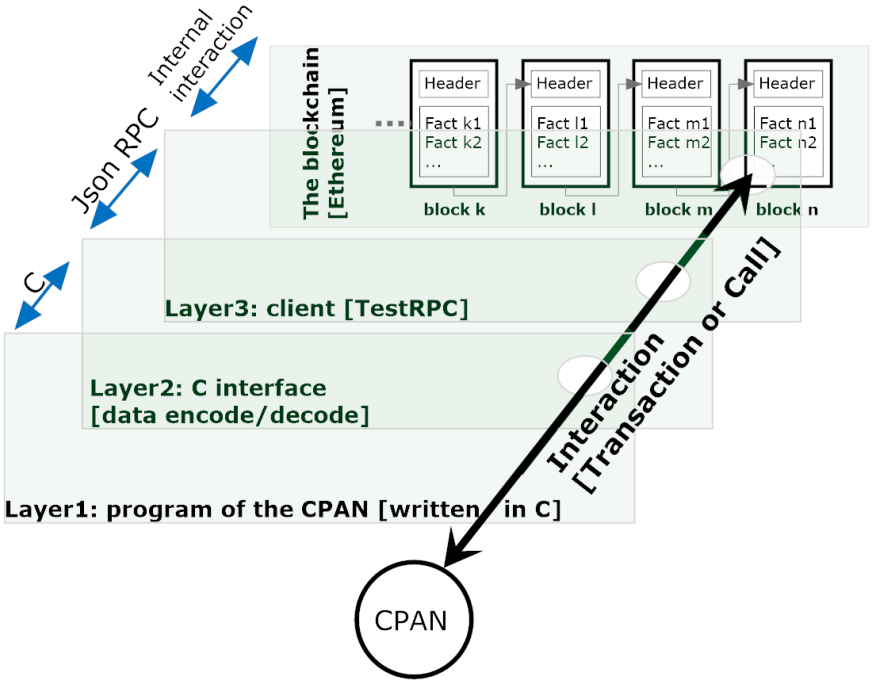


FIGURE 8.3 – Architecture en couche de *BCTrust*

8.2.2 Test et résultats

Dans cette étude nous nous intéressons pas à l'évaluation (temps d'exécution et consommation d'énergie) des CPANs, car ils sont considérés comme illimités en ressources. De ce fait, afin d'évaluer les performances du mécanisme de migration, nous avons mesuré le temps d'exécution et calculé la consommation d'énergie d'un device dans les cas suivants :

1. la $1^{ère}$ association d'un device à son réseau principal. Cette opération se fait en utilisant la version 3 de notre protocole (voir chapitre 7 page 91). Appelons cette association sécurisée "*classical association*" ;
2. la $n^{ème}$ association ($n \geq 2$) d'un device à son réseau principal (sans migration). Elle utilise le mécanisme d'association fourni par *BCTrust*. Appelons cette association sécurisée "*smart association*" ;
3. l'association d'un device pendant une opération de migration. Elle utilise le mécanisme d'association fourni par *BCTrust*. Appelons cette association sécurisée "*smart migration*".

La démonstration était réalisée sur un réseau réel composé d'ordinateurs représentant le device a_1 et les deux CPANs A et B. Le Tableau 8.1 page suivante décrit les caractéristiques des objets utilisés. Pour avoir des résultats plus correctes lors de l'évaluation du temps total d'association, (1) nous avons pris les résultats des associations des vrais capteurs (devices) avec des CPANs qu'on a obtenus dans le chapitre 7 page 91. Ensuite, (2) nous avons calculé le temps de communication entre "un CPAN avec un autre CPAN" via un canal sécurisé par l'algorithme RSA [139], et entre "un CPAN et la *blockchain* (transaction/call)". Enfin (3) on a calculé le temps total d'association. Nous avons réalisé 10 tests de *classical association* à "un saut" d'un device a_1 avec un CPAN A (voir chapitre 7 page 91), ensuite, 10 tests de *smart association* entre a_1 et le CPAN A, et enfin 10 tests de *smart migration* (migration du device a_1 du réseau géré par CPAN A vers le réseau de B). Le

Objet	Mémoire	Processeur (CPU)	Type du système d'exploitation
A	7,8 G*o*	Intel Xeon(R) CPU E5-1607 v3 @ 3.10GHz	Linux 32-bits
B	2 G*o*	Pintium(R) Dual-Core CPU T4400 @ 2.2GHz	Linux 64-bit
a_1	48 K*o*	Cortex-M3	✗

TABLEAU 8.1 – Matériels utilisés

diagramme de la Figure 8.4 présente les temps d'exécution totaux des associations. Pour mieux comprendre ces résultats, dans Tableau 8.2, on calcule les moyennes des temps d'exécution de chaque type d'association. *bc* signifie *blockchain*. Il est à noter que l'écart type "σ" des associations que nous avons effectuées pendant les tests est négligeable pour le mode sans sécurité et la *smart association*. Pour la *classical association* : $\sigma_c = 1.23$ *m*S, et pour la *smart migration* : $\sigma_{sm} = 1.43$ *m*S.

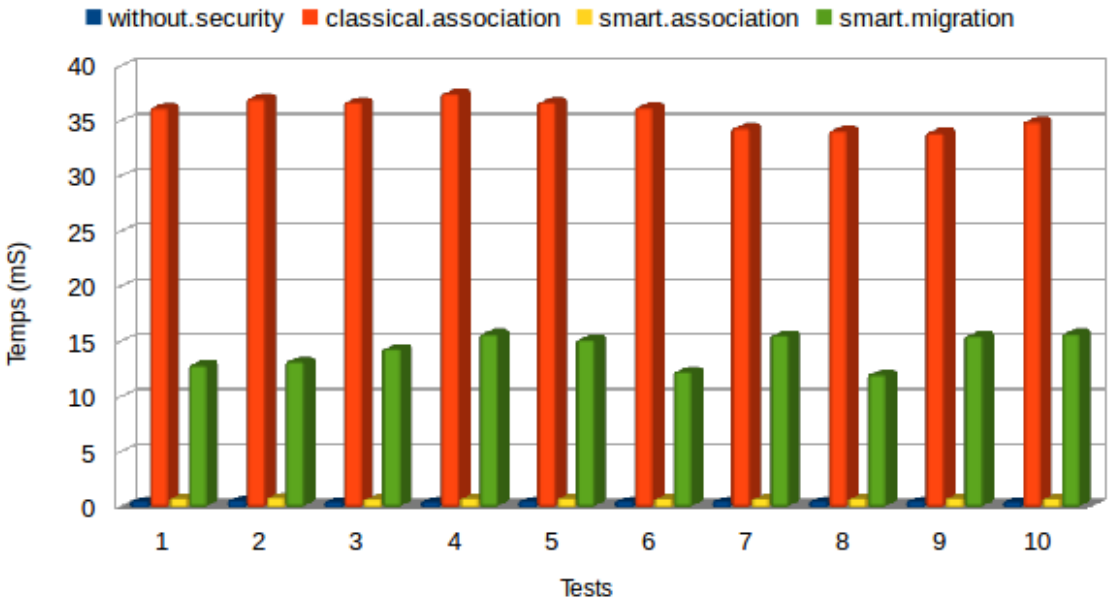


FIGURE 8.4 – Temps d'exécution totaux des associations

without security									
0.52									
classical association			**smart association**			**smart migration**			
$a_1 \leftrightarrow$ A	A $\leftrightarrow$ *bc*	*total*	$a_1 \leftrightarrow$ A	A $\leftrightarrow$ *bc*	*total*	$a_1 \leftrightarrow$ B	B $\leftrightarrow$ *bc*	B $\leftrightarrow$ A	*total*
37.50	0.25	37,75	0.52	0.25	0.77	0.61	0.25	13.40	14.26

TABLEAU 8.2 – La valeur moyenne du temps d'exécution de chaque type d'association en *m*S

D'abord, dans le mode sans sécurité (*without security*), un device échange seulement 2 messages avec un CPAN pour être associé à son réseau. Il envoie une *requête d'association* et reçoit une réponse, sans qu'il y ai des calculs.

Ensuite, dans la *classical association*, un device échange 4 messages avec le CPAN. Les deux entités s'authentifient mutuellement en utilisant des OTP, et génèrent des clés

symétriques. Le temps d'exécution de ce mode est égal à près que 72,6 fois le temps du mode non sécurisé.

Après, dans la *smart association*, le temps d'exécution est très optimisé, il représente seulement 1.48 fois le temps d'exécution dans le mode non sécurisé. Ce temps représente la durée d'échange de 2 messages seulement, entre le device et le CPAN (sans sécurité), plus la durée de validation de la transaction du CPAN par la *blockchain*. On peut constater que la durée de validation d'une transaction utilisant une copie locale d'une *blockchain Ethereum* privée prend en moyenne 0,000250 S, car dans ce cas on utilise pas un mécanisme de validation (voir section 4.1.1 page 70), comparé à la *blockchain Ethereum* publique qui met 14 S pour valider une transaction.

Enfin, dans la *smart migration*, le temps d'exécution est aussi optimal. Il représente 27.42 fois le temps d'association dans le mode non sécurisé, et nous fait gagner 62,22% du temps utilisé par la *classical association*. Cette optimalité en temps s'explique par le fait que le device échange uniquement 2 messages avec le CPAN (une *requête d'association* sans sécurité, et une *réponse d'association* qui nécessite une opération de chiffrement/déchiffrement authentifié de la clé k_b), la durée des transactions est très courte, et la communication entre CPANs -en utilisant l'architecture TCP/IP [85]- est très rapide, car les CPANs ne sont pas contraints en ressources et utilisent des hautes performances.

Maintenant pour prouver l'efficacité de *BCTrust* en énergie, nous calculons celle consommée par un device (ex. a_1) pour les différents types d'association. En utilisant les équations expliquées dans la section 7.2.5 page 101, nous avons obtenu les résultats suivants (voir Tableau 8.3).

	without security	**classical association**	**smart association**	**smart migration**
messages échangés	2	4	2	2
P_C (mW)	634.059862	1268.163482	634.060162	634.076350
P_P (mW)	-	5.715953	-	169.285428
T_C (S)	0.00052	0.034508	0.00052	0.00052
T_P (S)	-	0.002996	-	0.00009
E_C (mJ)	0.329711	43.761785	0.329711	0.329720
E_P (mJ)	-	0.017125	-	0.015236
E (mJ)	0.329711	43,77891	0.329711	0.344956

TABLEAU 8.3 – L'énergie consommée par le device de chaque type d'association

Où : P_C représente la puissance de communication, P_P est la puissance de traitement, T_C est le temps d'exécution de l'opération de communication entre le device et le CPAN, T_P est le temps de traitement, et E est l'énergie totale consommée pendant la durée d'association du device, qui représente la somme de l'énergie consommée lors du traitement E_P et celle de la communication E_C.
Remarque 1 : l'énergie perdue du device lorsqu'il est inactif pendant l'association est prise en compte dans cette évaluation.
Remarque 2 : "-" signifie une valeur négligeable.
Remarque 3 : pour obtenir E_P de la *smart migration*, nous avons choisi AES-GCM. Nous nous somme basé sur [24] (appliqué sur un hardware similaire au notre) afin d'obtenir le nombre de cycles d'horloge.

D'après les résultats du Tableau 8.3, on peut constater que l'énergie totale consommée par le device dans le mode non sécurisé et dans la *smart association* sont près que

identiques, car dans ces deux cas le device envoie seulement une *requête d'association* et reçoit une *réponse d'association* sans effectuer des traitements de sécurité. La petite différence s'explique par le fait que P_{idle} de la *smart association* est légèrement plus grand que celui du mode *without security*.

La consommation d'énergie augmente un peu dans le cas de la *smart migration*. Ceci est due à l'opération de déchiffrement AES-GCM de la clé de diffusion k_b par le device à la réception de la *réponse d'association*. Enfin, on remarque qu'à partir de la $n^{ème}$ opérations ($n \geq 2$), *BCTrust* permet au device d'économiser beaucoup d'énergie. En effet, la *smart association* et la *smart migration* permettent au device d'économiser, respectivement, ≈ 99,25% et ≈ 99,21% d'énergie par rapport à la *classical association*.

8.3 Conclusion

Cette version (version 4) du protocole peut être considérée comme étant finale, car non seulement elle assure tous les services de sécurité de base avec des bonnes performances, mais aussi, grâce à *BCTrust*, elle offre beaucoup plus de flexibilité et d'optimisation d'énergie et de temps pour les objets contraints. La migration des devices d'un réseau à un autre se fait d'une manière totalement transparente et sans aucune intervention humaine. Les CPANs on une vision globale sur tous les réseaux participants. Grâce à l'utilisation des *blockchains*, on peut avoir un système de traçage (logs) des opérations réseaux fiable, distribué et infalsifiable. L'utilisation des *blockchains* privées permet d'effectuer des transactions gratuitement, avec des délais de validation très courts.

Cependant, malgré la décentralisation qu'elle peut offrir aux systèmes, cette validation dépend uniquement d'un nombre limité d'objets validateurs, et l'ajout de nouveaux réseaux et leurs intégration à *BCTrust*, ne peut pas se faire d'une manière transparente. La version 4 du protocole est complète, en revanche si on aimerait avoir un système totalement décentralisé, qui ne limite pas le nombre de validateurs, et qui permet une hautes évolutivité du système, nous devons créer une autre approche alternative qui utilise une *blockchain* publique.

Chapitre 9

Version 5 (*Bubbles of trust*) : Système de sécurité décentralisé (*blockchain* publique)

9.1 Introduction

Comparé à la *blockchain* privée, la *blockchain* publique est payante, et la validation des transactions prend beaucoup plus de temps (14 secondes dans le cas d'*Ethereum*). Par contre les avantages de cette dernière, c'est qu'elle permet (1) une décentralisation totale du système, (2) une bonne évolutivité du réseau, et (3) plus de robustesse pour le service d'intégrité des données.

L'un des objectifs principaux de la *blockchain* publique, est que les objets participants n'ont pas besoin de s'authentifier pour effectuer des transactions, car c'est la *blockchain* elle même qui assure la légitimité de ses transactions. Cependant, si on veut appliquer ce mécanisme sur un système qui nécessite l'identification des objets des réseaux, cette anonymisation devient plutôt un inconvénient.

