Architecture de communication sécurisée d'une flotte de drones

Jean-Aimé Maxa

▶ **To cite this version:**

Jean-Aimé Maxa. Architecture de communication sécurisée d'une flotte de drones. Réseaux et télécommunications [cs.NI]. Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2017. Français. NNT : . tel-01570242v1

HAL Id: tel-01570242

https://theses.hal.science/tel-01570242v1

Submitted on 28 Jul 2017 (v1), last revised 3 Oct 2017 (v2)

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.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *28/06/2017* **par :**

Jean Aimé Maxa

Architecture de communication sécurisée d'une flotte de drones

---

JURY

Francine Krief	Professeur des Universités	ENSEIRB
Damien Sauveron	Maitre de conférences HDR	Université de Limoges
Rami Langar	Professeur des Universités	UPEM
Christophe Bidan	Professeur des Universités	Supélec
Nicolas Larrieu	Enseignant chercheur	ENAC
Mohamed Slim Ben Mahmoud	Architecte/direction du centre d'expertise Cybersécurité	ALTRAN
Damien Servant	Ingénieur	Delair Tech (Invité)

École doctorale et spécialité :
EDSYS : Informatique 4200018
Unité de Recherche :
Groupe RESCO du Laboratoire TELECOM, ENAC
Directeur(s) de Thèse :
Nicolas Larrieu et *Mohamed Slim Ben Mahmoud*
Rapporteurs :
Pr. Francine Krief et *Dr. Damien Sauveron*

Université de toulouse

Thèse

Présentée pour l'obtention du grade de DOCTEUR
DE L'UNIVERSITE de TOULOUSE
par

Jean Aimé Maxa

Architecture de communication sécurisée pour une flotte de drones

---

Spécialité : Informatique

Résumé

Grâce aux progrès de miniaturisation des systèmes embarqués, les mini-drones qu'on appelle en anglais *Small Unmanned Aerial Vehicle* (UAVs) sont apparus et permettent de réaliser des applications civiles à moindres coûts. Pour améliorer leurs performances sur des missions complexes (par exemple, pour contourner un obstacle), il est possible de déployer une flotte de drones coopératifs afin de partager les tâches entre les drones. Ce type d'opération exige un niveau élevé de coopération entre les drones et la station de contrôle. La communication entre les drones de la flotte est donc un enjeu important dans la réalisation des opérations d'une flotte de drones. Parmi les différentes architectures de communication qui existent, le réseau ad hoc s'avère être une solution efficace et prometteuse pour l'opération d'une flotte de drones. Un réseau ad hoc de drones ou UAV Ad hoc Network (UAANET) est un système autonome constitué d'une flotte de mini-drones et d'une ou plusieurs station(s) sol. Ce réseau peut être considéré comme une sous-catégorie d'un réseau ad hoc mobile (MANET) avec des caractéristiques spécifiques (vitesse importante des nœuds, modèle de mobilité spécifique, etc.) qui peuvent engendrer des baisses de performance du protocole de routage utilisé.

Par ailleurs, la nature partagée du support de transmission et l'absence d'une infrastructure fixe pour vérifier l'authenticité des nœuds et des messages posent un problème de sécurité des communications. Compte tenu du caractère critique des données de charge utile échangées (en effet, un attaquant peut capturer un drone et l'utiliser à des fins malveillantes), il est important que les messages échangés soient authentifiés et qu'ils n'ont pas été modifiés ou retardés par un attaquant. L'authentification des messages est donc un des objectifs à atteindre pour garantir la sécurité du système Unmanned Aerial System (UAS) final. Diverses solutions de sécurité ont été conçues pour les réseaux sans fil, puis ont ensuite été adaptées aux réseaux MANET. Ces solutions peuvent s'étendre à des applications pour les réseaux UAANET, c'est pourquoi nous proposons dans cette thèse une architecture de communication fiable et sécurisée pour les flottes des drones.

Dans ce travail, nous avons étudié en premier lieu l'application d'un réseau ad hoc mobile pour les flottes de drones. Nous examinons en particulier le comportement des protocoles de routage ad hoc existants dans un environnement UAANET. Ces solutions sont ainsi évaluées pour permettre d'identifier le protocole adéquat pour l'échange des données. Cela nous amène dans un deuxième temps, à proposer un protocole de routage intitulé Secure UAV Ad hoc routing Protocol (SUAP) qui garantit l'authentification des messages et détecte l'attaque *wormhole*. Cette attaque peut être définie comme un scénario dans lequel

un attaquant enregistre les paquets en un point, et les rejoue à un autre point distant. L'attaque *wormhole* est particulièrement dangereuse lorsqu'un protocole de routage réactif (qui utilise le nombre de sauts comme métrique d'une route) est utilisé. Pour contrer cette attaque, le protocole SUAP permet d'une part d'assurer des services de livraison de donnés (une vidéo de télésurveillance) entre un drone distant et une station sol. D'autre part, le protocole SUAP possède également des partitions de sécurisation qui se basent sur une signature et une fonction de hachage pour assurer l'authentification et l'intégrité des messages. En ce qui concerne l'attaque *wormhole*, une technique qui consiste à corréler le nombre de sauts et la distance relative entre deux nœuds voisins est utilisée. Ce mécanisme permet de déduire la présence ou non d'un tunnel *wormhole* dans le réseau.

En outre, cette architecture de communication est conçue avec une méthodologie de prototypage rapide avec l'utilisation d'une méthode orientée modèle pour tenir compte du besoin de validation du système UAS final. La validation est nécessaire pour certifier le fonctionnement de la flotte dans le cas où elle est utilisée pour l'échange des flux de commande. Cette méthode de conception est réalisée avec l'outil Mathworks Simulink & Stateflow qui possède un générateur automatique de codes et des outils de vérification formelle de conception.

Notre solution est ensuite évaluée à l'aide de deux moyens. D'une part, par simulation et émulation avec un outil hybride simulant un sous-ensemble de l'environnement UAANET (utilisation des modèles de mobilité réels et exécution sur un environnement Linux). Et d'autre part en environnement réel par une implémentation sur des cartes ARM intégrées aux drones DT18. Plusieurs tests ont été réalisés pour valider les résultats obtenus par simulation et émulation. Les résultats obtenus ont montré que le protocole SUAP assure les propriétés de sécurité d'authentification et d'intégrité et protège contre l'attaque *wormhole*. Il permet également l'échange des données temps réel avec une qualité de service acceptable.

Abstract

Advances in miniaturization of embedded systems have helped to produce small Unmanned Aerial Vehicles (UAVs) with highly effective capacity. In order to improve their capability in civilian complex missions (for instance, to bypass an obstruction), it is now possible to deploy UAV swarms, in which cooperative UAVs share different tasks. This type of operations needs a high level of coordination between UAVs and Ground Control Station (GCS) through a frequent exchange of information. The communication capabilities are therefore an important objective to achieve for effective UAV swarm operations. Several communication architectures can be used to allow communication between UAVs and GCS. Ad hoc network is one of them and is an effective and promising solution for multi-UAV systems. Such a network is called UAANET (UAV Ad hoc Network) and is an autonomous system made of a UAV swarm and one or several GCS (Ground Control Station). This network can also be considered as a sub category of the well-known MANET (Mobile Ad hoc network). However, it has some specific features (such as node velocity, specific mobility model) that can impact performance of routing protocols.

Furthermore, the nature of the wireless medium, along with the lack of fixed infrastructure, which is necessary to verify node and message authentication, create security breaches. Specifically, given the critical characteristic of the real-time data traffic, message authentication proves to be an important step to guarantee the security of the final UAS (composed of UAV swarm). Security of routing protocols has been widely investigated in wired networks and MANETs, but as far as we are aware, there is no previous research dealing with the security features of UAANET routing protocols. Those existing solutions can be adapted to meet UAANET requirements. With that in mind, in this thesis, we propose a secure and reliable communication architecture for a UAV swarm.

In this work, the creation of UAANET has first been concieved. In order to do this, we studied the impact of existing MANET routing protocols into UAANET to assess their performance and to select the best performer as the core of our proposed secure routing protocol. Accordingly, we evaluated those existing routing protocols based on a realistic mobility model and realistic UAANET environment. Based on this first study, we created a secure routing protocol for UAANET called SUAP (Secure UAV Ad hoc routing Protocol).

On the one hand, SUAP ensures routing services by finding routing paths between nodes to exchange real time traffic (remote monitoring video traffic). On the other hand, SUAP ensures message authentication and provides detection to avoid wormhole attack. The SUAP routing protocol is a reactive routing protocol using public key cryptography and

hash chains. In order to detect wormhole attack, a geographical leash-based algorithm is used to estimate the correlation between the packet traveled distance and the hop count value.

We also contribute to the certification of the secure communication system software through a Model-Driven Development (MDD) approach. This certification is needed to validate the operation of the UAV swarm, especially in cases where it is used to exchange control and command traffic. We used Simulink and Stateflow tools and formal verification tools of Matlab Software to design SUAP routing protocol.

The evaluation of the effectiveness of SUAP has been executed both through emulation and real experiment studies. Our results show that SUAP ensures authentication and integrity security services and protects against a wormhole attack. It also provides an acceptable quality of service for real-time data exchanges.

Remerciements

Ce manuscrit constitue l'aboutissement de travaux de recherche qui n'auraient pas été possibles, ou du moins qui ne se seraient pas déroulés dans les mêmes conditions, sans la participation de quelques personnes que je me dois de remercier.

Tout d'abord, je tiens à remercier mes deux directeurs de thèse, Nicolas Larrieu (enseignant-chercheur à l'ENAC) et Mohamed Slim Ben Mahmoud (enseignant-chercheur à l'ENAC), pour leurs encadrements et pour m'avoir fait confiance en me donnant ce sujet de thèse. Leur encadrement scientifique et leur grande disponibilité m'ont permis de travailler dans les meilleures conditions.

Je suis naturellement très reconnaissant pour le temps et le travail accordés par l'ensemble des membres du jury pour ma thèse : Mme Francine Krief, professeur à l'université de Bordeaux ; M. Damien Sauveron, maitre des conférences à l'Université de Limoges ; M. Rami Langar, professeur à l'université Paris-Est Marne-la-Vallée ; M. Christophe Bidan, professeur à supélec. En particulier, je remercie Mme Francine Krief et M. Damien Sauveron d'avoir accepté d'être les rapporteurs de cette thèse. Merci d'avoir participé à l'amélioration de ce document grâce à vos relectures poussées et tous ces retours constructifs.

Je suis aussi très reconnaissant envers l'ANRT et l'entreprise Delair Tech pour le financement de ma thèse. Mes remerciements sont aussi à l'attention de Bastien Mancini et Michael Delagarde, CEO de Delair Tech, pour avoir soutenu cette thèse. L'autonomie qu'ils ont su me laisser et les responsabilités qu'ils m'ont confiées ont rendu mon travail des plus agréables. Je souhaite également inclure dans mes remerciements, mes collèges de Delair Tech pour la bonne ambiance dans laquelle nous avons travaillé ensemble.

Je me dois également de remercier les membres de l'équipe TELECOM, axe Resco de l'ENAC que j'ai côtoyé : Alain, Quentin, Fabien, Mickael, Ons, Stefano (merci de m'avoir donné le nom : Giovanni amato), Rita et Gilles. Ils ont participé à l'ambiance de travail studieuse et décontractée. J'en profite pour remercier les autres collègues du bâtiment Z. En particulier, j'aimerais m'arrêter sur Serge Roux (merci en particulier pour ces moments où tu m'as aidé à configurer mon environnement de travail, et ce toujours dans la bonne humeur).

Mes remerciements vont également à l'ensemble des personnes, scientifiques ou non qui ont participé à la relecture de ce document durant sa rédaction. Je pense notamment à Souaad et à Coralie.

D'un point de vue plus personnel, je tiens particulièrement à remercier ma famille sans qui je ne serais pas là. Je pense en particulier à ma mère qui m'a encouragé durant ma thèse. Je pense également à mes trois frères : Rolland, Zico et Tony pour leur compagnie et leur bonne humeur. Je pense aussi à Haingo et Sarindra Ramanantsoa qui m'ont accueilli lorsque je venais d'arriver en métropole pour m'aider à m'intégrer à ma nouvelle vie.

Je tiens également mes sincères gratitudes à la famille Rajaonarivelo : Miora, Jemima, Manase, Lala (Aurélie) et Alain. Ils font partie de ma deuxième famille à Toulouse. Je les remercie pour leurs bonnes volontés et leurs fraternités.

Un grand merci également à mes amis de Toulouse (Pan Sebastien et Alain), mes amis de Bordeaux (Freedy et Mamitiana), de Montpellier : Stephan Célestin et notre cher président Antonio Nirina de Tananarive. Ils ont tous joué chacun à leur manière, un rôle lors de cette thèse. À celles et ceux (que j'ai oublié) qui, au moins pendant un moment, m'ont soutenu : merci.

Table des matières

Introduction générale **1**

1 Architecture de communication pour une flotte de drones **13**

1.1 Introduction 15

1.2 Systèmes aéronautiques coopératifs sans pilote 15

1.2.1 Description fonctionnelle d'un drone 15

1.2.2 Unmanned Aircraft/Aerial Systems 17

1.2.2.1 Charge utile 17

1.2.2.2 Station sol 17

1.2.3 Drones dans des opérations civiles 18

1.2.4 Flotte de drones 19

1.3 Architectures de communication possibles pour une flotte de drones 20

1.3.1 Architecture de communication centralisée 20

1.3.2 Architecture de communication par satellite 21

1.3.3 Architecture de communication réseau cellulaire (Réseau semi-centralisé) 22

1.3.4 Architecture de communication ad hoc 22

1.3.5 Réseau ad hoc de drones 25

1.3.6 Projets de réseau ad hoc de drones 29

1.4 Protocoles de communication d'un réseau ad hoc de drones 31

1.4.1 Liaison de données (*Medium Access Control*) 31

1.4.2 Niveau réseau : routage dans les réseaux UAANET 32

1.4.2.1 Protocole hiérarchique 34

1.4.2.2 Protocole réactif 34

1.4.2.3 Protocole proactif 36

1.4.2.4 Protocole géographique 38
1.5 Discussions et Conclusions 39

2 Sécurité dans un réseau ad hoc de drones 41
2.1 Introduction 42
2.2 Vulnérabilité des réseaux UAANET 43
2.3 Attaques dans les réseaux UAANET 45
2.3.1 Description (modèle) des attaquants 45
2.3.2 Analyse de risques des attaques 46
2.3.2.1 Attaques au niveau de la couche physique 46
2.3.2.2 Attaques au niveau de la couche liaison 46
2.3.3 Attaques liées aux protocoles de routage 48
2.3.3.1 Attaques pouvant être exécutées durant la phase de découverte de routes 49
2.3.3.2 Attaques pouvant être exécutées durant la phase de maintenance 53
2.3.4 Analyse des différentes attaques au niveau réseau 53
2.4 Solutions existantes et limites 54
2.4.1 Notions et mécanismes de base de la sécurité 54
2.4.2 Techniques cryptographiques 56
2.4.2.1 Cryptographie symétrique et asymétrique 56
2.4.2.2 Signatures 57
2.4.2.3 Authentification des messages 58
2.4.2.4 Fonctions de hachage 58
2.4.2.5 Certificat 59
2.4.2.6 Infrastructure à clés publiques 60
2.4.3 Authentification des nœuds 60
2.4.4 Systèmes de réputation 61
2.4.5 Discussions 62
2.5 Protocoles de routage ad hoc sécurisé existants 63
2.6 Discussions 68
2.7 Conclusions 72

3 Méthodologie de développement d'un logiciel embarqué pour un système de drones 73

3.1 Introduction . . . 74

3.2 Conception de systèmes embarqués critiques . . . 76

3.2.1 Norme de sûreté . . . 76

3.2.2 Cycle de vie de développement logiciel . . . 77

3.2.3 Processus de validation de logiciel dans la norme DO-178B . . . 79

3.3 Méthodologie orientée modèle . . . 80

3.3.1 Présentation des différentes étapes . . . 81

3.3.2 Analyse des modèles . . . 83

3.3.3 Avantages de l'utilisation de la méthodologie MDD (Model Driven Development) . . . 85

3.4 Certification d'un système UAS . . . 86

3.4.1 Apport de l'approche MDD dans la certification UAS . . . 88

3.5 Choix d'outils pour l'application de la méthodologie MDD . . . 90

3.5.1 Synthèse sur les chaînes d'outils . . . 94

3.6 Conclusions . . . 95

4 Protocole SUAP (Secure UAV Ad hoc routing Protocol) 96

4.1 Introduction . . . 98

4.2 Choix d'un protocole de routage . . . 99

4.2.1 Protocole expérimental et outil de test de protocole de routage UAA-NET . . . 99

4.2.2 Architecture de l'outil de test . . . 100

4.2.3 Résultats d'évaluation des performances . . . 103

4.2.3.1 Résultats de connectivité . . . 104

4.2.4 Délai moyen de ré-établissement d'une route après une perte de connectivité . . . 105

4.2.4.1 Résultats du délai et de l'*overhead* . . . 107

4.3 Protocole SUAP . . . 108

4.3.1 Modèle réseau et d'attaques considérées dans la conception du protocole SUAP . . . 108

4.3.2 Description du protocole SUAP . . . 110

4.3.3 Analyse des solutions existantes . . . 111
4.3.4 Protocole SAODV . . . 112
4.3.4.1 Signature numérique dans SAODV . . . 112
4.3.4.2 Chaîne de hachage dans SAODV . . . 115
4.3.5 Analyse des vulnérabilités de SAODV . . . 116
4.3.5.1 Attaque *wormhole* . . . 116
4.3.5.2 Attaque avec un seul attaquant . . . 117
4.3.5.3 État de l'art des solutions contre l'attaque *wormhole* . . . 118
4.3.5.4 Synthèse des travaux existants . . . 122
4.3.6 Proposition d'un mécanisme pour détecter et contrer l'attaque *wormhole* . . . 123
4.3.7 Proposition d'un mécanisme pour contrer l'attaque *wormhole* réalisé par un seul attaquant . . . 127
4.3.8 Limites du protocole SUAP . . . 129
4.3.9 Différents formats des paquets de contrôle dans SUAP . . . 131
4.3.9.1 Format des messages *SRREQ* . . . 132
4.3.9.2 Format des messages *Secure Hello* . . . 133
4.3.9.3 Format des messages *SRERR* . . . 133
4.3.10 Analyse de sécurité du protocole SUAP . . . 134
4.4 Vérification formelle des propriétés de sécurité du protocole SUAP . . . 135
4.4.1 Nécessité des procédures de vérification . . . 136
4.4.2 Choix de l'outil AVISPA . . . 137
4.4.3 Utilisation de l'outil AVISPA . . . 138
4.4.4 Cas d'application du protocole SUAP . . . 139
4.4.4.1 Analyse de spécification du protocole SUAP . . . 140
4.5 Mise en œuvre du protocole SUAP . . . 142
4.5.1 Architecture de l'algorithme SUAP . . . 142
4.5.2 Modélisation du protocole SUAP . . . 143
4.5.2.1 Partition de routage . . . 143
4.5.2.2 Partition de sécurisation . . . 148
4.5.2.3 Partition d'interfaçage du matériel avec la partition de sécurisation et de routage . . . 149