Dans ce dernier chapitre de cette partie, nous allons présenter la deuxième alternative de décentralisation de notre approche (avec celle du chapitre 8 page 105), appelée *BCTrust.v2* ou *BBTrust*. L'appellation *BBTrust* est l'abréviation de *Bubbles of trust* qui veut dire en Français *Bulles de confiance*. On peut imaginer ses bulles comme étant des zones virtuelles, permettant les objets qui y sont de s'identifier, de s'authentifier, de se faire confiance et de communiquer d'une manière sécurisée.

9.2 Le rapport entre *BBTrust* et nos approches précédentes

La Figure 9.1 page suivante explique ce que les anciennes versions du protocole et *BBTrust* peuvent fournir comme services pour sécuriser un système informatique. Elle montre également quelle est le lien entre cette version (Version 5) et les versions effectuées dans les chapitres précédents (Version 1 chapitre 5 page 79, Version 2 chapitre 6 page 87, Version 3 chapitre 7 page 91, et Version 4 chapitre 8 page 105). On peut constater que :

— un protocole qui est passé par les versions "1, 2, 3, et 4" ou "1, 2, 3, et utilise le service (d′) de 5", représente un protocole complet et adapté pour les objets limités en ressources. Il peut fournir tous les services de sécurité (a ou a′), (b), (c), (d ou d′). Dans ce cas, le protocole utilise la version 5 uniquement lors de la migration des

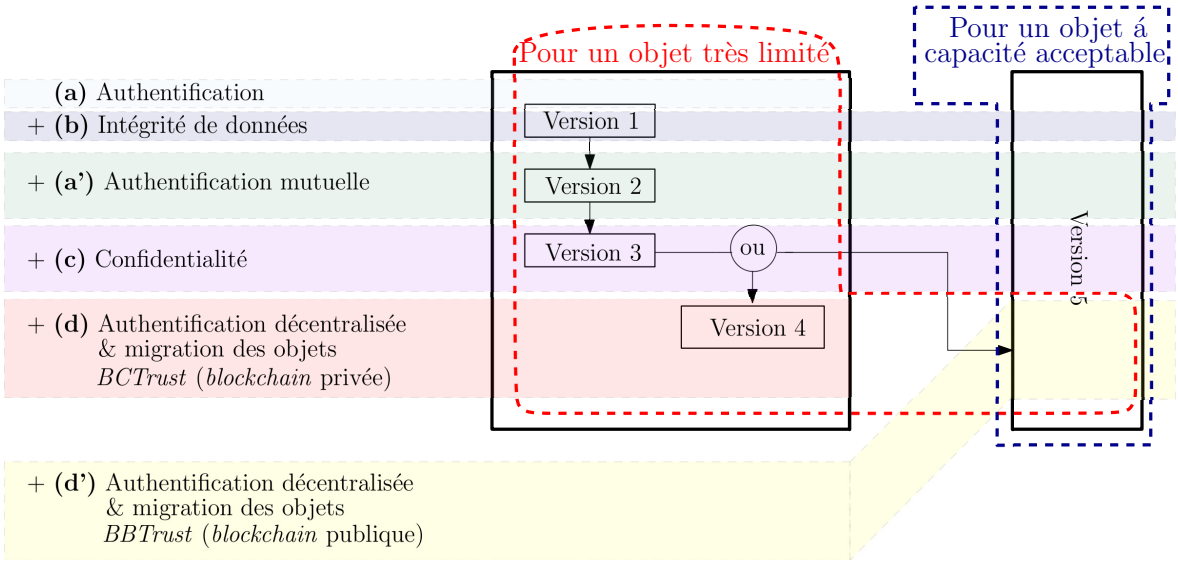


FIGURE 9.1 – *BBTrust* et ses services de sécurité.

objets, et c'est uniquement les CPANs qui envoient des transactions ;

— un protocole qui utilise uniquement la version 5 pour tous les services de sécurité (a′), (b), (c), (d′) est un protocole complet, et est adapté uniquement pour les objets ayant des capacités acceptables en ressources. En revanche, il offre plus d'évolutivité aux systèmes.

Dans ce qui suit, nous allons expliquer en détails le principe de fonctionnement de cette approche (*BBTrust*), les avantages qu'elle offre, et montrer comment elle peut sécuriser n'importe quelle architecture réseau utilisant n'importe quelle topologie. Mais, comme on a expliqué au-dessus, elle ne peut être déployée que sur des objets ayant des performances acceptables, et qui leurs permettent d'effectuer des transactions signées avec ECDSA [83]. Donc dans notre travail de recherche, nous allons appliquer cette approche sur les CPANs qui ne sont pas contraints en ressources.

Avant de commencer à détailler l'approche, nous allons d'abord définir les exigences de sécurité, puis établir notre modèle de menace (voir section 9.4 page ci-contre) afin de mieux comprendre les besoins en sécurité pour la protection des systèmes IoT. Ensuite, nous allons présenter brièvement quelques travaux connexes dans la section 9.5 page 119, et enfin, nous décrivons notre approche dans la section 9.6 page 121.

9.3 Éxigences de sécurité

En plus des services de sécurité expliqués dans le chapitre 2 page 31 (ex. authentification, intégrité, confidentialité, etc) , l'**identification** est une exigence primordial dans la majorité des cas d'utilisation de l'IoT. Elle représente le contraire de l'anonymat qui garantit qu'une entité communicante cache son identité par rapport aux autres entités du système. Par exemple, dans un scénario de parking intelligent, lorsqu'un capteur d'une place de stationnement envoie une notification au système de gestion, ce dernier, doit savoir exactement quel capteur communique afin de mettre à jour avec précision l'état des places de stationnement. Un autre exemple, dans un système de surveillance d'environnement, lorsqu'un capteur envoie à la plateforme de surveillance une information concernant le niveau d'eau d'un lac, ou concernant la détection de fumée dans une forêt, la plateforme doit connaître avec exactitude quel est le capteur qui a envoyer l'informa-

tion afin de connaître son emplacement et d'agir. Encore un dernier exemple, dans un réseau composé d'ordinateurs qui offre des services web, afin qu'un objet accède à une information ou une page web, il doit connaître l'adresse (identité) du serveur.

En effet, il est clair maintenant que l'identification dans ce cas de figure est essentiel. En revanche, pour qu'un identifiant soit valide, il doit être associé à une preuve qui le protège et qui garantit son intégrité. Le certificat x509 [77] en est un exemple. Ce dernier, permet d'associer un identifiant appelé le *common name* avec une paire de clés cryptographiques (publiques/privées). Dans la section 9.6.1 page 121, nous allons créer notre propre certificat, qui est très léger et conçu spécialement pour les objets connectés. Nous avons appelé ces certificats *tickets*.

9.4 Modèle de menace

Dans cette section nous présentons notre modèle de menace qui est basé sur celui de *Dolev-Yao* [48].

9.4.1 Modèle de réseau :

L'objectif principal de l'authentification est de permettre aux objets légitimes d'utiliser/fournir différents services et de communiquer d'une manière sûre et sécurisée via un réseau non sécurisé.

Á travers une architecture centralisée ou distribuée, chaque objet peut communiquer et échanger des messages avec un grand nombre d'autres objets participants. On suppose que ces messages traversent un réseau de communication non fiable (ex. Internet), qui peut avoir des pertes de paquets. Nous supposons également que tous les objets participants ne sont pas de confiance, et le nombre élevé d'objets dans le réseau augmente le risque d'inclure des objets compromis. De plus, les dispositifs existants sont de natures hétérogènes et n'appartiennent pas au même cas d'utilisation. La fonction réseau consiste uniquement à transmettre les paquets et ne fournit aucune garantie de sécurité telle que l'intégrité ou l'authentification. Ainsi, un utilisateur malveillant peut lire, modifier, supprimer ou injecter des messages dans le réseau.

9.4.2 Modèle d'attaque :

Dans cette étude, on suppose qu'un attaquant possède le contrôle total du réseau. Il peut sélectivement capturer, supprimer, rejouer, réorganiser, injecter, retarder, et modifier des messages. L'attaquant peut bénéficier d'une puissance de calcul et de stockage plus importante que les dispositifs du réseau. En revanche, on suppose que les objets sont protégés contre les attaques physiques [1], car il existe de nombreuses méthodes pour les protéger contre de telles attaques [26] (eg. en rendant ces informations lisibles uniquement par l'appareil lui-même).

9.4.3 Attaques :

Un attaquant peut avoir plusieurs objectifs, tel que l'obtention d'un accès au système, le vol d'informations, l'interruption d'un service, etc. Par conséquent, il peut générer dif-

1. Attaque physique : où un attaquant peut récupérer, physiquement, des informations secrètes comme les clés privées.

férents types d'attaques (voir section 2 page 33). A titre d'exemple, dans ce qui suit, nous allons citer quelques attaques connues :

Attaque Sybil : dans certains cas d'utilisation, l'attaquant simule l'existence de plusieurs entités (objets) qui envoient des informations erronées au serveur ou à l'application de gestion de services, afin de prendre des décisions dont l'attaquant a besoin. Le *Cooperative Intelligent Transportation System* (C-ITS) [11] représente l'une des cibles potentielles de cette attaque. En effet, dans un système C-ITS, les véhicules envoient continuellement de multiples informations à une infrastructure de gestion (ex. des *Cooperative Awareness Messages* (CAM) et des *Decentralized Environmental Notification Messages* (DENM) en suivant la norme Européenne, ou bien des *Basic Security Messages* (BSM) en suivant la norme Américaine). Ces informations concernent l'activité des véhicules ainsi que leur environnement, qui sont exploitées par le centre de gestion afin de fournir et d'améliorer de multiples services.

Par exemple, si le centre de gestion reçoit des messages de plusieurs véhicules informant d'un embouteillage ou d'un accident, alors il diffusera instantanément cette information à tous les véhicules de la zone pour les aider à trouver de meilleurs chemins. Ainsi, un attaquant peut envoyer de fausses informations au nom de plusieurs véhicules existants ou non existants afin de tromper les décisions du centre de gestion. Cette attaque peut être effectuée dans tous les cas d'utilisation nécessitant des informations provenant d'un certain nombre d'objets afin de choisir ou de prendre une décision quelconque.

Attaque par usurpation d'identité : contrairement à l'attaque Sybil, où l'attaquant essaie de créer de nombreuses identités fausses ou virtuelles, l'attaquant tente d'usurper l'identité d'un utilisateur légitime pour avoir ses privilèges.

Attaque de substitution de message : dans une attaque de substitution, l'attaquant intercepte des messages valides pendant leurs transmissions et les modifie de telle façon que les destinataires acceptent les messages contrefaits comme s'ils avaient été envoyés par l'expéditeur d'origine.

Déni de service : un déni de service (DoS) ou un DoS distribué (DDoS) est caractérisé par la tentative explicite de l'attaquant d'empêcher l'utilisation légitime d'un service [117]. Comme expliqué dans la section 2 page 33, il existe deux méthodes pour mener une telle attaque. (1) En exploitant une faille au niveau du protocole et (2) en inondant la cible. Les attaques DDoS et surtout les attaques par inondations comptent parmi les cyberattaques les plus dangereuses. Leur popularité est due à leur grande efficacité contre tout type de services, car elles n'exigent aucune identification/exploitation des failles d'un protocole ou d'un service donné, mais plutôt de les inonder tout simplement. Une attaque DDoS contre le mécanisme d'authentification provoquera des dommages importants tels que la paralysie de tout un système ou l'autorisation des utilisateurs illégitimes à utiliser le système.

Attaque par rejeu : un attaquant peut enregistrer sélectivement certains messages et les rejouer (sans modification) ultérieurement, car la vérification réussie d'un message ne certifie pas l'exactitude de l'heure (ou numéro de séquence) de son envoi. De cette manière, des informations inexactes peuvent être intentionnellement fournies aux objets.

Remarque : dans ce chapitre, nous nous intéressons uniquement aux attaques liées

au service d'authentification. Ainsi, nous ne considérons pas d'autres attaques telles que DoS/DDoS visant à rendre un objet hors service, ou des attaques de suppression de message où l'attaquant supprime sélectivement des messages lors de leurs transmissions.

Le Tableau 9.1 présente les exigences de sécurité et attaques considérées lors de l'évaluation de notre approche (Version 5).

Critères d'évaluation	**Considération**
Identification	✓
Authentification mutuelle	✓
Intégrité des données stockées	✓
Intégrité des données échangées	✓
Disponibilité	✓
Évolutivité	✓
Non répudiation	✓
Confidentialité	✗
Attaque sybil	✓
Attaque d'usurpation d'identité	✓
Attaque de substitution de messages	✓
Attaque de rejeu	✓
Attaque de suppression de messages	✗
Attaque de déni de service sur le mécanisme d'authentification	✓
Attaque de déni de service sur l'objet	✗

TABLEAU 9.1 – Les critères principaux de l'évaluation de *BBtrust*

Remarque : en chiffrant les transactions (ex. en utilisant le mécanisme de cryptographie de courbes elliptiques [109]), le service de confidentialité des données peut être assuré par *BBTrust*. Cependant, nous l'avons pas pris en compte car il n'est pas nécessaire dans notre approche (Version 5).

9.5 Travaux connexes

Récemment, de nombreux chercheurs et développeurs se sont intéressés à l'intégration des *blockchains* dans les systèmes de l'Internet des Objets. Cependant, très peu de travaux s'intéressent à la façon dont les *blockchains* peuvent aider à répondre aux exigences de sécurité de l'IoT. Dans cette section, nous examinons ces travaux, et montrons la rareté des thématiques qui assurent des services de sécurité.

Dans [34] *Christidis et al*, expliquent comment on peut intégrer les *blockchains* et les *smart contracts* dans les systèmes IoT. Ils fournissent en outre une liste concernant les avantages et les limites de l'utilisation de cette intégration. Et enfin, ils montrent que l'utilisation de la *blockchain* facilite le partage des services et des ressources. Cependant les auteurs n'ont pas expliqué comment cette intégration peut aider à améliorer la sécurité de l'IoT. De même, dans [107], *Malviya et al* faisaient une étude sur la possibilité de la sécurisation de l'IoT en se basant sur les fonctionnalités de la *blockchain*, et décrivaient certaines plates-formes IoT reposant sur la *blockchain* dans différents cas d'utilisation.

Huh et al [78], explique aussi une approche pour intégrer les *blockchain* à l'IoT. Cette approche repose sur l'idée de configurer chaque objet par un *smart contract* dédié qui définit ses actions. Cependant, leur travail est encore dans un état très embryonnaire. En

outre, aucun détail sur les cas d'utilisation considérés n'a été fourni. Enfin, ils gardent l'anonymat des objets utilisés, ce qui permet à tout utilisateur, même malveillant, d'utiliser le système. Il est vrai que la force de la *blockchain* repose sur le fait d'accepter toute sorte d'utilisateurs légitimes ou malveillants tant que leurs transactions sont valides, mais dans le cas des systèmes IoT, où l'identification et l'authentification des objets sont essentiels, cette solution n'est pas vraiment adaptées à ces besoins.