4.5.3 Apport de l'approche orientée modèle dans la conception du protocole SUAP . . . 151
4.5.4 Implémentation du protocole SUAP . . . 152
4.6 Conclusions . . . 153

5 Évaluation des performances du protocole SUAP par simulation et par expérimentation réelle 155

5.1 Introduction . . . 156
5.2 Cadre d'expérimentation . . . 157
5.2.1 Topologies réseau d'études . . . 157
5.2.1.1 Drones DT18 . . . 158
5.2.1.2 Stations de contrôle . . . 158
5.2.1.3 Liaison de communication . . . 159
5.3 Validation de la partition de routage . . . 159
5.3.1 Topologie de test de routage . . . 159
5.3.2 Métriques considérées . . . 160
5.3.3 Résultats obtenus et interprétation . . . 162
5.3.3.1 Taille d'overhead . . . 162
5.3.3.2 Stabilité d'une route . . . 162
5.3.3.3 Durée de ré-établissement d'une route . . . 165
5.3.3.4 Délai de bout en bout . . . 166
5.3.3.5 Délai moyen pour l'acheminement des paquets de données utiles . . . 168
5.3.3.6 Perte de paquet de données . . . 168
5.3.3.7 Discussion sur la validation de routage . . . 170
5.4 Validation des fonctions de sécurité du protocole SUAP . . . 170
5.4.1 Validation de la partition de sécurisation (authentification des messages) en environnement émulé . . . 171
5.4.1.1 Paramètres d'émulation . . . 171
5.4.1.2 Métriques considérées . . . 172
5.4.1.3 Taux de livraison des paquets . . . 173
5.4.1.4 Délai de bout en bout . . . 174
5.4.1.5 Overhead . . . 175

5.4.1.6 Connectivité 176
5.4.2 Validation de la partition de sécurisation (authentification des messages) en environnement réel 177
5.4.2.1 Taux de pertes des paquets 178
5.4.2.2 Overhead 179
5.4.2.3 Durée de vie moyenne d'une route 179
5.4.2.4 Pourcentage de routes établies entre station1 et Dr3 180
5.4.2.5 Délai de bout en bout 181
5.4.2.6 Évalutation du protocole AODV en environnement réel en présence d'un attaquant *blackhole* 182
5.4.3 Validation du mécanisme de détection contre l'attaque *wormhole* . . 184
5.5 Conclusions 188

6 Conclusions et perspectives 190
6.1 Travaux réalisés 191
6.1.1 Architecture de communication d'une flotte de drones 192
6.1.2 Besoin en sécurité dans un réseau UAANET 193
6.1.3 Application d'une méthodologie orientée modèle pour le développement du protocole SUAP 194
6.1.4 Validation fonctionnelle de la partition de routage et de sécurisation 194
6.2 Perspectives 195
6.2.1 Étude de performances complémentaires pour SUAP 196
6.2.2 Sécurité applicative des échanges en environnement UAANET . . . 196
6.2.3 Conception d'une infrastructure de gestion de clé adaptée au contexte UAANET 197

Publications 199

A Annexe A 201
A.1 Outils de simulation et d'émulation 201
A.1.1 Mesure passive 201
A.1.2 Mesure active 202
A.1.3 Introduction au framework *Virtualmesh* 202
A.1.4 Prise en compte de la mobilité dans le système 203

B Annexe B **206**

B.1 Vérification formelle avec AVISPA 206

B.1.1 Quelques outils de vérification formelle existants 206

B.1.2 Architecture logicielle de l'outil AVISPA 207

B.1.3 Simulation d'intrus dans AVISPA. Attaque de type confidentialité . 208

Bibliographie **228**

Table des figures

1 Architecture de communication pour une flotte de drones 13

1.1 Exemple de drone à voilure fixe 16

1.2 Exemple de drone multirotor 16

1.3 Illustration d'une station sol déployée sur le terrain 18

1.4 Exemple d'utilisation d'un drone pour une application de surveillance électrique réalisée par l'entreprise Delair-Tech 19

1.5 Principales architectures de communication permettant la communication pour les flottes de mini-drones 21

1.6 Réseau ad hoc de drones 25

1.7 Illustration des différents types de mobilité PPRZM 28

1.8 Illustration d'un mécanisme de routage ad hoc par l'établissement d'une route de A vers D 33

2 Sécurité dans un réseau ad hoc de drones 41

2.1 Attaque *blackhole* 50

2.2 Illustration de l'attaque *wormhole* 51

2.3 Attaque consommation de ressources 53

2.4 Classification des différentes vulnérabilités qui touchent le routage d'un réseau UAANET 55

2.5 Liste des techniques cryptographiques 57

2.6 Envoi de données sur plusieurs chemins 65

3 Méthodologie de développement d'un logiciel embarqué pour un système de drones 73

3.1 Cycle de développement en V 78
3.2 Méthodologie de développement orienté modèle pour les systèmes embarqués complexes 82
3.3 Illustration de l'enchainement de la vérification du modèle jusqu'à la génération de code 84
3.4 Illustration des rapports de vérification de notre modèle 84
3.5 Illustration des classes de partition de notre protocole de routage sécurisé . 90
3.6 Conception de la classe de partition de routage 91
3.7 Utilisation de Embedded Coder pour transformer les modèles en code source 92
3.8 Code de collage liant le système d'exploitation au modèle 93

4 Protocole SUAP (Secure UAV Ad hoc routing Protocol) 96

4.1 Scénario pour lequel notre protocole expérimental doit être adapté 100
4.2 Vue globale du système destiné à tester les protocoles de routage par émulation 101
4.3 Cas où le mécanisme *greedy forwarding* est inefficace 102
4.4 Extraction des états stables et états instables 105
4.5 Délai de ré-établissement de route après une perte de connexion par protocole (axes des abscissesh en ms) 106
4.6 Format d'extension d'un message SAODV 113
4.7 Illustration du mécanisme de chaîne de hachage dans SAODV 114
4.8 Format d'extension d'un message RREQ avec double signature 114
4.9 Illustration de l'attaque *wormhole* 117
4.10 Illustration de l'attaque avec un seul attaquant exécutant une version simplifiée de l'attaque *wormhole* 117
4.11 Illustration de la limite d'utilisation du mécanisme de *Geographical Leashes* 120
4.12 Attaque de type *wormhole* avec deux attaquants 123
4.13 Attaque (version simplifiée de l'attaque *wormhole*) avec un seul attaquant . 128
4.14 Format de l'extension Secure RREQ 132
4.15 Format de l'extension du paquet *Secure Hello* 134
4.16 Format de l'extension *Secure Error* 134
4.17 Diagramme des mécanismes du protocole SUAP 135
4.18 Procédure de vérification automatisée 137

4.19 Exemple d'échange dans le protocole SUAP avec 5 nœuds 139
4.20 Spécification HLPLS du nœud source 140
4.21 Diagramme de classe haut niveau représentant le protocole SUAP 142
4.22 Conception de la classe de partition de routage 143
4.23 Diagramme (version simplifiée) de blocs Simulink pour la fonctionnalité du routage 145
4.24 Diagramme à états-transitions de routage des paquets (version simplifiée) . 146
4.25 Diagramme à états-transitions d'ordonnancement des paquets (version simplifiée) 148
4.26 Modèle d'architecture pour la conception de la partition de sécurisation . . 149
4.27 Extrait des diagrammes à états-transitions pour le bloc content_extractor . 150
4.28 Synthèse des différentes démarches de validation pour la conception du protocole SUAP 152
4.29 Architecture du système Linux avec SUAP 154

5 Évaluation des performances du protocole SUAP par simulation et par expérimentation réelle 155

5.1 Topologie commune de test réel 158
5.2 Topologie de test pour la partition de routage du protocole SUAP 160
5.3 Mouvement du drone Dr2 161
5.4 Stabilité de route entre le nœud station 1 et le nœud Dr2 163
5.5 Variation de la puissance de réception du signal du drone Dr2 164
5.6 Stabilité moyenne d'une route entre Dr2 et la station sol en fonction de la puissance du signal du drone Dr2 165
5.7 Délai de ré-établissement d'une route en cas de perte de lien 166
5.8 Répartition des valeurs délai de bout en bout au cours de la mission 167
5.9 Délai moyen pour l'acheminement des paquets de données utiles 169
5.10 taux de pertes mesurés 169
5.11 Illustration de l'attaque *blackhole* 171
5.12 Deux attaquants *blackhole* 172
5.13 Taux de livraison pour les deux protocoles avec un et deux attaquants *blackhole* 173
5.14 Délai de bout en bout pour chaque protocole en variant le nombre d'attaquants 175

5.15 Résultats de connectivité sur un test d'une heure 176
5.16 Topologie considérée pour valider la partition de sécurisation en environnement réel 177
5.17 Valeur TTL du drone Dr1 (ttl=62), Dr2 (ttl=63) et Dr3 (ttl=64) 178
5.18 Stabilité d'une route avec la présence d'un attaquant *blackhole* 180
5.19 Pourcentage des paquets de contrôle conduisant à un établissement de route venant de Dr2 et de l'attaquant mesuré sur l'interface du nœud Dr3 181
5.20 Répartition des valeurs délai de bout en bout au cours de la mission 182
5.21 Topologie de validation de l'attaque *wormhole* 184
5.22 Taux de création passant par les nœuds légitimes et les attaquants 185
5.23 Taux de livraison des paquets en présence d'une attaque *wormhole* 187

Annexes **200**

A.1 Envoi d'un paquet IP à travers le framework *Virtualmesh* 203
A.2 Capture d'écran du nouveau protocole expérimental en fonctionnement . . . 204
A.3 Traces de trois drones scannant une zone 205

B.1 Architecture logicielle d'AVISPA 207
B.2 Simulation d'intrus dans AVISPA 208

Liste des tableaux

1 Architecture de communication pour une flotte de drones 13

1.1 Différents termes employés pour désigner un système UAS 17

1.2 Différentes terminologies des réseaux ad hoc de drones 24

1.3 Comparaison des différentes architectures de communication 24

1.4 Différents types de flux applicatifs dans un réseau ad hoc de drones 25

1.5 Comparaison entre les réseaux UAANET et MANET 29

2 Sécurité dans un réseau ad hoc de drones 41

2.1 Synthèse des protocoles de routage sécurisé existants 68

2.2 Ensemble des mécanismes retenus 71

3 Méthodologie de développement d'un logiciel embarqué pour un système de drones 73

3.1 Classification des pannes dans la norme DO-178B 77

3.2 Différents scénarios d'utilisation selon la réglementation française 87

3.3 Apport des normes DO-178C et DO-331 89

3.4 Outils applicables dans le cadre de notre méthodologie 94

4 Protocole SUAP (Secure UAV Ad hoc routing Protocol) 96

4.1 Trafics générés 103

4.2 Résultats de connectivité sur un test de 3h 105

4.3 *Overhead* (au niveau de l'interface du drone 1) 107

4.4 Délais moyens et maximums (au niveau de l'interface du drone 1) 107

4.5 Synthèse des quelques propositions existantes contre l'attaque *wormhole* . . 118

4.6 Tableau de notation 124

4.7 Tableau de correspondance entre la distance géographique T et le nombre de sauts 126

4.8 Illustration des différents champs du paquet requête 128

4.9 Résultats obtenus par l'utilisation des différents *back-ends* AVISPA sur le protocole SUAP 141

5 Évaluation des performances du protocole SUAP par simulation et par expérimentation réelle 155

5.1 Spécification du drone DT18 158

5.2 Différents flux échangés lors du test en vol 160

5.3 Synthèse des résultats obtenus pour l'évaluation du protocole AODV en émulation 161

5.4 *Overhead* du protocole pour un test de 28 min 162

5.5 Stabilité de la route entre Dr2 et station 1 162

5.6 Délai de rétablissement de route après une déconnexion 165

5.7 Délai moyen entre station 1 et Dr2 166

5.8 Délai moyen pour l'acheminement des paquets de données utiles 168

5.9 Synthèse des résultats obtenus en émulation et en test réel 170

5.10 *Overhead* sur le test d'une heure 175

5.11 Différents flux échangés lors du test en vol 178

5.12 Taux de pertes (%) mesurés entre chaque nœud 179

5.13 *Overhead* sur le test de 16 minutes 179

5.14 Stabilité d'une route en présence d'un attaquant *blackhole* 179

5.15 Délai de rétablissement de route après une déconnexion 180

5.16 Délai de bout en bout pour les paquets de contrôle et les trafics temps réel 181

5.17 Synthèse des résultats obtenus pour l'évaluation en environnement réel du protocole AODV en présence d'un attaquant *blackhole* 183

5.18 Paramètres de l'évaluation pour la validation du mécanisme de détection de l'attaque *wormhole* 184

5.19 Trafics générés : 1=Dr1, 2=Dr2, 3=Dr3, 4=Dr4 185

5.20 Synthèse des données envoyées et réceptionnées entre la communication Dr4 vers Station sol en utilisant le protocole SUAP 186

5.21 Synthèse des données envoyées et réceptionnées entre la communication Dr4 vers Station sol en utilisant le protocole AODV (sans sécurité) 187

Annexes **200**

B.1 Comparaison des outils de vérification formelle 207

Introduction générale

Contexte et Problématique

Les drones, en anglais *Unmanned Aerial Vehicles* (UAV), sont des aéronefs sans pilote à bord. Ils sont constitués d'un autopilote qui leur permet d'exécuter des commandes envoyées depuis une station sol. Ils ont des degrés d'autonomie variables qui dépendent de la supervision du pilote au sol. Généralement, ils embarquent une charge utile pour capturer des informations dans leur environnement. Les drones ont d'abord été employés dans le domaine militaire pour réaliser des missions dangereuses pour un pilote à bord (par exemple, évolution en zone hostile). Leur adaptation dans le secteur des applications civiles est récente et se concrétise par des missions de surveillance, de cartographie ou encore de prise de vue photo (ou vidéo).

Les drones modernes peuvent avoir différentes tailles afin de leur permettre de réaliser différents types de missions. Dans cette thèse, nous nous référerons principalement aux mini-drones. Un mini-drone est équipé d'un ensemble de systèmes micro-électromécaniques qui inclue des microprocesseurs, des adaptateurs radio sans fil et des charges utiles généralement limitées en poids et en taille (car le volume disponible à bord de l'habitacle d'un drone est souvent restreint). Cette contrainte peut être un obstacle à la réalisation des missions de longue durée (par exemple, une surveillance aérienne à la suite d'une catastrophe naturelle sur une vaste zone). Une solution alternative à ce problème est le déploiement d'un système multi-drones permettant la mise en œuvre d'un réseau de communication coopératif entre les drones. La mise en place d'une telle structure de communication nécessite la collaboration des drones d'une même flotte. Cette coopération est rendue possible grâce à un algorithme de coordination fiable, qui échange en continu des trafics de signalisation. Plusieurs types d'architectures de communication peuvent être envisagés pour établir la communication d'un système multi-drones, nous les analyserons dans la prochaine section. Notre choix s'est tourné vers les réseaux ad hoc mobiles, car ils sont adaptés à l'environnement d'exécution d'une flotte de drones (grâce à la prise en compte, par exemple, de la mobilité des nœuds du réseau).

Les réseaux ad hoc sans fil sont composés de nœuds ayant la possibilité de communiquer de manière autonome par ondes radio. Les nœuds interagissent entre eux et peuvent coopérer pour échanger des messages. Un nœud peut initier la communication en envoyant des messages aux autres nœuds qui agissent en tant que relais. Dans ce cas, un relais permet à des nœuds se trouvant hors de portée radio de communiquer. Ces réseaux sont appelés ad hoc dans la mesure où ils ne nécessitent aucune infrastructure fixe. Le mode de fonctionnement ad hoc se distingue alors du mode infrastructure, dans lequel les nœuds du réseau communiquent entre eux à travers un point d'accès qui peut être relié à un réseau fixe ou à un réseau cellulaire.

Par ailleurs, les spécificités d'un réseau ad hoc sans fil sont à double tranchant, car, d'une part, elles permettent une mise en place peu coûteuse et rapide de réseaux de communi-

cation, et d'autre part, elles engendrent des difficultés lors de la conception de certains services tels que le routage, la qualité de service ou encore la sécurité. En effet, en l'absence d'une entité centrale, les nœuds doivent vérifier l'intégrité des messages échangés dans le réseau. Plusieurs raisons justifient le besoin en matière de sécurité d'un réseau ad hoc sans fil. Nous pouvons citer, par exemple, les fonctions sensibles des nœuds, comme le routage ou encore leur configuration. Il est aussi à mentionner l'existence d'attaques dans le réseau (certains exemples seront évoqués dans le chapitre 2). Il faut donc sécuriser les paquets de routage pour contribuer à la sécurité du système de drones. Afin de déployer une flotte de drones, il est donc nécessaire de proposer une architecture de communication permettant aux nœuds, non seulement de coopérer, mais aussi de sécuriser les messages véhiculés.

Note sur le déroulement de la thèse

Cette thèse [1] a été effectuée à l'École Nationale de l'Aviation civile (ENAC [2]) (un établissement public administratif dépendant de la DGAC), dans le laboratoire ENAC /équipe TELECOM/axe de recherche Resco (réseau de communication) et à l'entreprise Delair-Tech [3], une entreprise toulousaine spécialisée dans la conception de drones professionnels.

Architecture de communication d'une flotte de drones

L'architecture de communication d'une flotte de drones est définie comme un ensemble de règles et de mécanismes qui déterminent les modes d'échanges des informations relatives à la mission et à la flotte de drones. Plusieurs structures de réseaux sans fil sont envisageables pour établir une communication entre les différentes entités du système UAS.

Tout d'abord, il est possible de déployer un réseau à communication directe entre la station de contrôle et chaque drone du réseau. Dans cette architecture, chaque drone est connecté directement à la station sol avec un lien dédié. La station sol est considérée comme étant un nœud central et communique avec tous les drones simultanément.

Ce type d'architecture a plusieurs inconvénients :

— tout d'abord, il nécessite une quantité de bande passante proportionnelle au nombre de nœuds ;

— il engendre une latence (non négligeable) de transmission entre deux drones voisins, car les données doivent transiter vers la station sol avant d'être retransmises vers le destinataire ;

— des obstacles (par exemple, la présence d'une montagne) peuvent bloquer la communication entre la station sol et un drone.