Dans [19], *Bahga et al* proposent une plateforme *Blockchain* pour l'Internet des objets industriels, qui améliore les fonctionnalités des plateformes existantes basées sur le *Cloud Computing*. Cependant, pour sécuriser les devices, ils ne s'appuient que sur une paire de clés, générée par le device lui-même, sans aucun contrôle. Par conséquent, tout utilisateur peut exploiter le système.

Ruta et al [141], proposent une nouvelle architecture orientée services (*Service-Oriented Architecture* (SOA)) qui exploite la *blockchain* pour les opérations d'enregistrement, de découverte, de sélection et de paiement. Ces opérations sont implémentées sous forme de *smart contracts* permettant une exécution distribuée et fiable de ces dernières. En revanche, leur approche est basée sur une *blockchain* privée, ce qui limite ses performances.

Hardjono et al [74] proposent *ChainAnchor*, qui représente une méthode de préservation de la vie privée d'un device IoT lors de sa mise en service dans un système de *Cloud Computing*. *ChainAnchor* permet aux propriétaires de devices d'être rémunérés pour vendre leurs données aux fournisseurs de services, et incite les propriétaires et les fournisseurs à partager des données de manière confidentielle. Cependant, le fait que son objectif soit l'anonymat total des objets participants n'est pas adapté à l'IoT. En effet, il existe de nombreux cas où l'identification est nécessaire. Pour finir, *Ouaddah et al* [133] [132] proposent *FairAccess*, qui représente un framework -basé *blockchain*- de contrôle d'accès pour les sytèmes IoT. Les politiques de sécurité de *FairAccess* sont stockées dans une *blockchain* privée. Ainsi, l'authenticité, l'authentification (car les validateurs sont pré-configurés) et les mises à jour des politiques sont toujours garanties. Malgré les avantages qu'offre cette approche, (1) le fait qu'elle dépend d'une *blockchain* privée limite son évolutivité et sa flexibilité, et surtout, (2) elle ne supporte qu'un seul ensemble de politiques de sécurité. Autrement dit, si on a deux systèmes IoT qui ont des règles différentes alors il peuvent pas co-exister dans une même *blockchain FairAccess*.

Pour résumer, le Tableau 9.2 expose et compare les travaux expliqués ci-dessus.

Approche	**Identification/ Authentification**	**Type de *blockchain***	**Implémentation**
Christidis et al [34]	Non spécifié	Non spécifié	✗
Malviya et al [107]	Non spécifié	Non spécifié	✗
Huh et al [78]	✗	Publique	✓
Bahga et al [19]	✗	Publique	Partiellement
Ruta et al [141]	✓	Privée	✓
Hardjono et al [74]	✗	Publique	✗
Ouaddah et al [133] [132]	✓	Privée	✗

TABLEAU 9.2 – Résumé des les travaux connexes

La majorité de ces travaux s'appuient sur le mécanisme de sécurité décrit dans *Bitcoin* (voir section 4.2 page 71) ou *Ethereum* (voir section 4.3 page 72), où chaque objet utilise une paire de clés pour utiliser le système. Ce mécanisme assure un anonymat total, et permet à n'importe quel device -y compris les malveillants- à exploiter le système. Par

conséquent, ces solutions ne sont pas adaptées à beaucoup de systèmes IoT, car la plupart des cas d'utilisation de l'Internet des objets exige une authentification des objets.

Les seuls solutions qui introduisent les services d'identification et authentification, sont celles basées sur les *blockchains* privées. Néanmoins, ces solutions souffrent de la restriction du système utilisé, et elle ne sont pas flexibles ni évolutives (ex. il est difficile d'ajouter un nouveau objet validateur).

De plus, la majorité des travaux décrits sont dans une phase embryonnaire, où seule la description de l'approche est rapidement fournie et où aucune implémentation ou simulation n'a été réalisée.

9.6 Approche (principe de fonctionnement de *BBTrust*)

L'objectif principale de notre approche (Version 5) est de créer des zones virtuelles sécurisées dans un environnement d'une *blockchain* publique. Chaque objet doit communiquer uniquement avec ceux appartenant à sa zone, et considère tout autre objet comme étant malicieux. On appelle ces zones *bulles de confiance* ou en anglais *bubbles of trust* (*BBTrust*). Ainsi, une bulle de confiance est une zone où tout ses membres peuvent se faire confiance. Elle est protégée et inaccessible par les objets non membres. Afin de réaliser un tel système, nous avons utilisé une *blockchain* publique qui implémente des *smart contrats*. La communication dans le système est sous forme de transactions qui doivent être validées par cette *blockchain*. Par exemple, si un objet A envoie un message à un autre objet B, alors (1) A envoie un message à la *blockchain* (*bc*) sous forme d'une transaction, (2) si *bc* authentifie A et que la transaction est correcte, alors *bc* valide la transaction. Et enfin, (3) B peut lire le message.

Dans ce qui suit, nous décrivons le cycle de vie complet d'un objet dans un système IoT sécurisé par *BBTrust*.

9.6.1 Procédure d'initialisation

BBTrust peut être appliquée sur un grand nombre de cas d'utilisation IoT tel que les maisons/villes intelligentes, les usines automatisées, les réseaux véhiculaires, etc. Afin qu'un système soit sécurisé par cette approche, d'abord, il faut qu'il passe par une procédure d'initialisation. Imaginons qu'on a une maison intelligente composée d'une machine à laver intelligente, un micro-onde intelligent, un réveille connecté, un smartphone, etc. Si on veut sécuriser cette maison par *BBTrust*, alors l'administrateur (le propriétaire de la maison) doit effectuer cette procédure d'initialisation. En effet, d'abord il désigne un objet (ex. smartphone) comme étant maître (*Master*) et lui installe une pair de clés de courbe elliptique privée/publique (*privM*/*pubM*). Le maître représente l'autorité de certification du groupe (bulle) d'objets du système. Ensuite, il installe des pairs de clés de courbe elliptique personnalisées privée/publique (*privF*/*pubF*) au reste des objets qu'on appelle désormais disciples (*Followers*). Après, il attribue pour chaque *Follower* une structure de donnée personnalisée que nous avons appelé *ticket*. Ce *ticket* est un certificat léger d'une taille de 64 $octets$ qui permet d'associer l'identité de l'objet à sa paire de clés. Ce certificat contient : (1) un identifiant de la bulle nommé *GroupeID* (ou *GrpID*), (2) un *ObjectID* (ou *ObjID*), qui est l'identifiant du *Follower* dans la bulle, (3) *PubAddr*, qui est l'adresse publique du *Follower*. Cette adresse est obtenue en prenant les 20 premiers $octets$ (20FirstBytes) du hach *Keccak(SHA-3)* [33] de *pubF*. Et (4) une *Signature* ECDSA couvrant le hach Keccack du *GroupID*, *ObjectID* et *PubAddr*. Cette signature est effectuée en utilisant la clé privée du *Master* de la bulle. ECDSA offre de nombreux

avantages par rapport aux algorithmes de signature traditionnels tels que *Rivest Shamir Adleman* (RSA), notamment en ce qui concerne les tailles de clés et les temps de génération/vérification de signature, et est plus adapté aux contextes IoT [97] [41]. La structure du *ticket* est comme suit :

```
========================================
GroupID : XX
ObjectID: YY
PubAddr : @@
========================================
Signature (keccakhash(XX||YY||@@))
========================================
```

9.6.2 Fonctionnement du système

Le schéma de la Figure 9.2 page suivante présente notre approche et toutes les étapes du cycle de vie du système de sécurité.

D'abord, comme illustré dans la phase (A), les objets connectés sont très variés et peuvent appartenir à différents domaines tel que l'industrie, la médecine, l'environnement, l'agriculture, etc. Ensuite, la phase (B) représente la procédure d'initialisation, où un *Master* choisit un *GrpID*, et signe un *ticket* pour chaque *Follower*. Une fois que le groupe est préparé, la phase (C) consiste en la création de la bulle au niveau de la *blockchain*. Pour faire ceci, le *Master* envoie une transaction qui contient son *ObjID* et le *GrpID* choisi. La *blockchain* vérifie l'unicité du *GrpID* du *ObjID* et du *PubAddr*, puis, si la transaction est valide, alors la bulle est créée (ex. bulle F9, bulle 0A, etc).

Remarque : il faut savoir que n'importe quel device n'appartenant à aucune bulle peut être *Master*, et a le droit de créer sa propre bulle.

Ensuite, dans la phase (D), les *Followers* à leurs tours, envoient des transactions afin d'être associés à leurs bulles respectives. Au niveau de la *blockchain*, le *smart contract* vérifie l'existence de la bulle, l'unicité de l'identifiant du *Follower*, et la validité du *ticket* en utilisant la clé publique du *Master* de la bulle. Si l'une de ces conditions n'est pas satisfaite, alors l'objet ne pourra pas être associé à la bulle. L'Algorithme 17 page 124 décrit les paramètres et fonctions de base de *BBTrust*, tandis que l'Algorithme 18 page 124 résume la phase d'association.

Une fois que la $1^{ère}$ transaction (*association request*) du *Follower* est réussie, ce dernier n'aura plus besoin d'utiliser son *ticket* pour s'authentifier (il n'aura plus besoin de le mettre dans une transaction). Pour mieux comprendre, la Figure 9.3 page 125 montre un exemple qui explique en détail le processus d'authentification dans *BBTrust*. Dans cet exemple, nous décrivons un device *Follower* appelé F muni d'un *ticket* signé par un *Master* M. Le *ticket* comprend un *GrpID* = XX, un *ObjID* = YY, une adresse *PubAddr* dérivée de la clé publique du *Follower* (*pubF*), et enfin, une signature de ses données par la clé privée du *Master* (*privM*).

La procédure d'authentification des *Followers* en utilisant *BBTrust* est comme suit :

1. la première transaction d'un *Follower* représente une requête d'association. Cette transaction est signée par (signedWith{}) la clé privée du *Follower* (*privF*), et contient un *ticket* d'authentification ;

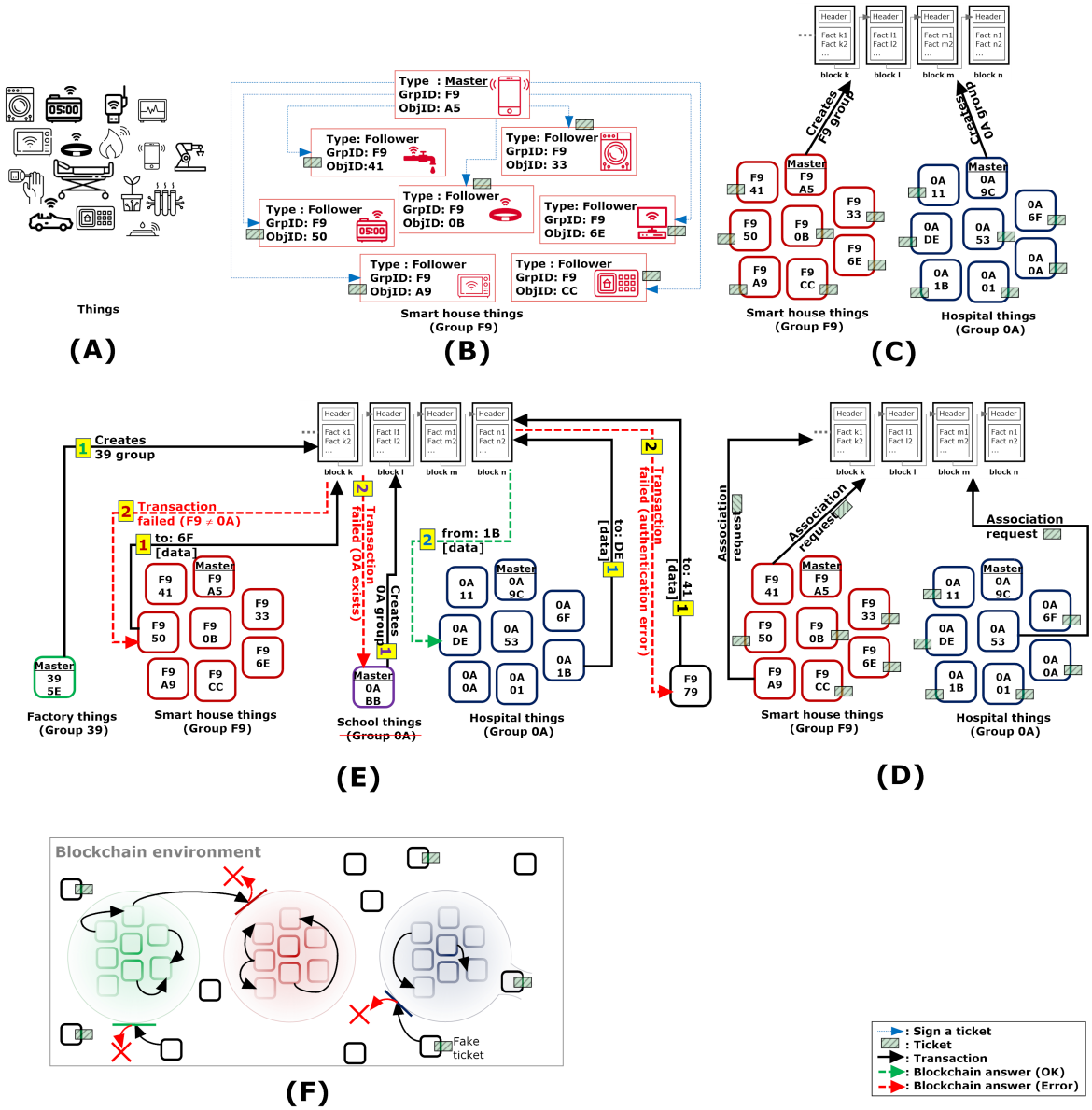


FIGURE 9.2 – Principe de fonctionnement de *BBTrust*

2. en utilisant *pubF*, la *blockchain* vérifie la signature (verifiedWith{}) de la transaction reçue afin de valider son intégrité. Ensuite, elle vérifie le *ticket* à l'aide de *pubM*, car le *ticket* est censé être signé par le *Master* ;
3. si le *ticket* est valide, alors, la *blockchain* sauvegarde une information associant le *GrpID*, l'*ObjID* et la clé *pubF* (ex. XX↔YY↔pubF) ;
4. la transaction n ($n \geq 2$), représente toute autre transaction que celle de l'association. Elle contient (1) la charge utile de la transaction (data), (2) le *GrpID* (ex. XX), (3) l'*ObjID* (ex. YY), et (4) la signature ECDSA des champs précédents en utilisant *privF* ;
5. la *blockchain* vérifie l'intégrité de transaction$_n$ en vérifiant la validité de la signature par *pubF* ;
6. si la signature est valide, alors la *blockchain* vérifie si *pubF* est associé avec les champs *GrpID* et *ObjID* de la transaction ;
7. s'ils sont associés, alors ;
8. l'objet F est authentifié avec succès, sinon il est rejeté.

Algorithm 17: Définition des paramètres et fonctions

parameter :
- bc : Blockchain
- obj : Object
- $sender$: Object
- $receiver$: Object
- const $failed$: State
- define $master$: 0
- define $follower$: 1

Function : `ObjIdExists` (Integer $objId$, Blockchain b)
`// vérifie si l'identifiant de l'objet existe dans la blockchain ou pas`
Function : `GrpIdExists` (Integer $grpId$, Blockchain b)
`// vérifie si l'identifiant du groupe existe dans la blockchain ou pas`
Function : `AddrIdExists` (Integer $objAddr$, Blockchain b)
`// vérifie si l'adresse de l'objet existe dans la blockchain ou pas`
Function : `Error` ()
`// retourne un message d'erreur`

Algorithm 18: Les règles du *smart contract* pour l'association des objets aux bulles

begin
- **if** (`ObjIdExists` $(obj.id, bc) = true$) **then**
  - return `Error` ()
- **if** `AddrIdExists` $(obj.addr, bc)$ **then**
  - return `Error` ()
- **if** $(obj.type = master)$ **then**
  - **if** `GrpIdExists` $(obj.grpId, bc) = true$ **then**
    - return `Error` ()
- **else if** $(obj.type = follower)$ **then**
  - **if** `GrpIdExists` $(obj.grpId, bc) = false$ **then**
    - return `Error` ()
  - **if** $(bc.TicketVerif(obj.ticket) = failed)$ **then**
    - return `Error` ()
- **else**
  - return `Error` ()
- `// Association terminée avec succèss`

La Figure 9.2 page précédente, phase (E), montre comment *BBTrust* effectue le contrôle d'accès des objets et leurs transactions. Par exemple, (1) contrairement au Master 5E qui peut créer le groupe 39, le Master BB ne peut pas créer le groupe 0A, car ce dernier existe déjà. Et (2), contrairement au message accepté échangé de 1B vers DE qui appartient à son groupe 0A, le message échangé de l'objet 50 appartenant au groupe F9 vers l'objet 6F appartenant au groupe 0A est rejeté. L'Algorithme 19 page ci-contre résume les règles de communication dans *BBTrust*.

Enfin, La Figure 9.2 page précédente, phase (F), montre une vue globale du système. Les objets certifiés (ayant des *tickets*) peuvent être ajoutés à leurs groupes à tout moment. Théoriquement, le nombre d'objets par groupe est illimité, car *BBTrust* s'appuie sur une architecture complètement décentralisée. Les objets sans *ticket* ou possédant des faux *tickets* ne peuvent pas être associés aux bulles. Par conséquent, ils ne peuvent pas communiquer avec les objets appartenant à ces bulles (les objets qui sont déjà associés aux

$pubAddr$: Keccak(pubF)
$ticket$: (XX‖YY‖$pubAddr$)_signedWith{privM}
transaction$_1$: ($ticket$)_signedWith{privF}
transaction$_n$: (data‖XX‖YY)_signedWith{privF}
↔: associé avec

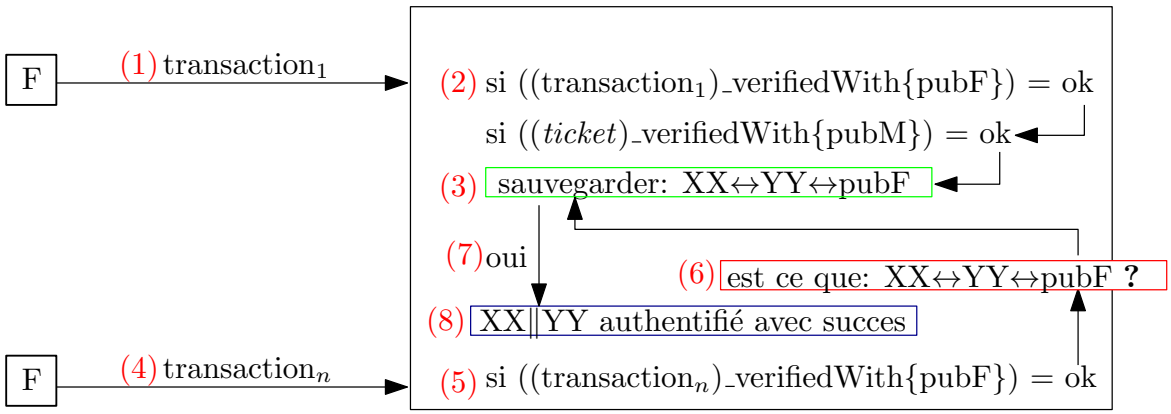


FIGURE 9.3 – Principe de fonctionnement du mécanisme d'authentification de *BBTrust*

Algorithm 19: Les règles du *smart contract* de la communication dans BBTrust

```
begin
    if (ObjIdExists (sender.id, bc) = false
    or (ObjIdExists (receiver.id, bc) = false) then
        return Error ()
    if (sender.grpId ≠ receiver.grpId) then
        return Error ()
    if (bc.SignVerif(sender.msg)) = failed then
        return Error ()

    // Échange de données terminé avec succès
```

bulles). Grâce à la signature des transactions, l'authentification de l'objet et l'intégrité des données échangées sont assurées. Et pour finir, les bulles sont totalement séparées et les objets appartenant à différentes bulles ne peuvent pas envoyer ou recevoir d'informations les uns des autres.

9.6.3 Résumé de la version 5 (BBTrust)

Pour résumer, en fonction du type de l'objet, les règles du *smart contract* sont définies comme suit :

Master : il ne peut créer qu'une seule bulle en utilisant un identifiant de groupe unique, qui n'existe pas dans la *blockchain*. Une fois que la bulle est créée, le rôle spécifique d'un *Master* consiste en la signature des *tickets*. Si un *Master* est hors service, ceci ne perturbe pas le fonctionnement de la bulle (à part l'ajout de nouveaux devices).

Follower : (1) il n'est associé que si sa bulle existe. (2) Il ne peut pas appartenir à plus d'une bulle. (3) Il ne peut pas créer une nouvelle bulle. Et (4) sa première transaction nécessite

un *ticket* signé par la clé privée du *Master* de la bulle.

les deux : (1) l'identifiant de l'objet doit être unique. (2) L'adresse publique et la paire de clés de l'objet doivent être uniques. (3) Les messages doivent être échangés entre des objets appartenant à une même bulle. Et (4) toutes les transactions doivent être signées et vérifiées.

Notre approche est implémentée en utilisant *Ethereum* comme *blockchain*. Ce choix repose sur le fait que : (1) il a le deuxième plus grand *ledger* après *Bitcoin* [123]. (2) Il assure des transactions sécurisées basées sur la cryptographie des courbes elliptiques. (3) Il permet la création des *smart contracts*. (4) Il facilite le développement des applications décentralisées appelées *dApps*. (5) Il est suivit par une grande communauté. Et enfin, les développeurs d'*Ethereum* et sa communauté travaillent sur la régulation et la stabilisation des montants des frais liés à l'utilisation des *smart constracts*[2].

S'appuyer sur une *blockchain* publique apporte beaucoup d'avantages :

— les *blockchains* sont des systèmes décentralisés très résilients, qui sont autonomes pour assurer leur propre fonctionnement (validation des blocs, consensus, etc), ce qui renforce notre approche qui hérite de ces propriétés ;

— les *blockchains* publiques connues, tels que *Bitcoin* et *Ethereum*, sont très robustes et résistants contre la falsification et l'altération des données, ainsi, les informations stockées concernant les objets de confiance sont fiables ;

— une fois que le *smart contract* est déployé, les utilisateurs ne peuvent pas le modifier, car le contrat est envoyé et validé en tant que transaction.

La forte exigence derrière la restriction de la communication entre bulles réside dans le fait que la création d'une nouvelle bulle est donnée à n'importe quelle entité. En effet, si la communication entre bulles était autorisée, si un utilisateur malveillant crée une nouvelle bulle dans le but de communiquer avec une bulle ciblée (ex. une bulle hospitalière), il y parviendra. Sachant que la communication entre les différents cas d'utilisation peut se produire dans les systèmes IoT, *BBTrust* peut évoluer afin de pouvoir supporter la coopération entre un ensemble de bulles ligitmes.

9.7 Évaluation et résultats

9.7.1 Contexte et cas d'utilisation

Comme expliqué ci-dessus, l'un des points fort de notre approche repose sur son adéquation à la majorité des scénarios et secteurs de l'Internet des Objets, tout en assurant une intégration facile de nouveaux objets, services et cas d'utilisation. Dans cette section, nous évaluons notre approche (Version 5) par rapport au temps d'exécution, à la consommation d'énergie, ainsi que le coût financier de certains scénarios. Les cas d'utilisation pris en compte dans l'étude des coûts financiers sont :

Maison intelligente : est une maison équipée d'une technologie permettant aux occupants

2. https://smartereum.com/6777/buterin-expresses-concern-over-stabilizing-ethereum/

de contrôler ou de programmer à distance un ensemble d'appareils électroniques automatisés. Dans cette étude, nous allons prendre le cas d'une maison équipée (1) d'un dispositif ayant la fonction de stocker une liste des courses et de la commander en ligne suivant un certain programme. (2) Un réfrigérateur intelligent, programmé pour garder en permanence certains aliments. Si l'un de ces aliments est consommé, il envoie un message pour l'ajouter à la liste des courses. (3) Une machine à laver intelligente, qui surveille le niveau de lessive. Si le niveau atteint un certain seuil, alors le lave-linge envoie un message pour rajouter la lessive à la liste des courses. (4) Un système d'arrosage à distance, qui peut être déclenché par un smartphone. (5) Un aspirateur robot commandé également par un smartphone.

Gestion des déchets : la gestion des déchets représente un grand défi, notamment dans les grandes villes. L'un des problèmes connus est le routage des camions à ordures [61]. En effet, un camion doit ramasser toutes les poubelles même lorsqu'elles sont vides, ce qui entraîne une perte de carburant et de temps, ainsi qu'une augmentation du CO_2, ce qui a un effet néfaste sur l'environnement. L'opération de ramassage peut être plus coûteuse si on prend en compte les points de ramassage spécifiques pour le plastique, le papier, le verre ou d'autres matériaux spéciaux, où les poubelles sont souterraines et ont une procédure très complexe pour les vider, ce qui dans de nombreux cas provoque des embouteillages. Cela engendre plus de pollution, de perte de temps et d'argent. Heureusement, l'IoT peut apporter plusieurs solutions. Par exemple, on peut installer des capteurs à l'intérieur des poubelles qui envoient des informations sur l'état de remplissage à une application de gestion. En se basant sur ces informations, l'application décide d'ajouter ou non le point de ramassage à la liste des points à parcourir. Ensuite, elle peut utiliser des techniques pour définir le chemin le moins coûteux pour le camion à ordures.

Usine intelligente : l'évolution des systèmes de cyber-physique introduit la $4^{ème}$ générations de l'industrialisation, appelé *industrie 4.0* ou usines intelligentes (smart factories) [98]. D'après [163], une usine intelligente se caractérise par un système multi-agent auto-organisé assisté par des technologies de *Big data* [82]. En d'autres termes, dans une usine intelligente, de nombreuses machines automatisées tel que des bras automatisés, des robots et des véhicules autonomes (sans conducteur), communiquent entre eux et avec le monde extérieur (ex. clients, partenaires, autres sites de production) afin d'avoir une bonne organisation et de fournir une meilleur production.

Radar intelligent : est un radar routier mis en place à distance afin d'éviter la manipulation de son hardware. Sa fonction principale est de mesurer la vitesse des usagers de la route. S'il détecte un dépassement de vitesse, il envoie un message contenant la plaque d'immatriculation et la vitesse de conduite du véhicule au système de gestion.

Migration des objets : enfin, le dernier cas d'utilisation est sur la migration des objets de notre système de réseau de capteur sans fil.

Comme expliqué dans le chapitre 8 page 105 Figure 8.2 page 106, grâce au mécanisme *BCTrust*, une fois qu'un objet (ex. capteur) fasse sa première association (voir chapitre 6 page 87 et chapitre 7 page 91), il peut migrer en toute sécurité d'un groupe vers un autre. Lors de cette migration il y a un échange de données entre les CPANs qui se font confiance mutuellement car ils sont pré-configurés et connus par une *blockchain* privée.

Cependant cette pré-configuration limite considérablement l'évolutivité et la flexibilité des systèmes déployant *BCTrust* et qui peuvent rajouter à tout moment des nouveaux

CPANs. Pour palier à ce problème, on peut intégrer le principe des *bulles de confiances* (*BBTrust*). C'est à dire, un ensemble de CPANs qui se font confiance peuvent former une bulle entre eux et communiquer en toute confiance.

Pour mieux comprendre, dans cette étude, on suppose qu'on a un système de monitoring d'une centrale nucléaire composé de différents services. Chaque service couvre un ensemble d'objets (capteurs et CPANs) appartenant à une même catégorie. Par exemple les objets de type A fournissent des informations sur le taux d'irradiation, les objets de types B contrôle la distribution électrique, les objets de type C concernent les objets liés à la sécurité des bâtiments de la centrale, etc.

Chaque service est géré par un ensemble de CPANs, par exemple $\{A_1, A_2,.., A_l\}$ pour le service A, $\{B_1, B_2,.., B_m\}$ pour le service B, $\{C_1, C_2,.., C_n\}$ pour le service C. Et chaque CPAN est responsable à un groupe de capteurs, par exemple A_1 gère $\{a_{1-1}, a_{1-2},.., a_{1-q}\}$, A_2 gère $\{a_{2-1}, a_{2-2},.., a_{2-r}\}$, etc.