1. Financement CIFRE
2. http ://www.enac.fr/
3. http ://www.delair-tech.com/

Un autre type de communication possible est le réseau de communication cellulaire. Il se base sur une topologie centralisée et consiste à diviser un territoire en plusieurs zones. Chaque zone est gérée par une station de base dont la tâche principale est de gérer la communication d'un groupe de nœuds. Cette structure de communication est la base des technologies de la téléphonie mobile, comme le GSM, GPRS, UMTS, LTE [Mis04] et des communications de données sans fil, comme le Wi-Fi et le Wimax [LC08]. Contrairement aux réseaux satellitaires (qui seront évoqués dans le paragraphe suivant), les réseaux cellulaires utilisent généralement des émetteurs de faible puissance. Ils pourraient donc être une solution qui permettrait, d'une part d'établir une communication entre les drones, grâce à l'infrastructure des opérateurs téléphoniques existante en profitant de leur portée et, d'autre part, de gérer les contraintes associées à la mobilité. Cependant, un frein à leur déploiement reste le coût de communication non négligeable, et difficilement amorti par une utilisation peu fréquente des drones.

Il est également possible de recourir à une communication par satellite pour faire communiquer deux nœuds géographiquement éloignés. Nous pouvons citer deux types de satellites : le satellite géostationnaire et le satellite orbital. Le satellite géostationnaire reste fixe par rapport à une référence géographique, en tournant autour de la terre à la même vitesse, tandis qu'un satellite orbital couvre une zone géographique variable. Ce type de communication peut être utilisé pour un dialogue entre une station sol et un drone. Nous pouvons envisager de l'utiliser également pour un système multi-drones dans lequel les communications entre drones passent par un satellite. Cependant, cette approche a un impact négatif sur les performances, notamment sur la latence engendrée par la transmission ou le coût d'exploitation d'un satellite. De plus, pour que cette approche fonctionne, il faut que les nœuds soient sur la ligne visée couverte par le satellite. Or, en cas de présence d'obstacles (par exemple, arbre ou montagne, etc.), le signal entre un drone et un satellite peut être bloqué.

Le dernier type de réseau de communication que l'on peut considérer est le réseau ad hoc mobile (MANET pour Mobile Ad hoc NETwork). Dans le cas d'une flotte de drones, ce réseau s'intitule « réseau ad hoc de drones » (UAANET pour UAV Ad hoc NETwork). Un réseau ad hoc mobile est un réseau sans fil et sans entité centrale (contrairement à un réseau de communication centralisé ou cellulaire). Il se base sur la capacité des nœuds à coopérer pour former un réseau entre eux. Chaque nœud relaye ainsi le message de l'émetteur jusqu'à la destination. Cette coordination permet aux nœuds de se déplacer librement, ce qui peut causer des changements fréquents de la topologie du réseau. Pour faire face à cette difficulté, le réseau doit utiliser des protocoles de communication fiables permettant de reconstruire une topologie communicante en tout temps. Les protocoles de routage en sont des exemples de ces protocoles. Ils ont pour rôle de trouver une route entre deux nœuds communicants et de s'adapter aux changements fréquents dans des topologies dynamiques.

Dans une flotte de drones, il est nécessaire de faire coopérer les drones entre eux pour une meilleure exécution de la mission. Cette coordination est réalisée par un échange régulier de messages dans un environnement mobile dynamique. Pour cela, il est judicieux de choisir le système de communication le mieux adapté aux caractéristiques des systèmes de mini-drones coopératifs.

Le système de flotte de drones est également caractérisé par une capacité de charge utile et par une capacité en énergie restreinte. En effet, dans la plupart des cas, un mini-drone ne peut pas supporter trop de charge, car les équipements à bord conditionnent son altitude et son endurance.

En outre, les drones sont aussi connus pour être utilisés dans des zones dangereuses où les obstacles sont nombreux et dans lesquelles, il est difficile d'avoir une couverture cellulaire ou d'établir une communication directe entre une station et un drone.

Par conséquent, pour toutes ces raisons, les réseaux ad hoc sans fil semblent être les mieux adaptés au réseau de communication d'une flotte de drones.

Sécurité de communication

Une autre caractéristique d'un réseau ad hoc de drones, commune au réseau ad hoc sans fil, est sa vulnérabilité aux attaques. Celles-ci trouvent leur origine dans les points de vulnérabilités suivants :

— Les communications radio sans fil sont sujettes à des interférences et la nature ouverte du médium pose la question de la confidentialité des données, lors d'écoutes illicites. Typiquement, avec une antenne commerciale configurée sur une certaine fréquence, un attaquant peut écouter les communications sans y avoir été invité.

— La communication multisaut et sans fil est justifiée par le fait que la portée radio des drones est limitée. Une communication multisaut est donc nécessaire. Toutefois, elle peut engendrer des menaces de sécurité notamment des attaques contre la construction et la maintenance des routes ou des attaques d'altération de données à travers l'insertion, la modification ou la suppression des paquets de routage.

— La vulnérabilité inhérente aux réseaux UAANET est aussi due à l'absence d'une entité de communication centrale, car plusieurs types d'attaques peuvent être exécutés ; par exemple, une attaque d'usurpation d'identité, qui consiste à rajouter un nœud malveillant à l'ensemble des nœuds d'un réseau.

D'autres types d'attaques plus spécifiques aux réseaux MANET, comme l'attaque *blackhole* [TS07] ou l'attaque *wormhole* [MNS08], sont également possibles. L'attaque *blackhole* consiste à intercepter les paquets de données en interférant avec la phase de découverte de route par la publication de faux paquets. Une autre attaque connue sous le nom de

wormhole consiste à créer un tunnel virtuel entre deux attaquants. Chacun des deux attaquants va se rapprocher des deux nœuds cibles afin d'intercepter le trafic. L'objectif est d'utiliser le tunnel pour véhiculer telles quelles des informations de routage et faire croire ainsi à deux nœuds géographiquement éloignés qu'ils sont voisins. Cela signifie que les attaquants n'auront pas besoin de modifier le paquet, ce qui rend cette attaque résistante aux mécanismes d'authentification. À la suite de cette attaque, les paquets de données vont transiter dans le tunnel, ce qui permettra aux attaquants de les intercepter. L'efficacité des méthodes d'interception de l'attaque *wormhole* justifie la nécessité de trouver un moyen de la détecter.

Par ailleurs, il est important de spécifier que d'autres types d'attaques peuvent également être réalisés. Par exemple, sur la couche liaison, un nœud peut créer un déni de service en saturant le médium par des trames de contrôle et empêcher ainsi les autres nœuds de communiquer. Des attaques liées à la couche MAC IEEE 802.11 [OP05], qui exploitent certains aspects du protocole, peuvent également créer un déni de service. En ce qui concerne la couche application, les attaques à ce niveau sont généralement communes à tous les types de réseau, mais leur méthode est spécifique à l'application visée. Dans le chapitre 2, une analyse des risques des attaques existantes est présentée.

Dans le cas d'un réseau UAANET, les solutions de sécurité appliquée aux réseaux filaires ne sont pas appropriées en raison des spécificités du réseau ad hoc mobile, et de l'absence dans ce d'infrastructure fixe.

Pour sécuriser les réseaux UAANET, il est possible d'apporter des briques de sécurité sur toutes les couches du modèle TCP/IP. Les attaques associées à la couche physique (comme généralement : l'écoute illicite, l'interférence et l'attaque *jamming*) sont liées à la technique de transmission utilisée. Elles sont généralement difficiles à réaliser dans un réseau à forte mobilité, comme le réseau UAANET, car l'attaquant doit rester à la portée du nœud cible pour rompre la communication. Puisque les drones ont généralement des mobilités importantes, nous avons décidé d'écarter les attaques liées à la couche physique. Il en est de même pour la couche liaison, l'objectif des attaques à ce niveau étant généralement de lancer une attaque par déni de service selon différentes approches. Certaines solutions de détection d'attaque par déni de service ont été proposées dans la littérature [ZWN04]. La plupart d'entre elles se basent sur des mécanismes d'évaluation de confiance entre voisins pour détecter le nœud malveillant. Ces approches d'évaluation de confiance soulèvent l'hypothèse selon laquelle un protocole de routage fiable, assurant l'authentification et l'intégrité des paquets de routage, existerait dans le réseau. Ainsi, il nous parait judicieux d'apporter une brique de sécurité au niveau réseau, dans un premier temps, avant de sécuriser la couche liaison. Dans le chapitre 2, nous détaillerons différents types d'attaques qui peuvent intervenir sur chaque couche, mais nous considérerons uniquement la couche réseau dans notre solution.

Généralement, pour sécuriser la couche réseau, il convient d'assurer différents services de sécurité, comme l'authentification, l'intégrité, la confidentialité, la disponibilité et la non-répudiation. Dans notre cas, nous considérerons que l'authentification et l'intégrité des trafics de signalisation, car notre objectif est de proposer un protocole de routage sécurisé qui puisse proposer des routes précises et fiables en présence d'attaquants. La confidentialité des paquets de routage n'est pas nécessaire, car les trafics de signalisation échangés ne contiennent pas d'informations secrètes. Ils sont généralement envoyés en temps réel et, ainsi, leur réinjection dans le réseau sera détectée par le système d'horodatage existant dans le protocole de routage (qui peut être protégé par un algorithme d'authentification des messages).