Ces services doivent être virtuellement séparés, autrement dit, un capteur a_{6-3} (géré par le CPAN A_6) ne doit en aucun cas pouvoir migrer à un groupe géré par B_7 ou par C_1 mais il a la possibilité de migrer vers un groupe géré par A_2 car les CPANs de l'ensemble $\{A_1, A_2,.., A_l\}$ appartiennent à une même bulle. Par conséquent, ils peuvent se faire confiance et échanger des données entre eux en toute sécurité.

Dans tous les scénarios décrits, plusieurs messages et informations sensibles sont envoyés sur le réseau. Si un utilisateur malveillant forge, modifie ou rejoue ces messages, les conséquences seront désastreuses. Ainsi, l'authentification et l'intégrité de ces messages sont cruciales.

9.7.2 Cadre d'évaluation

Afin d'évaluer le temps d'exécution et l'énergie consommée de notre approche, nous avons utilisé 3 objets : 2 ordinateurs ayant des capacités similaires et un *Raspberry Pi*. Un ordinateur est désigné en tant que *Master* et les 2 autres sont des *Followers*. Le Tableau 9.3 décrit leurs caractéristiques. Les applications déployées sur ces objets, sont développées en langage *C++*[3]. Comme expliqué dans la section 9.6.3 page 125, nous avons

Type d'objet	CPU			RAM	Système d'exploitation
	architecture	mode d'opération	vitesse max		
Raspberry PI	armv6l	32-bits	700 MHz	450 M*o*	Raspbian 4.9.41
HP laptop	x86_64	64-bits	2600 MHz	8 G*o*	Ubuntu 14.04

TABLEAU 9.3 – Caractéristiques des objets

déployé *Ethereum* comme *blockchain*. L'approche se base sur un *smart contract* que nous avons développé en langage *Solidity* [57]. Pour les interactions entre les objets et la *blockchain*, nous avons créé une interface *C++* qui permet de encoder/décoder les données vers/depuis *Ethereum*. Ces interactions sont effectuées en utilisant *JSON*[4] *RPC*[5] [2].

3. Code source de BBTrust : `https://github.com/MohamedTaharHAMMI/BubblesOfTrust-BBTrust-/tree/master`
4. *JSON* est un format de données textuelles standard.
5. RPC : pour Remote Procedure Call, en Français, appel de procédure à distance.

Dans cette étude, nous ne nous intéressons pas au temps de la communication, car cela dépend de la technologie réseau utilisée. Notre intérêt concerne l'étude de l'impact de notre approche sur les objets, et puisque nous utilisons une *blockchain* publique, où nous n'avons aucun contrôle sur son fonctionnement, toutes nos mesures sont prises au niveau des objets.

Les résultats présentés concernent 100 tests, où nous avons mesuré :

1. t.assoc : le temps nécessaire pour préparer une requête d'association ;
2. t.data : le temps nécessaire pour préparer un message de données ;
3. CPU.assoc : la puissance consommée du processeur (CPU) pour préparer une requête d'association ;
4. CPU.data : la puissance consommée du processeur pour préparer un message de données ;
5. NIC.assoc : la puissance consommée du contrôleur d'interface réseau (*Network Interface Controller* (NIC)) pour établir une association (envoyer une requête + recevoir une réponse) ;
6. NIC.data : la puissance du NIC pour un message de données (envoyer un message + recevoir un reçu).

Sans prendre en considération le coût de la signature des *tickets* pour les nouveaux *Followers*, la consommation du *Master* est toujours ≤ à celle du *Follower* car il n'a pas besoin de joindre un *ticket* dans la première transaction, et il se comporte comme un objet normal pour le reste des opérations. Par conséquent, nous considérons uniquement les mesures des consommations des *Followers*.

Type d'objet	t.assoc (mS)		t.data (mS)		CPU.assoc (mW)		CPU.data (mW)		NIC.assoc (mW)		NIC.data (mW)	
	M	É-T	M	É-T	M	É-T	M	É-T	M	É-T	M	É-T
Ras.PI	28.03	0.045	0.82	0.029	64.16	8.19	16.29	1.10	89.24	14.01	31.22	15.11
Laptop	1.56	0.13	0.04	0.001	9.76	2.04	3.35	0.87	16.14	2.69	12.54	4.51

TABLEAU 9.4 – Moyennes (M) et Écart-Types (É-T) des résultats obtenus.

Temps

D'après les colonnes 2, 3, 4 et 5 du Tableau 9.4, le temps moyen nécessaire pour réaliser une demande d'association est de 1.56 mS pour le Laptop. Le Raspberry Pi a besoin de plus de temps, avec une valeur de 28.03 mS. En revanche, dans les deux cas l'écart-type est petit, ce qui prouve la stabilité de l'approche. Par rapport à ces résultats, la préparation d'un message de données consomme moins de temps : 0.04 mS pour le Laptop et 0.82 mS pour le Raspberry Pi. La raison d'une telle différence repose sur la complexité de la requête d'association par rapport au message de données. La valeur de l'écart-type est également petite.

Consommation de puissance

Les colonnes 6, 7, 8 et 9 du Tableau 9.4 montrent que le Raspberry Pi consomme 64.16 mW pour établir une requête d'association, tandis que le Laptop consomme uniquement 9.76 mW. Concernant le message de données, le Raspberry Pi nécessite 16.29 mW comparé à 3.35 mW dans le cas du Laptop.

Comme expliqué ci-dessus, ces différences sont dues à la complexité de la requête d'association par rapport à la préparation d'un simple message de données.

Enfin, d'après les colonnes 10, 11, 12 et 13, le Raspberry Pi nécessite 89.24 mW pour exécuter une requête d'association, alors que le laptop n'a besoin que de 16,14 mW. Pour un message de données, le Raspberry Pi consomme 31.22 mW et 12.54 mW nécessaires pour le Laptop. Cette grande différence entre les consommations du CPU et de la NIC ne fait que confirmer que les communications réseau sont les opérations les plus coûteuses pour un système.

Les Figures 9.4 et 9.5 page suivante décrivent l'impact de l'opération de préparation des messages de données sur le CPU et la consommation d'énergie de la mémoire RAM dynamique (*Dynamic Random Access Memory* (DRAM)) pour les objets testés. Ces 2 figures expriment 3 phases du fonctionnement du système : (1) une phase de repos ; (2) l'exécution d'une boucle qui envoie 100 messages avec une pause de 100 mS entre chaque message ; et (3) le retour à la phase de repos. Les mesures sont réalisées à l'aide de *RAPL*[6]. Ce dernier représente un outil de mesure de l'activité totale du CPU et de la DRAM qui ne permet pas d'isoler les mesures par un processus choisi.

La Figure 9.4 décrit les résultats du Laptop où la boucle est exécutée à la $12^{ème}$ secondes. Quant à la Figure 9.5 page suivante, elle décrit les résultats du Raspberry Pi où la boucle est exécutée à la $15^{ème}$ secondes.

Dans les deux cas, on peut constater que l'impact de la boucle de traitement de messages est vraiment négligeable. Les autres pics existants sont liés à l'activité du système d'exploitation.

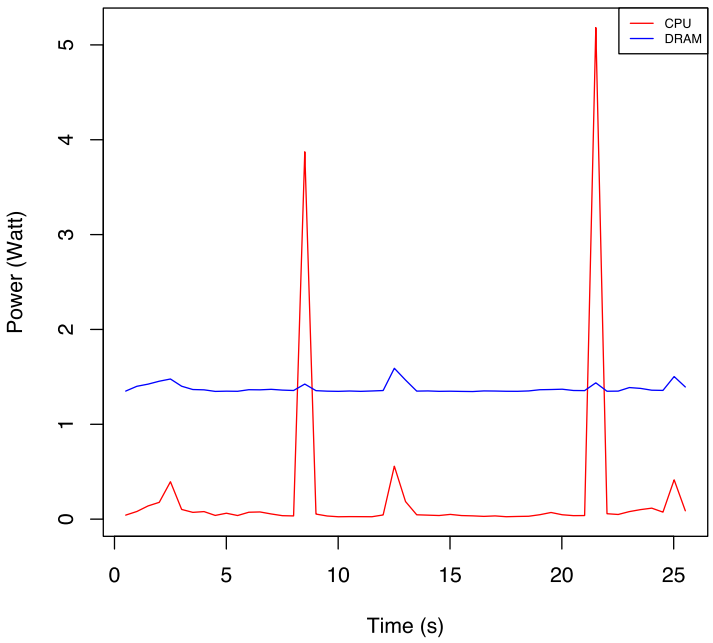


FIGURE 9.4 – L'impact du traitement des messages de données sur le Laptop

Évaluation des exigences de sécurité

Avant d'entamer l'étude sur le coût financier de *BBTrust*, dans ce qui suit, nous allons

6. RAPL : https ://github.com/kentcz/rapl-tools

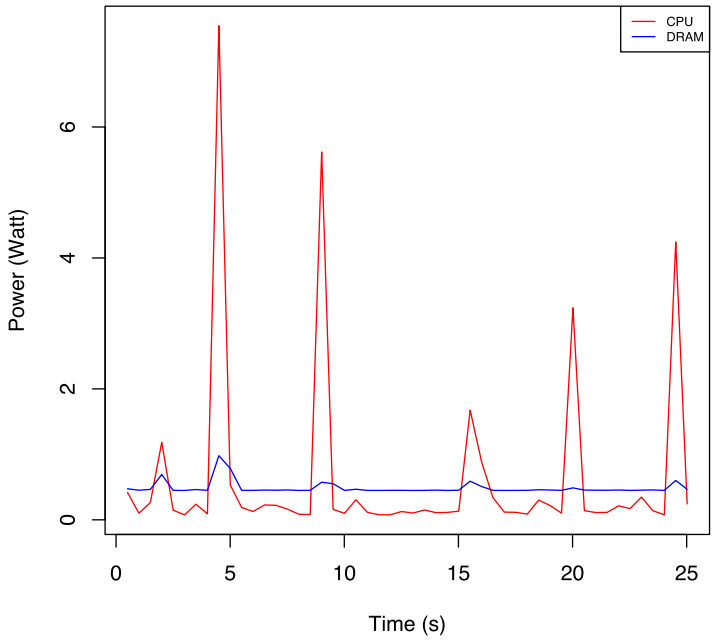


FIGURE 9.5 – L'impact du traitement des messages de données sur le Raspberry Pi

d'abord expliquer comment notre approche répond aux différentes exigences de sécurité présentées dans la section 9.3 page 116 et comment elle peut protéger les systèmes qui la déploient contre les attaques présentées dans la section 9.4 page 117.

Authentification mutuelle et intégrité des messages : hormis le créateur de la bulle (Master), chaque objet utilise un *ticket* lors de sa première transaction, qui est, comme décrit dans la section 9.6.1 page 121, l'équivalent d'un certificat. Ces *tickets* sont livrés uniquement aux objets légitimes pendant la phase d'initialisation. Tous les messages échangés sont signés par l'algorithme *ECDSA* en utilisant les clés privées des expéditeurs. Les signatures sont vérifiées par les clés publiques correspondantes. Ces dernières font partie des constituants des *tickets* (l'adresse publique du *ticket* et dérivée à partir de la clé publique). Ainsi, les signatures assurent à la fois l'authentification des objets et l'intégrité des messages. De plus, comme expliqué dans la section 4.1 page 69, la *blockchain* est considérée comme étant fiable, donc toutes les informations y parvenant sont correctes.

Identification : chaque objet possède une identité (*ObjID* associée à un *GrpID* et à son adresse publique (générée à partir de sa clé publique)). La fiabilité de cette identité est assurée par la signature du *Master* dans le *ticket*. Chaque message de cet objet est signé par sa clé privée associée à son identité. Par conséquent, le système peut facilement l'identifier.

Non répudiation : le fait que chaque message doit être signé par une clé privée qui n'est connue que par son objet propriétaire, empêche ce dernier de nier avoir effectuer une ou plusieurs transactions.

Évolutivité : Sachant que les réseaux pair à pair représentent l'une des meilleures solu-

tions pour avoir une évolutivité à grande échelle [105], et que les *blockchains* publiques reposent sur ce type de réseau, prouve l'évolutivité et les bonnes performances réseau de *BBTrust*.

Protection contre les attaques Sybil : dans notre conception, chaque objet ne peut avoir qu'une seule identité et chaque identité ne peut avoir qu'une seule paire de clés. Chaque message de communication doit être signé par la clé privée associée à cette identité. De plus, toutes les identités doivent être approuvées par le système, ainsi un attaquant ne peut pas utiliser de fausses identités.

Protection contre les attaques d'usurpation : comme pour le cas de l'authentification ou la protection contre l'attaque *Sybil*, un attaquant ne peut pas usurper l'identité d'un autre objet, puisqu'il a besoin de sa clé privée.

Protection contre la substitution de messages : puisque tous les messages sont signés, si un attaquant en modifie ou en substitue un, il doit le signer avec une clé privée valide. Cependant, seuls les objets légitimes sont munis de *tickets* (paires de clés valides) lors de la phase d'initialisation.

Protection contre les attaques de rejeu : tous les messages sont sous forme de transactions. Chaque transaction est horodatée et nécessite une phase de consensus pour qu'elle soit valide. Par conséquent, un attaquant ne peut pas rejouer les messages car le mécanisme de consensus les rejettera. Pour plus d'information, [94] décrit comment les *blockchains* sont résistants contre les attaques de rejeu.

Protection contre les attaques Dos/DDos : la décentralisation totale de l'architecture de la *blockchain* la rend robuste contre les attaques DoS/DDoS. Ceci est due à la duplication et la répartition des services sur différents objets du réseau. C'est-à-dire, même si un attaquant parvient à bloquer un objet, il ne peut pas bloquer tous les autres objets. En outre, les transactions sont coûteuses, ce qui décourage un attaquant de dépenser de l'argent pour envoyer un grand nombre de transactions.

Coût financier

Les transactions d'une *blockchain* publique sont payantes, par conséquent une étude sur le coût financier de notre approche appliquée à différents cas d'usage est nécessaire. Car il faut savoir que si le coût d'un protocole de sécurité est plus grand que les dégâts potentiels d'une attaque informatique, le protocole est inacceptable. De ce fait, nous allons reprendre les cas d'utilisations présentés dans la section 9.7.1 page 126.

Maison intelligente : pour le scénario de la maison intelligente, on suppose que (1) la machine à laver intelligente envoie 1 requête par semaine pour ajouter la lessive sur la liste des courses, ce qui nécessite 1 *transaction* par semaine. Cette transaction implique 1 *call* afin de récupérer l'information. (2) Le frigo intelligent envoie deux commandes par semaine. (3) L'application de liste des courses fait deux commandes par semaine. (4) Le système d'arrosage intelligent est déclenché deux fois par semaine. Enfin (5) L'aspirateur robot est utilisé 3 fois par semaine. Ainsi, 40 *transactions* et 40 *calls* sont lancés par mois.

Gestion des déchets : comme expliqué précédemment, le ramassage des poubelles n'est pas une tache simple, notamment quand il s'agit des poubelles souterraines qui néces-

sitent une procédure compliquée pour être vidées. Le fait de connaître d'avance l'état des poubelles (pleines ou pas) permet d'économiser du temps et de diminuer l'émission du CO_2. En revanche, il est très difficile d'estimer les bénéfices financiers que notre approche peut apporter, car cela dépend du nombre de points de ramassage que le camion doit traverser, de leur localisation et de nombreux autres paramètres (par exemple, la carte de la ville et la géographie). De ce fait, si le coût des transactions (envoyées par les poubelles) dépasse le prix du carburant, alors dans ce cas notre solution ne peut pas être applicable. Dans ce qui suit, ce scénario ne sera pas traité.

Usine intelligente : on prend le cas d'une usine intelligente qui fonctionne 12 heures par jour. Elle possède 30 bras robotisés (appartenant à 2 types différents) et 10 véhicules autonomes. Chaque produit est traité par deux bras robotiques de différents types. Une fois que le premier bras termine le traitement du produit, il le passe au second et lui envoie un message pour le déclencher. Une fois que le deuxième traitement est terminé, le deuxième bras envoie un message au véhicule autonome afin de transporter le produit final. Enfin, une notification est envoyé par le véhicule à une application de gestion une fois que son travail est terminé. Chaque produit nécessite 30 minutes pour être fabriqué (15 minutes pour chaque machine). Par conséquent, chaque machine autonome envoie 1 *transaction* toutes les 15 minutes et déclenche 1 *call*. Ainsi, 1920 transactions et 1920 *calls* sont utilisés par jour (45360 transactions/*call* par mois).

Radar intelligent : pour ce cas d'usage, prenons le cas de la ville de Paris. En 2015, les radars ont détecté 703957 violations de vitesse [7]. En prenons cette exemple on obtient 58663.08 violations de vitesse par mois.

Migration des objets : pour ce dernier scénario, on prend un réseau très dynamique couvrant 5 groupes de 10 objets chacun (9 devices gérés par un CPAN). On suppose que (1) pour chaque mois, on rajoute 1 nouveau device à son cluster de base (5 nouveaux devices par mois pour tous le réseau), et que (2) chaque jour, une opération de migration s'effectue au niveau de chaque groupe (150 opérations de migration par mois). Par conséquent, 5 *transactions* par mois pour les nouvelles opérations d'association, et 150 *transactions* avec 150 *calls* sont effectués par mois pour les opérations de migration.

Le Tableau 9.5 page suivante présente des estimations sur le coût financier des scénarios étudiés concernant le nombre de *transactions* et *calls* pour un mois. Les valeur du Tableau son obtenues en utilisant l'Algorithme 20 page suivante. Pour cette évaluation, nous avons utilisé la crypto-monnaie *Ethereum Classic* (ETC) [8]. En revanche, d'autres crypto-monnaies, plus ou moins chères que l'ETC peuvent être utilisées.

D'après les résultats du Tableau 9.5 page suivante, on peut constater que le coût de *BBTrust* est plus qu'acceptable pour les scénarios de la maison intelligente et de la migration des objets. En ce qui concerne le cas de l'usine intelligente et le radar intelligent, comparé à l'argent que ces derniers rapportent, le coût dépensé par notre approche est également acceptable.

Limites de *BBTrust*

Notre approche souffre de deux principaux problèmes. Le premier est due par le fait que

7. http ://www.lefigaro.fr/actualite-france/2016/02/20/01016-20160220ARTFIG00011-les-50-radars-qui-flashent-le-plus-en-france.php

8. La valeur du ETC pendant la rédaction du manuscrit (16 Avril 2018) est de 1 ETC = 13.02 EUR

Algorithm 20: Bubble of trust cost calculator

```
const _trans_cost    = 500          // gas
const _call_cost     = 20           // gas
const _gas_in_eth    = 0.00001      // ETH
const _eth_in_euro = 13.02          // Euro

Function Cost (double trans_number, double call_number)
begin
    return ((trans_number × _trans_cost) + (call_number × _call_cost)) × _gas_in_eth ×
    _eth_in_euro;
```

Scénario	Objet	# Trans	# Calls	Prix (Gas)	Prix (ETC)	Prix (Euro)
Maison Intelligente	liste des courses	8	8	4160	0.0416	0.5416
	frigo	8	8	4160	0.0416	0.5416
	machine à laver	4	4	2080	0.0208	0.2708
	système d'arrosage	8	8	4160	0.0416	0.572
	aspirateur robot	12	12	6240	0.0624	0.8125
	Total	**40**	**40**	**20800**	**0.208**	**2.7385**
Usine Intelligente	bras robotisés	43200	43200	22464000	224.64	2924.8128
	véhicules autonomes	2160	2160	1123200	11.232	146.2406
	Total	**45360**	**45360**	**23587200**	**235.872**	**3071.0534**
Radar Intelligent	**radar**	**58663.08**	**58663.08**	**30504801.6**	**305.04802**	**3971.7252**
Migration	$1^{ère}$ association	5	0	2500	0.025	0.3255
	migration	150	150	78000	0.78	10.1556
	Total	**155**	**150**	**80500**	**0.805**	**10.4811**

TABLEAU 9.5 – L'estimation du coût financier de *BBTrust* par mois

BBTrust est basée sur une *blockchain* publique. En effet, selon le protocole de consensus utilisé, les transactions (blocs) sont validées chacune (chacun) après une certaine période de temps définie (ex. 14 secondes dans le cas d'*Ethereum*). Ainsi, les transactions (messages) envoyées par les devices ne seront validées qu'après cette période. Par conséquent les systèmes temps réel ne peuvent pas déployer *BBTrust*.

Remarque : ce problème n'existe pas dans le cas des *blockchains* privées.

Le deuxième problème est que dans certains scénarios, deux différents services (ex. le service de santé et le service scolaire) peuvent avoir un échange de certaines informations. Cependant avec la version actuelle de *BBTrust*, l'échange sécurisé d'informations entre deux bulles légitimes n'est pas encore possible.

9.8 Conclusion

Comme expliqué dans la section 9.2 page 115, *BBTrust* peut être utilisée comme un protocole complet qui assure les services de sécurité de base, notamment l'authentification mutuelle, la confidentialité et l'intégrité des données. Mais aussi être déployée comme un mécanisme complémentaire pour d'autres protocoles tel que le notre qui est passé par les versions 1,2,3 et 4.

Notre approche permet d'avoir un système d'authentification totalement décentralisé qui permet d'assurer une authentification efficace des objets. La disponibilité, la traçabilité et l'intégrité des données sont également assurées par ce mécanisme. Enfin, il faut savoir que même si le service de confidentialité des données n'a pas été abordé dans ce chapitre, ce dernier peut être intégré (ex. en chiffrant les transactions avec un mécanisme de cryptographie de courbes elliptiques [109]).

Dans ce chapitre, on a montré comment cette approche peut répondre aux exigences de sécurité tout en gardant une bonne performance du système et du réseau.

Les résultats des différentes évaluations en terme de temps, d'énergie, et en coût financier montre les performances et l'adaptabilité de notre approche pour des cas d'utilisations divers, utilisant des architectures, des objets, des technologies et des protocoles différents.

Quatrième partie

Conclusion et perspectives

L'Internet des objets représente une idéologie globale qui supprime les frontières entre le monde physique et le monde virtuel, et qui couvre la plupart des systèmes informatiques et technologies d'information. En effet, l'IoT regroupe plusieurs domaines tel que les réseaux de capteurs sans fils, les villes intelligentes, les véhicules connectés, les systèmes de contrôle commande, etc. De ce fait, un système IoT est un système hétérogène utilisant différentes technologies, déployées sur des architectures et des plateformes différentes et implémentées sur des matériaux informatiques (hardware) très variés.

Généralement, il utilise des technologies de communications sans fil afin de connecter des objets intelligents et autonomes. Ces objets ont la capacité de rassembler, analyser, traiter, générer et échanger des informations afin de fournir des services avancés.

Cependant, les problèmes de sécurité informatique ralentissent considérablement l'évolution et le déploiement rapide de cette technologie de pointe. De plus, cette dernière, ne peut pas utiliser les protocoles de sécurité classiques (ex. TLS [45]) et les solutions existante car la plupart ne permettent pas d'assurer des bonnes performances ou bien ils ne sont pas adaptées aux capacités des objets qui sont généralement très limités terme de stockage de calcul et en énergie.

Pour palier à ces problèmes, pendant mes années de thèse, j'ai crée progressivement un protocole de sécurité léger et robuste qui est conçu spécialement pour les systèmes IoT et adapté à leurs performances. Notre approche est passée par plusieurs versions.

Au début nous avons crée un protocole basique (publié dans [71]) qui permet d'assurer le service d'authentification des devices afin de protéger les réseaux contre les attaques d'usurpation d'identité. Nous avons crée également un mécanisme de gestion de clés appelé *"Personnalisation"* de clés, qui représente une méthode sécurisée et flexible de distribution des clés pré-partagées, qui protège les objets contre les attaques d'usurpation interne. Enfin, cette version assure aussi le service d'intégrité des données afin de les conserver contre toutes altérations. Le mécanisme d'authentification utilise une méthode basée sur les mots de passe à usage unique et le principe de challenge/réponse. Ces deux algorithmes sont robustes et très légers et nous permettent d'avoir un mécanisme d'authentification efficace et adapté aux objets limités. Quant aux mécanismes de *"Personnalisation"* et d'intégrité de données, nous avons utilisé l'algorithme HMAC-SHA256 qui est une fonction de hachage réputée par sa résistance contre les différents types d'attaques de cryptanalyse.

Ensuite, nous avons amélioré le service d'authentification simple, en une authentification mutuelle entre le device et le CPAN gérant le réseau (publié dans [72]) afin de pouvoir garantir la légitimité du réseau utilisé. Cette version améliore le mécanisme de génération de clé du mode *unicast* et rajoute un autre mécanisme, appelé *hidden broadcast key*, pour l'échange de clé de *broadcast*.

Après, dans une troisième version du protocole (publiée dans [69] et dans [68]) nous avons rajouté le service de confidentialité. Ce dernier est basé sur le standard AES. Ce dernier représente l'algorithme de chiffrement de référence grâce à son optimalité, rapidité, robustesse et légèreté. Comme mode d'opération, nous avons opté pour le GCM et le CCM (au choix). Ces modes permettent d'assurer à la fois les services de confidentialité et d'intégrité de données. Ainsi, nous avons enlevé le HMAC-SHA256, ce qui optimise encore plus notre protocole.

La version 4 du protocole (publiée dans [67]) améliore encore le service d'authentification et fournit un mécanisme d'authentification décentralisé basé sur les *blockchains* privées. Ce mécanisme permet la migration sécurisée des objets mobiles qui se déplace d'un réseau vers un autre.

Enfin, afin d'avoir plus de flexibilité et d'évolutivité du système, nous avons finalisé

notre approche par un mécanisme, appelé *Bubbles of Trust* ou *BBTrust* (publié dans [70]), qui permet d'avoir un système d'authentification totalement décentralisé et applicable à la plupart des cas d'utilisation IoT. *BBTrust* est basé sur une *blockchain* publique, ce qui lui permet d'hériter de sa puissance, notamment l'intégrité des données stockées, la décentralisation complète du système, la non répudiation des opérations, l'évolutivité du réseau, la disponibilité et la traçabilité des informations.

Pour l'évaluation de notre approche, d'abord nous avons réalisé des modèles de menace qui sont des procédures nous permettant d'analyser un protocole de sécurité afin d'identifier ses objectifs et ses vulnérabilités. Pour faire cela, nous avons opté pour le modèle de *Dolev-Yao* [48] qui représente le modèle le plus utilisé.

Ensuite, nous avons réalisé une validation formelle en utilisant *Scyther* [39]. Ce dernier, est un outil/langage qui permet de vérifier la robustesse et la sûreté des protocoles de sécurité.

Une fois que les résultats de l'évaluation formelle sont favorables, nous avons implémenté, au fur et à mesure, notre protocole en langage C pour les versions 1,2,3 et 4, puis en langage C++ pour la version 5. Nous avons déployé le code source du protocole de sécurité des 3 premières versions sur une véritable architecture de réseau de capteur sans fil . Tandis que pour la version 4 et 5, nous avons réalisé une implémentation réelle mais qui n'a été déployée et testée que sur des Raspberry pi et des ordinateurs afin d'avoir une preuve du concept.

Dans ce travail, nous avons proposé un protocole de sécurité léger et efficace, qui peut être déployer sur différentes architecture et technologies IoT et permet de protéger leurs systèmes et leurs données.

Bubbles of trust est l'un des rares systèmes de sécurité décentralisés qui permet d'assurer principalement l'identification et l'authentification des objets ainsi que l'intégrité des données échangées. De ce fait, BBTrust représente un noyau par lequel des applications d'internet des objets et de technologie d'information peuvent être protégées.

Notre travail nécessite des tests sur un grand réseau d'objets afin de mieux évaluer ses performances. Il demeure exploitable et pourrait être amélioré (ex. rajouter le service de confidentialité).

Enfin, afin d'avoir une version mature de *Bubbles of trust* on doit créer un mécanisme qui permet l'échange sécurisé des informations entre deux ou plusieurs bulles légitimes.