Besoin en certification d'un système UAS

Un autre aspect complémentaire de notre travail porte sur la certification du système UAS. En effet, pour permettre à un drone de voler dans l'espace aérien national, il est nécessaire de certifier le système UAS qui l'englobe. Cette certification passe par un processus de validation rigoureux des différents sous-systèmes qui le composent. En France, l'utilisation des drones est encadrée par l'arrêté du 17 décembre 2015 relatif à la conception et à l'utilisation des aéronefs civils qui circulent sans personne à bord [Fra15]. Cette réglementation se base sur une classification de la zone d'évolution en quatre scénarios :

— Scénario 1 : vols en vue directe du télépilote se déroulant hors zone peuplée, à une distance horizontale inférieure à 200 m du télépilote.

— Scénario 2 : vols hors zone peuplée, à une distance horizontale maximale de rayon 1 km du télépilote et de hauteur inférieure à 50 m au-dessus du sol ou d'obstacles artificiels, sans aucune personne au sol dans la zone d'évolution. Extension à 150 m de hauteur pour les aéronefs de moins de 2 kg.

— Scénario 3 : vols en agglomération ou à proximité d'un rassemblement de personnes ou d'animaux, en vue directe du télépilote, à une distance horizontale maximale de 100 m du télépilote.

— Scénario 4 : vols hors vue directe, hors zone peuplée et ne répondant pas aux critères du scénario S 2.

Il s'agit ainsi pour chaque scénario de démontrer le bon fonctionnement de tous les éléments de sécurité exigés.

Delair-Tech a été la première entreprise à obtenir en octobre 2012 une certification de la DGAC (Direction Générale de l'Aviation Civile) permettant de mettre en opération des drones hors de portée de vue du pilote. L'ajout de l'architecture de communication sécurisée développée dans cette thèse doit s'intégrer au système déjà mis en place. Il est donc nécessaire de passer par un processus de certification complémentaire à celui mené en

2012. Ainsi, pour permettre à notre système d'être certifié, nous intéresserons également à la validation du système final, en assurant la sécurité fonctionnelle du système par des sous-ensembles de recommandations de la norme DO-178 [HB07]. Ici, il s'agit de la certification logicielle (notre algorithme de routage sécurisé) embarquée dans le drone. Dans le monde de l'informatique embarquée, le concept de sécurité fonctionnelle désigne l'évaluation et la prévision des risques de dysfonctionnement. Il s'agit alors de prouver à un organisme de certification que nous respectons un processus de développement conforme aux normes du secteur des aéronefs.

En outre, il est important de savoir qu'il n'existe aujourd'hui aucun standard lié à la conception des logiciels pour les drones. Toutefois, nous pouvons suivre les recommandations déjà bien avancées dans les secteurs aéronautiques. En effet, les logiciels avioniques sont soumis à des contraintes de certification fortes, permettant de constater que l'on ne déplore à ce jour aucun crash d'avion lié à une défaillance logicielle. Le standard de base utilisé par l'industrie (au sens large) est la norme IEC 61508 [Bel06] qui se décline en plusieurs versions selon le secteur. Pour l'aéronautique, il s'agit du standard DO-178 qui regroupe un ensemble de recommandations relatives aux différents aspects de la certification des logiciels avioniques : la planification de développement logiciel, de vérification, de gestion de configuration, de mise en place de règles et de codage, de conception, de validation des résultats de tests et de leur couverture, etc. Dans la norme DO-178, la procédure de certification d'un système consiste à vérifier que le concepteur maîtrise l'ensemble du processus de développement logiciel.

En effet, les organismes de certification tiennent à vérifier que le développement du logiciel a été effectué de manière rigoureuse, en utilisant un enchaînement de tâches bien déterminées. Il est donc nécessaire, pour un concepteur de logiciels dédiés aux aéronefs, de maîtriser son programme de développement et d'être capable de justifier chaque ligne du code final produit, chaque variable et chaque opérateur logique. Il faut donc être en mesure de fournir des documentations, des preuves formelles et des tests supplémentaires à l'organisme de certification.

Le sous-ensemble de recommandations considéré dans cette thèse est la modélisation haut niveau de la spécification du cahier des charges, qui permet dans un deuxième temps de pouvoir générer du code à partir de ce modèle. Ensuite, des outils de vérification formelle sont utilisés pour vérifier le modèle et comparer la conformité entre le modèle et le code généré. Pour réaliser cette tâche, nous avons utilisé les outils Matlab Simulink et Stateflow qui permettent de modéliser un système et de générer du code automatiquement à partir des modèles haut niveau. Ils sont associés à des outils de vérification formelle, propriétaires de Matlab, qui permettent d'automatiser les jeux de tests unitaires, la documentation et la vérification de correspondance entre le modèle dérivé du cahier des charges et le code source généré.

Contributions

Le réseau ad hoc mobile est une solution adéquate pour la communication des drones entre eux et avec la station sol, tout en tenant compte de leur liberté de mouvement. Dans l'architecture ad hoc, chaque aéronef aura une tâche bien spécifique qui sera de relier l'information de charge utile ou de générer un flux temps réel vers la station sol. Les drones échangeront une variété de types de messages selon les besoins. En raison de la spécificité d'un réseau ad hoc sans fil, le réseau UAANET est intrinsèquement vulnérable et plusieurs types d'attaques sophistiquées peuvent être exécutées à son encontre. Sans un bloc de sécurité adéquat, les paquets de routage peuvent être modifiés volontairement par un attaquant et la topologie du réseau peut ainsi s'en trouver modifiée. De faux paquets peuvent également être générés et distribués dans le réseau, créant ainsi des distorsions et diminuant la performance du réseau. Une attaque *wormhole*, qui consiste à créer une route fictive entre deux nœuds distants, peut également être exécutée. Cette attaque est généralement difficile à résoudre dans la littérature des réseaux MANET. Il apparait donc que l'objectif final de cette thèse réside dans la création d'une nouvelle architecture de communication permettant d'une part de proposer une route fiable pour couvrir les besoins en matière de qualité de service des informations en temps réel et, d'autre part, de sécuriser cette recherche de route afin de conserver l'intégrité du réseau. Nous proposons dans cette thèse une architecture de communication sécurisée conçue pour répondre à ces exigences. Cette architecture est adaptée aux réseaux de flottes de drones coopératifs. Elle est étudiée pour la réalisation de ce genre de système comme le système UAS de Delair-Tech. En effet, elle a été implémentée sur des drones DT-18 et testée de façon extensive dans un environnement réel.

Pour mener à bien notre étude, il a été nécessaire, dans un premier temps, de créer un outil hybride d'émulation et de simulation, permettant, d'évaluer, dans un contexte d'environnement UAANET la performance des protocoles de routage existants (pour choisir un protocole de départ) ainsi que celle du protocole proposé dans cette thèse. En effet, afin de sélectionner un protocole de départ pour notre architecture réseau de communication, il était nécessaire de considérer la spécificité du réseau UAANET et de la formaliser dans un outil de test. Une raison supplémentaire justifiant la création de cet outil est la possibilité d'avoir une seule version de code source, qui puisse à la fois être testée en local et qui fonctionne avec le système embarqué des drones DT-18 utilisés. Cet outil, développé au cours de cette thèse, permet d'interfacer des machines virtuelles Linux dans le simulateur OMNET++. Grâce à cet outil, nous avons pu justifier le choix d'AODV en tant que squelette de notre protocole de routage sécurisé.

Dès lors, la première contribution proposée dans ce travail de thèse concerne la conception orientée modèle d'un protocole de routage pour le réseau UAANET. La difficulté à associer le développement de système aéronef embarqué et la phase de certification de

ce système nous ont conduits à un travail approfondi sur le processus de développement d'un logiciel embarqué dans une flotte de drones. Cette méthodologie de développement et de prototypage rapide prend en compte les contraintes de sûreté et de sécurité associées au système à mettre en œuvre dès l'établissement du cahier des charges. Elle est basée sur la modélisation des fonctionnalités et la transformation automatisée des modèles en code source. Les modèles présentent en effet des avantages non négligeables en termes de rapidité et de réutilisabilité du développement, ainsi qu'en vérification et en validation des systèmes modélisés. La transformation automatisée contribue à garantir la sécurité du code généré par la comparaison de conformité entre le code généré et le modèle de haut niveau. Les codes générés sont exécutés dans des systèmes d'exploitation critiques à travers l'outil d'émulation et de simulation. Cette méthodologie a été créée pour permettre une implémentation modulaire et réutilisable. Elle a donc été utilisée pour développer la partition de routage de notre protocole qui a ensuite été évalué en environnement réel.

La deuxième contribution est la proposition d'un algorithme de routage sécurisé pour le réseau UAANET. Ce protocole de routage, intitulé SUAP (Secure UAV Ad hoc routing Protocol), assure l'authentification ainsi l'intégrité des messages, et protège contre l'attaque *wormhole*. L'authentification est assurée par une signature numérique et l'intégrité est protégée par un algorithme de hachage à sens unique. Ces propriétés ont été ajoutées à un algorithme de *geographical leashes* pour créer une corrélation entre le nombre de sauts et la distance relative entre deux nœuds. Cette approche assure ainsi la sécurité des paquets de découverte des voisins pour protéger la connaissance des différents voisins, et pour détecter simultanément un tunnel *wormhole*. Par la suite, nous nous appuyons sur l'information de voisinage obtenue par le mécanisme s'appuyant sur le *geographical leashes* pour décider de la confiance attribuée aux nœuds voisins. Ainsi, dans un deuxième mécanisme contre l'attaque *wormhole*, l'identité de chaque nœud est considérée dans le calcul d'une chaîne de hachage imbriquée afin de protéger les paquets de découverte de route. Ces mécanismes ont été conçus avec une approche orientée modèle. Ce protocole de routage SUAP a ensuite été formellement vérifié par des outils tels que AVISPA pour la partie sécurité, et par des outils inhérents à Matlab Simulnik & Stateflow (Simulink design verifier, Model coverage) pour la sécurité de conception (c'est-à-dire les exigences non fonctionnelles).