Chapitre 10

Références

[1] Announcing the Advanced Encryption Standard (AES) . Technical report, November 2001. 91

[2] Fast Ethereum RPC client for testing and development . *Online. Test RPC. https ://github.com/ethereumjs/testrpc* , 2015. 109, 128

[3] Bitcoin Developer Guide. Technical report, Bitcoin, 2017. 71

[4] Ethereum Development Tutorial. Technical report, Ethereum, 2017. 72

[5] 3GPP. 3GPP Technical Specification Group Services and System Aspects, 3GPP System Architecture Evolution (SAE). In *TS33.401, Security architecture (Release 9)*, volume 9.3.1. 3rd Generation Partnership Project, 2010. 43

[6] 3GPP. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 and 128-EIA3. In *128-EEA3 and 128-EIA3 Specification*. 3rd Generation Partnership Project, 2011. 43, 44

[7] ETSI 3GPP LTE. *Universal Mobile Telecommunications System (UMTS) LTE*, Jan 2009. 43

[8] Bernard Aboba, Larry Blunk, John Vollbrecht, James Carlson, and Henrik Levkowetz. Extensible authentication protocol (EAP). Technical report, 2004. 48, 49

[9] Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Ersin Uzun, and Lixia Zhang. Interest flooding attack and countermeasures in Named Data Networking. In *IFIP Networking Conference, 2013*, pages 1–9. IEEE, 2013. 33

[10] Khaldoun Al Agha, Gerard Chalhoub, Alexandre Guitton, Erwan Livolant, Saoucene Mahfoudh, Pascale Minet, Michel Misson, Joseph Rahme, Thierry Val, and Adrien Van Den Bossche. Cross-layering in an Industrial Wireless Sensor Network : Case Study of OCARI. *JNW*, 4(6) :411–420, 2009. 62

[11] Muhammad Alam, Joaquim Ferreira, and José Fonseca. Introduction to intelligent transportation systems. In *Intelligent Transportation Systems*, pages 1–17. Springer, 2016. 118

[12] Wi-Fi Alliance. The State of Wi-Fi Security : Wi-Fi CERTIFIED WPA2 Delivers Advanced Security to Homes, Enterprises and Mobile Devices. Technical report, 2012. 54

[13] Wi-Fi Alliance. Discover Wi-Fi, 2015. 51

[14] Jeffrey G Andrews, Stefano Buzzi, Wan Choi, Stephen V Hanly, Angel Lozano, Anthony CK Soong, and Jianzhong Charlie Zhang. What will 5G be ? *IEEE Journal on selected areas in communications*, 32(6) :1065–1082, 2014. 41

[15] Paul Arana. Benefits and vulnerabilities of Wi-Fi protected access 2 (WPA2). *INFS 612*, pages 1–6, 2006. 56, 57

[16] Jean-Luc Archimbaud, Catherine Grenet, and Marie-Claude Quidoz. Recommandations de securite destinees aux administrateurs systemes et reseaux du CNRS pour l'installation de reseaux locaux sans fil WiFi. page 10, February 2011. 56

[17] Atmel. ZigBee PRO Pack and Analysis with Sniffer. In *Application note*, sep 2013. 13, 89

[18] available from : https ://www.sigfox.com/en/sigfox-iot-technology overview. Sigfox overview. Technical report, 2017. 35

[19] Arshdeep Bahga and Vijay K Madisetti. Blockchain platform for industrial internet of things. *Journal of Software Engineering and Applications*, 9(10) :533, 2016. 120

[20] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3) :362–399, 2000. 94

[21] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of Activity : Extending Bitcoin's Proof of Work via Proof of Stake. *ACM SIGMETRICS Performance Evaluation Review*, 42(3) :34–37, 2014. 70

[22] G Berzins, R Phillips, J Singh, and P Wood. Inmarsat-Worldwide mobile satellite services on seas, in air and on land. In *Malaga International Astronautical Federation Congress*, 1989. 45

[23] A. N. Bikos and N. Sklavos. LTE/SAE Security Issues on 4G Wireless Networks. *IEEE Security Privacy*, 11(2) :55–62, March 2013. 41

[24] Joseph Birr-Pixton. Cifra. *https ://github.com/ctz/cifra*, 2018. 112

[25] Andrea Bittau, Mark Handley, and Joshua Lackey. The final nail in WEP's coffin. pages 15–pp, 2006. 53

[26] Dieter Bong and Andreas Philipp. Securing the smart grid with hardware security modules. In *ISSE 2012 Securing Electronic Business Processes*, pages 128–136. Springer, 2012. 117

[27] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications : the insecurity of 802.11. Technical report, 2001. 53

[28] Better Buys. Estimating password cracking times. Technical report. 97

[29] Ed Callaway, Paul Gorday, Lance Hester, Jose A Gutierrez, Marco Naeve, Bob Heile, and Venkat Bahl. Home networking with IEEE 802.15. 4 : a developing standard for low-rate wireless personal area networks. *IEEE Communications magazine*, 40(8) :70–77, 2002. 57

[30] Nancy Cam-Winget, Russ Housley, David Wagner, and Jesse Walker. Security flaws in 802.11 data link protocols. *Communications of the ACM*, 46(5) :35–39, 2003. 54

[31] Nancy Cam-Winget, Russ Housley, David Wagner, and Jesse Walker. Security flaws in 802.11 data link protocols. *Communications of the ACM*, 46(5) :35–39, 2003. 55

[32] Miguel Castro, Barbara Liskov, et al. Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999. 73

[33] Shu-jen Chang, Ray Perlner, William E Burr, Meltem Sönmez Turan, John M Kelsey, Souradyuti Paul, and Lawrence E Bassham. Third-round report of the sha-3 cryptographic hash algorithm competition. *NIST Interagency Report*, 7896, 2012. 121

[34] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4 :2292–2303, 2016. 119, 120

[35] T Clancy and W Arbaugh. Extensible authentication protocol (EAP) password authenticated exchange. Technical report, 2006. 49

[36] Thomas Clausen and Philippe Jacquet. Optimized link state routing protocol (OLSR). 2003. 63

[37] Ethereum community. Ethereum Homestead Documentation. *Online. http ://www.ethdocs.org/en/latest/index.html*, 2016. 73

[38] Ethereum community. Ethash. *Etherium, wiki*, apr 2017. 73

[39] Cas Cremers. Scyther. *Draft*, February 2014. 98, 99, 140

[40] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology : Beyond bitcoin. *Applied Innovation*, 2 :6–10, 2016. 71

[41] Erik De Win, Serge Mister, Bart Preneel, and Michael Wiener. On the performance of signature schemes based on elliptic curves. In *International Algorithmic Number Theory Symposium*, pages 252–266. Springer, 1998. 122

[42] L Peter Deutsch. GZIP file format specification version 4.3. 1996. 55

[43] ARM developer. *Cortex-M3 Technical Reference Manual*, 2010. 102

[44] Tim Dierks. The transport layer security (TLS) protocol version 1.2. 2008. 87

[45] Tim Dierks and Eric Rescorla. Rfc 5246 : The transport layer security (tls) protocol. *The Internet Engineering Task Force*, 2008. 139

[46] Fred J Dietrich. The Globalstar satellite cellular communication system : design and status. In *Wescon/97. Conference Proceedings*, pages 180–186. IEEE, 1997. 45

[47] Robert C Dixon. *Spread spectrum systems : with commercial applications*, volume 994. Wiley New York, 1994. 36

[48] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2) :198–208, 1983. 96, 117, 140

[49] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for IoT security and privacy : The case study of a smart home. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 618–623. IEEE, 2017. 69

[50] Angela Doufexi, Simon Armour, Michael Butler, Andrew Nix, David Bull, Joseph McGeehan, and Peter Karlsson. A comparison of the hiperlan/2 and ieee 802.11 a wireless lan standards. *IEEE Communications magazine*, 40(5) :172–180, 2002. 36

[51] Morris J Dworkin. SP 800-38C. Recommendation for block cipher modes of operation : The CCM mode for authentication and confidentiality. 2004. 15, 94, 97, 101

[52] Morris J. Dworkin. SP 800-38D. Recommendation for Block Cipher Modes of Operation : Galois/Counter Mode (GCM) and GMAC. Technical report, Gaithersburg, MD, United States, 2007. 15, 94, 97, 101

[53] D Eastlake 3rd and Paul Jones. US secure hash algorithm 1 (SHA1). Technical report, 2001. 61

[54] Burton I Edelson and Louis Pollack. Satellite communications. *Science*, 195 :1125–1133, 1977. 45

[55] Sinem Coleri Ergen. ZigBee/IEEE 802.15. 4 Summary. *UC Berkeley, September*, 10 :17, 2004. 60, 62

[56] Scott Fluhrer, Itsik Mantin, Adi Shamir, et al. Weaknesses in the key scheduling algorithm of RC4. In *Selected areas in cryptography*, volume 2259, pages 1–24. Springer, 2001. 53

[57] Ethereum foundation. Solidity documentation. *Online manual. https ://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html*, 2017. 17, 106, 109, 128

[58] The Wireshark Foundation. Wireshark 2.0.3 and 1.12.11 Released. April 22, 2016. https://www.wireshark.org/news/20160422.html accessed online on 2016-06-28. 84, 89

[59] Alan Freier, Philip Karlton, and Paul Kocher. The secure sockets layer (SSL) protocol version 3.0. 2011. 45

[60] Thomas Fuhr, Henri Gilbert, Jean-René Reinhard, and Marion Videau. Analysis of the Initial and Modified Versions of the Candidate 3GPP Integrity Algorithm 128-EIA3. In *Selected areas in cryptography,TS 33.401*, volume 7118, pages 230–242. Springer, 2011. 43, 65

[61] Radek Fujdiak, Pavel Masek, Petr Mlynek, Jiri Misurec, and Ekaterina Olshannikova. Using genetic algorithm for advanced municipal waste collection in smart city. In *Communication Systems, Networks and Digital Signal Processing (CSNDSP), 2016 10th International Symposium on*, pages 1–6. IEEE, 2016. 127

[62] Annabelle Gawer and Michael A Cusumano. *Platform leadership : How Intel, Microsoft, and Cisco drive industry innovation*. Harvard Business School Press Boston, 2002. 47

[63] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. Decentralization in Bitcoin and Ethereum Networks. *arXiv preprint arXiv :1801.03998*, 2018. 73

[64] Brian Gladman. A specification for Rijndael, the AES algorithm. *at fp. gladman. plus. com/cryptography-technology/rijndael/aes. spec*, 311 :18–19, 2001. 38, 39

[65] S Gopalakrishnan. A survey of wireless network security. *International Journal of Computer Science and Mobile Computing*, 3(1) :53–68, 2014. 35, 36

[66] Neil Haller, Craig Metz, Phil Nesser, and Mike Straw. A one-time password system. Technical report, 1998. 10, 79

[67] M. T. Hammi, P. Bellot, and A. Serhrouchni. Bctrust : A decentralized authentication blockchain-based mechanism. In *IEEE Wireless Communications and Networking Conference (WCNC)*, page 6, Barcelone, Espagne, April 2018. 21, 139

[68] M. T. Hammi, E. Livolant, P. Bellot, P. Minet, and A. Serhrouchni. A safe communication protocol for iot devices. *submitted in Annals of Telecommunications*, page 15, 2018. 21, 139

[69] Mohamed Hammi, Erwan Livolant, Patrick Bellot, Ahmed Serhrouchni, and Pascale Minet. A lightweight iot security protocol. In *1st Cyber Security in Networking Conference (CSNet2017)*, 2017. 21, 139

[70] Mohamed Tahar Hammi, Badis Hammi, Patrick Bellot, and Ahmed Serhrouchni. Bubbles of trust : a decentralized blockchain-based authentication system for iot. *Computers & Security*, 2018. 21, 140

[71] Mohamed Tahar Hammi, Erwan Livolant, Patrick Bellot, Ahmed SERHROUCHNI, and Pascale Minet. Mac sub-layer node authentication in ocari. In *Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), International Conference on*, pages 1–6. IEEE, 2016. 21, 139

[72] Mohamed Tahar Hammi, Erwan Livolant, Patrick Bellot, Ahmed Serhrouchni, and Pascale Minet. A lightweight mutual authentication protocol for the iot. In *International Conference on Mobile and Wireless Technology*, pages 3–12. Springer, 2017. 21, 139

[73] Theodore G Handel and Maxwell T Sandford. Hiding data in the OSI network model. In *International Workshop on Information Hiding*, pages 23–38. Springer, 1996. 80

[74] Thomas Hardjono and Ned Smith. Cloud-based commissioning of constrained devices using permissioned blockchains. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 29–36. ACM, 2016. 120

[75] Russell Housley. Using advanced encryption standard (aes) counter mode with ipsec encapsulating security payload (esp). 2004. 94

[76] Russell Housley, Warwick Ford, William Polk, and David Solo. Internet X. 509 public key infrastructure certificate and CRL profile. Technical report, 1998. 48

[77] Russell Housley, Warwick Ford, William Polk, and David Solo. Internet x. 509 public key infrastructure certificate and crl profile. Technical report, 1998. 117

[78] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing iot devices using blockchain platform. In *Advanced Communication Technology (ICACT), 2017 19th International Conference on*, pages 464–467. IEEE, 2017. 119, 120

[79] Min-Shiang Hwang, Chao-Chen Yang, and Cheng-Yeh Shiu. An authentication scheme for mobile satellite communication systems. *ACM SIGOPS Operating Systems Review*, 37(4) :42–47, 2003. 45, 47

[80] ITU. Internet of Things Global Standards Initiative. *Recommendation ITU-T Y.2060*, June 2015. 7

[81] Biggs John. Hackers release source code for a powerful DDoS app called Mirai, October 2016. 8

[82] Saint John Walker. Big data : A revolution that will transform how we live, work, and think, 2014. 127

[83] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1) :36–63, 2001. 109, 116

[84] Ashkan Kalantari, Gan Zheng, Zhen Gao, Zhu Han, and Bjorn Ottersten. Secrecy analysis on network coding in bidirectional multibeam satellite communications. *IEEE Transactions on Information Forensics and Security*, 10(9) :1862–1874, 2015. 45

[85] CJ Kale and TJ Socolofsky. TCP/IP tutorial. 1991. 112

[86] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side channel cryptanalysis of product ciphers. In *European Symposium on Research in Computer Security*, pages 97–110. Springer, 1998. 65

[87] Mahmoud Khasawneh, Izadeen Kajman, Rashed Alkhudaidy, and Anwar Althubyani. A survey on wi-fi protocols : Wpa and wpa2. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 496–511. Springer, 2014. 57

[88] Sudhir Khatwani. *Litecoin Cryptocurrency : A Complete Guide for Absolute Beginners*, November 2017. 70

[89] Mak King and Michael J Riccio. Military Satellite Communications : Then and Now. *Crosslink*, 11(1) :40–47, 2010. 45

[90] Sunny King and Scott Nadal. Ppcoin : Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012. 71

[91] Paul C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, pages 104–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. 8

[92] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. Dtls based security and two-way authentication for the internet of things. *Ad Hoc Networks*, 11(8) :2710–2723, 2013. 13, 101, 103

[93] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. {DTLS} based security and two-way authentication for the Internet of Things. *Ad Hoc Networks*, 11(8) :2710–2723, 2013. 15

[94] Nir Kshetri. Blockchain's roles in strengthening cybersecurity and protecting privacy. *Telecommunications Policy*, 2017. 132