Il est important de remarquer que les différentes partitions du protocole SUAP ont été validées fonctionnellement par des études de performances réalisées en environnement réel. Ce travail d'évaluation des performances représente également une contribution importante (la troisième), car peu de travaux de recherche font le choix de considérer une flotte de drones, comme contexte d'expérimentation réel, en raison des contraintes techniques matérielles et logistiques liées à ce choix.

Structure du mémoire

Ce mémoire de thèse est organisé en six chapitres principaux.

Le premier chapitre présente les différentes architectures de communication possibles pour une flotte de drones. Dans ce chapitre, l'intérêt des réseaux ad hoc de drones sera mis en avant (ils seront nommés tout au long de ce manuscrit par le terme UAANET), et leurs caractéristiques seront présentées. Par la suite, nous introduirons les protocoles de communication nécessaires dans les réseaux UAANET. Ce chapitre comprend ainsi une étude synthétique des protocoles de communication existants afin de pouvoir identifier ceux qui répondent les mieux à nos besoins.

Le deuxième chapitre aborde les besoins en termes de sécurité d'un réseau ad hoc de drones. Il présente les différentes vulnérabilités de ce type de réseau et les services de sécurité qu'il est nécessaire d'assurer. Ce chapitre fait également une synthèse des protocoles de routage sécurisé existants pour les réseaux MANET, ce qui permettra de dégager les principales fonctionnalités que nous avons retenues afin de justifier la proposition de notre protocole SUAP. Ainsi, à l'issue de ce chapitre, l'ensemble des besoins en matière de sécurité pour notre architecture de communication sera formalisé.

Le troisième chapitre traite de l'aspect méthodologique permettant d'atteindre l'objectif de certification du système final SUAP. Ce chapitre décrit donc la méthodologie de prototypage rapide de notre système UAS, contribuant à la validation de notre système final. En effet, faire évoluer des flottes de drones et des avions civils dans le même espace aérien nécessite la garantie de l'innocuité et de l'immunité des différents éléments physiques et logiciels composant le système UAS. La méthodologie utilisée dans cette thèse permet, d'une part, de contribuer à la sûreté du système final et, d'autre part de réduire le temps et les coûts de développement. Nous présenterons donc dans ce chapitre les différentes étapes abstraites de la méthodologie et la liste des outils nécessaires à sa mise en œuvre.

Par la suite, le chapitre 4 détaillera le protocole de routage SUAP sécurisé que nous proposerons dans le cadre de cette thèse. Nous présenterons les différents modules de ce protocole et l'architecture logicielle de sa mise en œuvre. Dans la première section, nous détaillerons la méthode de sélection d'un protocole de routage initial, qui a permis, par une étude comparative d'évaluation de performance des principaux protocoles de routage du domaine ad hoc, d'identifier le protocole AODV comme le protocole le plus adéquat pour l'environnement UAANET. Seront présentés dans la deuxième section, les mécanismes de sécurité retenus pour constituer le protocole et, dans la troisième section, la vérification des propriétés de sécurité faite avec AVISPA. Enfin, dans la dernière section sera étudiée la mise en œuvre du protocole SUAP avec l'approche MDD (Model Driven Development).

Le chapitre 5 sera consacré à la validation des différentes partitions du protocole SUAP : routage, mécanisme d'authentification et enfin la partition capable de contrer une attaque

wormhole. Les plans de tests et de validation y seront présentés ainsi que les résultats des mesures expérimentales obtenus. Dans ce chapitre, les études en environnement réel et l'étude par émulation seront abordées. Ces expérimentations permettent d'évaluer et de valider le fonctionnement de notre proposition dans son environnement de déploiement réel.

Le dernier chapitre conclut ce mémoire. Il résume le travail effectué ainsi que les contributions. Il est suivi d'un ensemble de perspectives permettant de compléter certaines problématiques restées ouvertes ou qui n'auront pu être traitées au cours de cette thèse.

Chapitre 1

Architecture de communication pour une flotte de drones

L'architecture de communication d'une flotte de drones définit les règles d'échanges entre les différentes entités (consitutées par les drones et la station sol) dans le réseau. Différents types d'architecture peuvent être envisagés en fonction des caractéristiques des nœuds et du cahier des charges de la mission. Parmi ces différentes architectures, l'architecture ad hoc sans fil sera mise en avant, car elle assure flexibilité et robustesse aux communications entre les drones. Un réseau ad hoc sans fil est caractérisé par une architecture qui s'adapte au nombre des nœuds, à leur position géographique et à leur mobilité. Il permet aux différents nœuds de coopérer sans l'utilisation d'une infrastructure réseau fixe préexistante et/ou centralisée. Appliqué aux flottes de drones, ce réseau s'intitule UAANET pour Uav Ad hoc NETwork. Il partage des caractéristiques similaires aux réseaux Mobile Ad hoc Network (MANET), mais présente aussi des particularités en matière de degré de connectivité et de mobilité. Ces caractéristiques sont à prendre en compte dans la définition et la conception d'un protocole de communication dans un réseau UAANET.

Contents

1.1 Introduction 15
1.2 Systèmes aéronautiques coopératifs sans pilote 15
1.2.1 Description fonctionnelle d'un drone 15
1.2.2 Unmanned Aircraft/Aerial Systems 17
1.2.3 Drones dans des opérations civiles 18
1.2.4 Flotte de drones 19
1.3 Architectures de communication possibles pour une flotte de drones 20
1.3.1 Architecture de communication centralisée 20
1.3.2 Architecture de communication par satellite 21
1.3.3 Architecture de communication réseau cellulaire (Réseau semi-centralisé) 22
1.3.4 Architecture de communication ad hoc 22
1.3.5 Réseau ad hoc de drones 25
1.3.6 Projets de réseau ad hoc de drones 29
1.4 Protocoles de communication d'un réseau ad hoc de drones . . 31
1.4.1 Liaison de données (*Medium Access Control*) 31
1.4.2 Niveau réseau : routage dans les réseaux UAANET 32
1.5 Discussions et Conclusions 39

1.1 Introduction

Une architecture de communication définit l'ensemble des entités nécessaires à la communication ainsi que les règles dictant les échanges entre elles. Ces ensembles de règles permettent la mise en place d'un réseau d'interconnexion et la mise en œuvre des services réseau (par exemple, la mise en place d'un routage multicast [1]). Dans le cas d'une flotte de drones, la définition et la mise en place d'une architecture de communication permettent de gérer l'échange des flux d'informations entre la station de contrôle et les drones. Une flotte de drones peut être définie comme un ensemble de drones coopératifs qui partage les tâches entre les drones de la flotte pour réaliser des missions complexes (par exemple, dans le but d'étendre la portée de la communication en relayant les données entre les drones). Différentes catégories de réseaux de communication peuvent être envisagées pour établir la communication au sein d'une flotte de drones. Dans ce chapitre, nous étudierons ces différentes architectures et présenterons leurs avantages et leurs inconvénients. Nous mettrons en évidence les avantages d'un réseau ad hoc de drones (UAANET) pour assurer la communication au sein d'une flotte de drones.

Dans ce chapitre, une description succincte d'un système aéronautique coopératif [2] sans pilote sera faite en premier lieu. Ensuite, les différentes architectures de communication, susceptibles d'être retenues pour la mise en œuvre d'une flotte de drones, seront analysées. Nous nous appuierons sur chacun de leurs avantages et inconvénients pour conclure sur l'architecture adéquate. Par la suite, l'accent sera mis sur l'architecture de communication ad hoc, en analysant son application pour une flotte de drones et les différents protocoles de communication nécessaires à son fonctionnement.

1.2 Systèmes aéronautiques coopératifs sans pilote

Cette section est consacrée aux systèmes aéronautiques sans pilote constituant le réseau de communication.

1.2.1 Description fonctionnelle d'un drone

Un drone désigne un aéronef sans pilote à bord, muni d'un autopilote [3] embarqué et pouvant être télécommandé à distance depuis une station sol. Dans la littérature, deux termes sont habituellement employés : UAV et Remotely Piloted Aircraft System (RPAS). A la différence de l'UAV, le RPAS requiert la présence d'un télépilote actif au sol mais pas

1. Le multicast est une technique de diffusion de messages à partir d'une source unique vers un groupe bien défini de récepteurs
2. Traduit de l'anglais *Unmanned Aerial System*
3. L'autopilote est un module physique et logiciel permettant une assistance au pilotage.

obligatoirement d'un autopilote à bord. Le terme UAV, quant à lui, désigne un drone possédant à bord un autopilote actif qui le pilote.

Les drones peuvent remplir diverses missions, historiquement dans le cadre d'applications militaires (surveillance d'une cible, renseignement, etc.), mais aussi, récemment, dans le cadre d'applications civiles (audiovisuel, industrie, agriculture). Dans ce manuscrit, nous nous intéressons exclusivement aux drones du domaine civil.

Un drone civil dispose de divers capteurs et embarque une charge utile. Ces éléments participent à son autonomie et lui permettent d'exécuter diverses applications. Par exemple, dans le domaine de l'audiovisuel, la réalisation d'un reportage a été pendant longtemps executée par des avions ou des hélicoptères pilotés. Non seulement ces missions peuvent être dangereuses et inadaptées aux pilotes, mais peuvent aussi engendrer des coûts conséquents. Les drones civils réalisent actuellement ces tâches en s'adaptant aux différents types et conditions de missions.