[95] Nandakishore Kushalnagar, Gabriel Montenegro, and Christian Schumacher. IPv6 over low-power wireless personal area networks (6LoWPANs) : overview, assumptions, problem statement, and goals. IETF, IETF, RFC4919, August 2007. 58, 60

[96] Christophe Lagane. BrickerBot, un destructeur d'objets connectés qui agit, avril 2017. 8

[97] Kristin Lauter. The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless communications*, 11(1) :62–67, 2004. 122

[98] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3 :18–23, 2015. 127

[99] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A comparative study of wireless protocols : Bluetooth, UWB, ZigBee, and Wi-Fi. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 46–51. Ieee, 2007. 57

[100] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017. 70

[101] Xiaoyu Li, Yves Quere, Eric Rius, Ingrid Peuziat, and Iwan Le Berre. Des capteurs communicants longue portee a tres bas debit en bande UHF, pour le suivi de la plaisance. *Comite d'organisation*, page 45, November 2015. 45

[102] Helger Lipmaa, Phillip Rogaway, and David Wagner. CTR-mode encryption. In *First NIST Workshop on Modes of Operation*, 2000. 39

[103] Gavin Lowe. A hierarchy of authentication specifications. In *Computer security foundations workshop, 1997. Proceedings., 10th*, pages 31–43. IEEE, 1997. 99

[104] D. Lozneanu, A. Costache, and M. Romanca. Applications of energy model in WSN nodes. In *2014 International Conference on Optimization of Electrical and Electronic Equipment*, pages 852–857, May 2014. 102

[105] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2) :72–93, 2005. 132

[106] Kris Maine, Carrie Devieux, and Pete Swan. Overview of IRIDIUM satellite network. In *WESCON 95. Conference record.'Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies*, page 483. IEEE, 1995. 45

[107] Hitesh Malviya. How blockchain will defend iot. 2016. 119, 120

[108] Itsik Mantin. A Practical Attack on the Fixed RC4 in the WEP Mode. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 395–411. Springer, 2005. 66

[109] V Gayoso Martinez, F Hernández Álvarez, L Hernández Encinas, and C Sanchez Avila. Analysis of ecies and other cryptosystems based on elliptic curves. *J Inform. Assurance and Security*, 6(4) :285–293, 2011. 119, 135

[110] Moffat Mathews and Ray Hunt. Evolution of wireless LAN security architecture to IEEE 802.11 i (WPA2). In *Proceedings of the fourth IASTED Asian Conference on Communication Systems and Networks*, 2007. 54, 55

[111] Craig Mathias. The best choice for enterprise IoT networking is Wi-Fi, December 2014. 50

[112] Craig Mathias. Wi-Fi and the Internet of Things : (Much) more than you think, January 2015. 50, 51

[113] Alexander Maximov and Dmitry Khovratovich. New state recovery attack on RC4. In *Annual International Cryptology Conference*, pages 297–316. Springer, 2008. 66

[114] Máire McLoone and John V McCanny. *System-on-chip architectures and implementations for private-key data encryption*. New York : Kluwer Academic/Plenum Publishers, 2003. 102

[115] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 369–378. Springer, 1987. 71

[116] Matthias Mettler. Blockchain technology in healthcare : The revolution starts here. In *e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on*, pages 1–3. IEEE, 2016. 69

[117] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2) :39–53, 2004. 118

[118] John C Mitchell, Arnab Roy, Paul Rowe, and Andre Scedrov. Analysis of EAP-GPSK authentication protocol. In *International Conference on Applied Cryptography and Network Security*, pages 309–327. Springer, 2008. 66

[119] Michael H Morris and J Don Trotter. Institutionalizing entrepreneurship in a large company : A case study at ATandT. *Industrial Marketing Management*, 19(2) :131–139, 1990. 47

[120] D M'Raihi, M Bellare, F Hoornaert, D Naccache, and O Ranen. HOTP : An HMAC-based one-time password algorithm. IETF, RFC 4226, December 2005. 10, 79

[121] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. Totp : Time-based one-time password algorithm. Technical report, 2011. 79

[122] Hyung G Myung. Technical overview of 3GPP LTE. *Polytechnic University of New York*, 2008. 41

[123] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system, 2008. 16, 69, 126

[124] N.Sornin (Semtech), M.Luis (Semtech), T.Eirich (IBM), T.Kramp (IBM), O.Hersent (Actility). LoRaWAN Specification. *LoRa Alliance*, 1.0, 2015. 35, 37

[125] Loutfi Nuaymi. *WiMAX : technology for broadband wireless access*. John Wiley and Sons, 2007. 47, 48

[126] Magnus Nystrom. Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512. 2005. 10, 11, 59, 80

[127] Toshihiro Ohigashi and Masakatu Morii. A practical message falsification attack on WPA. *Proc. JWIS*, 2009. 54, 66

[128] Toshihiro Ohigashi and Masakatu Morii. A practical message falsification attack on WPA. Technical report, 2009. 54

[129] Eng Hwee Ong. Performance analysis of fast initial link setup for IEEE 802.11 ai WLANs. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pages 1279–1284. IEEE, 2012. 51

[130] Eng Hwee Ong, Jarkko Kneckt, Olli Alanen, Zheng Chang, Toni Huovinen, and Timo Nihtila. IEEE 802.11 ac : Enhancements for very high throughput WLANs. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*, pages 849–853. IEEE, 2011. 51

[131] Zigbee Alliance Organization. Zigbee specification. *Document 053474r20*, 2012. 60, 61, 67, 68

[132] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess : a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, 9(18) :5943–5964, 2016. 120

[133] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Towards a novel privacy-preserving access control model based on blockchain technology in iot. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pages 523–533. Springer, 2017. 120

[134] Raj Pandya, Davide Grillo, Edgar Lycksell, Philippe Mieybegue, Hideo Okinaka, and Masami Yabusaki. IMT-2000 standards : Network aspects. *IEEE Personal Communications*, 4(4) :20–29, 1997. 40

[135] Douglas Pike, Patrick Nosker, David Boehm, Daniel Grisham, Steve Woods, and Joshua Marston. PoST White Paper. 71

[136] Christoph Rensing, Martin Karsten, and Burkhard Stiller. AAA : a survey and a policy-based architecture and framework. *IEEE network*, 16(6) :22–27, 2002. 54

[137] Eric Rescorla and Nagendra Modadugu. Rfc 6347, datagram transport layer security version 1.2. *Internet Engineering Task Force*, 2012. 13, 101

[138] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978. 48

[139] Ronald L Rivest, Adi Shamir, and Leonard M Adleman. Cryptographic communications system and method, 1983. US Patent 4,405,829. 48, 110

[140] Ayan Roy-Chowdhury, John S Baras, Michael Hadjitheodosiou, and Spyro Papademetriou. Security issues in hybrid networks with a satellite component. *IEEE Wireless Communications*, 12(6) :50–61, 2005. 45

[141] Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, and Eugenio Di Sciascio. Semantic blockchain to improve scalability in the internet of things. *Open Journal of Internet Of Things (OJIOT)*, 3(1) :46–61, 2017. 120

[142] Erik Seedhouse. *SpaceX : making commercial spaceflight a reality*. Springer Science and Business Media, 2013. 45

[143] Pablo Lamela Seijas, Simon J Thompson, and Darryl McAdams. Scripting smart contracts for distributed ledger technology. *IACR Cryptology ePrint Archive*, 2016 :1156, 2016. 71

[144] Janusz J Sikorski, Joy Haughton, and Markus Kraft. Blockchain technology in the chemical industry : Machine-to-machine electricity market. *Applied Energy*, 195 :234–246, 2017. 69

[145] William Allen Simpson. PPP challenge handshake authentication protocol (CHAP). 1996. 10, 67, 79

[146] John S Sobolewski. Cyclic redundancy check. 2003. 53

[147] Nisha Ashok Somani and Yask Patel. Zigbee : A low power wireless technology for industrial applications. *International Journal of Control Theory and Computer Modelling (IJCTCM) Vol*, 2 :27–33, 2012. 60

[148] Junhyuk Song, Radha Poovendran, Jicheol Lee, and Tetsu Iwata. The aes-cmac algorithm. Technical report, 2006. 38

[149] Data Encryption Standard. Data encryption standard. *Federal Information Processing Standards Publication*, 1999. 46

[150] NIST-FIPS Standard. Announcing the advanced encryption standard (AES). *Federal Information Processing Standards Publication*, 197 :1–51, 2001. 91

[151] Dorothy Stanley, Jesse Walker, and Bernard Aboba. Extensible authentication protocol (EAP) method requirements for wireless LANs. Technical report, 2005. 49

[152] statista.com. online statistics portal, and one of the world's most successful statistics databases. 2016. 7

[153] Adam Stubblefield, John Ioannidis, Aviel D Rubin, et al. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. In *NDSS*, 2002. 66

[154] Michel Terre. Wifi, le Standard 802.11, couche physique et couche MAC (Version 1.1). Technical report, Mars 2007. 51

[155] Erik Tews. Attacks on the WEP protocol. *IACR Cryptology ePrint Archive*, 2007 :471, 2007. 66

[156] Erik Tews and Martin Beck. Practical attacks against WEP and WPA. In *Proceedings of the second ACM conference on Wireless network security*, pages 79–86. ACM, 2009. 66

[157] Dave Thaler and B Aboba. RFC 5218, What Makes For a Successful Protocol ? Technical report, 2008. 54

[158] Dylan Tweney. The Hidden DOWNSIDE of Wireless Networking. 42(6) :36, June 2006. 53

[159] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks : Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1313–1328. ACM, 2017. 66

[160] Pavel Vasin. Blackcoin's proof-of-stake protocol v2. *URL : https ://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf*, 2014. 71

[161] Steven J Vaughan-Nichols. Achieving wireless broadband with WiMax. volume 37, pages 0010–13. IEEE Computer Society, 2004. 47

[162] Lizhe Wang, Gregor Von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing : a perspective study. *New Generation Computing*, 28(2) :137–146, 2010. 72

[163] Shiyong Wang, Jiafu Wan, Daqiang Zhang, Di Li, and Chunhua Zhang. Towards smart factory for industry 4.0 : a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, 101 :158–168, 2016. 127

[164] Joshua Wright. Killerbee : practical zigbee exploitation framework. In *11th ToorCon conference, San Diego*, 2009. 67

[165] Takuma Yamamoto. *Fujitsu : what mankind can dream technology can achieve*. Toyo Keizai, 1992. 47

[166] Wei Ye and John Heidemann. Medium access control in wireless sensor networks. In *Wireless sensor networks*, pages 73–91. Springer, 2004. 62

[167] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Huaimin Wang. Blockchain Challenges and Opportunities : A Survey. *Work Pap*, 2016. 71

[168] Hubert Zimmermann. OSI reference model–The ISO model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4) :425–432, 1980. 51

[169] Eustathia Ziouva and Theodore Antonakopoulos. CSMA/CA performance under high traffic conditions : throughput and delay analysis. *Computer communications*, 25(3) :313–321, 2002. 48

[170] Alf Zugenmaier and Hiroshi Aono. Security technology for sae/lte. *Ntt Docomo Technical Journal*, 11(3) :27–30, 2009. 41

Index

A

Approche 79, 87, 91, 105, 121
Association 21, 37, 42, 46, 60, 63, 64, 80
Authentification 8, 31, 42, 48, 58, 61, 79, 87, 91, 105, 115
Autonomie 27

B

Blockchain 16, 69, 105, 115

C

Confidentialité persistante 32
Cyberattaque 8, 33, 117

E

Energie consommée 80, 101, 110, 128
Evaluation 83, 88, 96, 109, 126, 130, 139
Evolutivité 32

I

Identification 18
Implémentation 19, 63, 77, 120, 126
Internet des Objets (IoT) 7, 27, 64, 91
Interopérabilité 27

L

Langage C 17, 84, 109
Langage C++ 128
Langage formel 96, 98
Langage Solidity 17, 109

M

Man In The Middle 97
Message Authentication Code (MAC) 38, 42, 44
Mode d'opération 91

N

Non rejeu 32, 96, 118

O

Objet 7, 27, 77
OCARI 62, 77, 89

R

Réseau 35, 69
Résilience 32

S

Sécurité 8, 31, 36, 79, 87, 91, 105, 115
Smart contract 18, 72, 109, 121
Sous couche MAC 37, 49, 60, 62, 80

T

Taille du marché IoT 27
Technologie de communication 35, 64
Temps d'exécution 80, 83, 89, 101, 110, 128

W

WSN 7, 36, 79, 140

Titre : Sécurisation de l'Internet des objets

Mots clés : Sécurité, authentification, Internet des objets, Blockchain, réseau capteur sans fil, intégrité, confidentialité, optimisation d'énergétique.

Résumé :

L'Internet des objets (IoT) a bouleversé le monde des technologies de l'information.

Ce nouveau phénomène devient inévitable et concerne déjà presque tous les domaines, de l'horlogerie aux usines automatisées.

L'IdO simplifie notre vie quotidienne et crée de la valeur pour les particuliers et les entreprises. Les objets, également appelés entités, sont très hétérogènes, utilisent différentes technologies de communication et sont généralement des périphériques à capacité limitée. Par conséquent, la sécurisation de tels systèmes soulève de nombreux défis. Les entités en communication doivent s'authentifier mutuellement et protéger l'intégrité et la confidentialité des données qu'elles échangent, tout en utilisant des algorithmes légers, rapides et économes en énergie.

Dans ce manuscrit, nous proposons des systèmes de sécurité robustes et innovants, conçus spécialement pour des objets limités. Nous avons effectué des implémentations réelles et les résultats obtenus prouvent l'efficacité de nos protocoles.

Title : Securing the Internet of things

Keywords : Security, Authentication, Internet of Things, Blockchain, Wireless Sensor Network, Integrity, Confidentiality, Energy-efficiency.

Abstract :

The Internet of Things (IoT) has overturned the information technology world. This new phenomenon is becoming inescapable and already covers almost all fields, from watchmaking to automated factories.

IoT simplifies our everyday life and creates value for people and businesses. Things, also called entities, are very heterogeneous, use different communication technologies and, generally, are limited capacity devices. Therefore securing such systems raises many challenges.

Communicating entities should authenticate each other and protect the integrity and the confidentiality of the data they exchange while using lightweight, fast and energy-efficient algorithms.

In this manuscript, we propose robust and innovative security systems, designed especially for constrained IoT devices. We carried out real implementations and the obtained results prove the efficiency of our protocols.