Il existe deux types de drones civils, ce sont les voilures fixes et les multirotors (illustrés par les figures 1.1 et 1.2).

— Les multirotors peuvent être des quadricoptères, hexacoptères, octocoptères ou décacoptères. Ce type de drones est souvent utilisé pour des vols stationnaires ou à très faible vitesse, ce qui les rend particulièrement adaptés à la prise de vue aérienne, photo ou vidéo ;

— les drones à voilure fixe, quant à eux, permettent de couvrir de longues distances ou d'atteindre de hautes altitudes, ils sont plutôt dédiés aux applications topographiques sur de grandes surfaces ou de grands linéaires.

FIGURE 1.1 : Exemple de drone à voilure fixe

FIGURE 1.2 : Exemple de drone multirotor

Les drones reçoivent les données de contrôle et de commande (qu'on appelle trafic C2 (Contrôle et Commande)) depuis la station sol à une fréquence déterminée. Ils renvoient aussi vers la station sol les informations de configuration concernant les conditions de vol (position, vitesse, etc.) et les données acquises par la charge utile. Ces données permettent à l'opérateur au sol de surveiller le vol et d'intervenir potentiellement sur le drone en lui envoyant des commandes.

1.2.2 Unmanned Aircraft/Aerial Systems

Le *Unmanned Aircraft/Aerial System* (UAS) est un terme avancé par l'administration fédérale de l'aviation américaine, « Federal Aviation Admnistration » (FAA), pour regrouper les différentes dénominations indiquées dans le tableau 1.1, l'objectif étant de les unifier afin d'utiliser le principe de système soit utilisé. Dans ce cas, le UAS est composé de drones, de différentes liaisons de communication et de transfert de données, d'une ou plusieurs stations sol et de systèmes additionnels sécurisant la mission. Ces derniers sont ajoutés pour fiabiliser la mission et répondre aux réglementations et aux exigences de certification. Il est à noter que la FAA ainsi que l'agence européenne de la sécurité aérienne (EASA), à travers la Direction Générale de l'Aviation Civile (DGAC), préconisent le contrôle permanent du système UAS par un système au sol [DVP08], restriction imposée pour le déploiement d'une flotte de drones et qui sera évoquée dans le chapitre 5. Afin d'en tenir compte, il a été nécessaire d'utiliser une station sol dédiée pour piloter chaque drone de la flotte.

Nom	**Acronyme**
UAS	Unmanned Aircraft System
UAV System	Unmanned Aerial Vehicle System
RPAS	Remotely Piloted Aircraft System
RPV System	Remotely Piloted Vehicle System

Tableau 1.1 : Différents termes employés pour désigner un système UAS

1.2.2.1 Charge utile

La charge utile d'un drone est l'ensemble des systèmes embarqués qui lui permettent de réaliser sa mission, par exemple, un appareil photo, une caméra vidéo, une caméra multi spéctrale, une caméra thermique, des sondes ou tout autre type de capteurs (mésurant, entre autres, des paramètres dans l'air, tels que la température, la pression ou encore le niveau de pollution). Grâce à sa charge utile, un système de drones peut être utilisé pour inspecter une zone donnée, la cartographie d'une zone déterminée, des relevés thermiques sur des bâtiments d'usine, etc.

1.2.2.2 Station sol

La station sol (illustrée par la figure 1.3) est un ensemble d'entités physiques et de logiciel qui permettent de contrôler le mouvement des drones. Selon le type de station utilisé, elle peut être munie d'une interface homme-machine qui permet à l'opérateur au sol de surveiller en temps réel la position d'un drone à l'aide d'une carte topographique sur laquelle l'itinéraire du drone est superposé. Il peut également configurer l'altitude et les

FIGURE 1.3 : Illustration d'une station sol déployée sur le terrain

paramètres de la charge utile.

Grâce à une liaison en temps réel entre la station sol et le drone, le logiciel de supervision (appelé souvent logiciel Ground Control Station (GCS)) permet de visualiser la vidéo capturée par le drone ou encore les photos prises. Toutes les données traitées par ce logiciel sont géoréférencées et peuvent être ainsi enregistrées ou partagées avec d'autres systèmes d'informations géographiques. La station sol dispose également d'équipements de télécommunication en radiofréquence, qui représentent la partie physique caractérisée par l'émetteur, et le récepteur et leurs antennes associées.

Il est important de souligner que la station communique avec les drones à travers des liaisons de commande et contrôle sur un lien de communication montant depuis le sol vers les drones. Dans l'autre sens, la station sol récupère un panel d'informations télémétriques et vidéos sur un lien de communication descendant.

1.2.3 Drones dans des opérations civiles

Généralement, les applications civiles des drones peuvent être divisées en trois catégories : l'audiovisuel, l'industrie et l'agriculture.

- Domaine de l'audiovisuel : il regroupe la plupart des professionnels de l'industrie du drone. Les utilisations sont les suivantes : la photographie aérienne, la cinématographie, la réalisation de reportages sur une zone difficile d'accès.
- Domaine de l'industrie : certains scénarios d'applications ont été réalisés, tels que :
  - la télésurveillance de pipelines pétroliers [HZSS05] ;
  - la télésurveillance des voies ferroviaires [MB15] ;
  - la cartographie, la topographie ;

— l'inspection détaillée (sites industriels, bâtiments, ouvrages d'art).

- Agriculture : dans le domaine de l'expérimentation agricole, les drones ont un potentiel de développement important, car ils permettent de récupérer de manière économique des données essentielles à l'échelle des microparcelles. En effet, cette échelle rend possible l'extraction des pixels de photo pour en permettre une analyse précise et recueillir ainsi l'information de l'image brute issue d'un appareil photo.

Un exemple d'utilisation de drones pour une mission de surveillance est illustré par la figure 1.4.

FIGURE 1.4 : Exemple d'utilisation d'un drone pour une application de surveillance électrique réalisée par l'entreprise Delair-Tech

1.2.4 Flotte de drones

Nous avons évoqué depuis le début du chapitre le fonctionnement d'un système de mini-drones et les différents types d'applications civiles réalisables. Il a été évoqué que l'utilisation de ces mini-drones apportait une plus-value qui se manifeste en termes de performances et de productivité. Néanmoins, l'aptitude d'un système muni d'un seul drone est limitée dans l'espace et dans le temps [BS T13]. Donc, pour une mission de plusieurs heures et qui nécessite de parcourir une zone large, l'utilisation d'un seul drone peut être limitée en matière d'autonomie de batterie et de portée (notamment en cas de présence d'obstacless). La collaboration de plusieurs drones est une solution qui offre plusieurs avantages :

— Coût : le coût d'acquisition et de maintenance des mini-drones mis en jeu dans une flotte de drones peut être moindre que l'utilisation d'un seul drone de grande taille.

— Mise à l'échelle : l'usage d'un seul drone ne permet de couvrir qu'une zone limitée. En effet, l'étendue de la mission dépend de la batterie et de la présence d'obstacless (par exemple, une montagne). Avec une flotte de drones, il est possible de créer un relais de drones pour contourner les éventuels obstacles et agrandir la zone de couverture

de l'opération. Cette fonction sera développée dans ce chapitre, avec le concept du réseau ad hoc de drones, à l'origine d'un réseau de communication autonome.

— Survie de la mission : la mission peut être interrompue en cas de panne d'un élément physique ou logiciel du système UAS. Toutefois, avec une flotte de drones, l'opération peut se poursuivre en cas de panne d'un des drones ou en cas d'épuisement de la batterie en reconfigurant les ressources de la flotte.

— Latence : il a été montré dans [YCBE10] que la densité de nœuds est proportionnelle au temps d'exécution de la mission. Plus le nombre de nœuds dans le réseau est élévé, plus le temps d'exécution de la mission est optimisé.

1.3 Architectures de communication possibles pour une flotte de drones

1.3.1 Architecture de communication centralisée

Une architecture de communication centralisée de flotte de drones [FB09] est caractérisée par un lien sans fil direct entre un nœud centralisé (par exemple, la station sol) et les drones aux alentours. Dans cette architecture, chaque drone est directement connecté à la station sol pour transmettre les données de la charge utile et pour recevoir le flux de commande et de contrôle. Les drones ne sont pas directement connectés entre eux, ce qui nécessite d'envoyer les informations entre drones voisins en passant par la station sol. Dans ce cas, la station sol agit comme un nœud relais. Une illustration de cette architecture est montrée par la figure 1.5.a.

Cette architecture comporte plusieurs inconvénients :

— La taille de la bande passante et le débit dépendra du nombre de drones dans le réseau, puisque la station sol communique avec chacun des drones. En conséquence, pour supporter une forte densité du réseau, il convient de fournir une quantité suffisante de bande passante qui peut être importante ;

— La latence de transfert d'un paquet de données entre deux drones voisins peut être longue en raison du relais obligé par la station sol [LZL13] ;

— En cas de présence d'un obstacle entre un drone et la station sol (montagne ou bâtiment, par exemple), le signal peut être bloqué et empêcher ainsi les trafics C2 d'être exécutés par les drones. En conséquence, les nœuds ne peuvent s'éloigner que d'une distance maximale de la station sol, limitant ainsi la distance d'opération des drones. Des équipements radio avancés peuvent aussi être déployés, permettant ainsi d'étendre la portée de communication, mais ce genre de système peut générer une forte puissance de transmission [CHKV06], ce qui est interdit d'un point de vue légal [FW12].

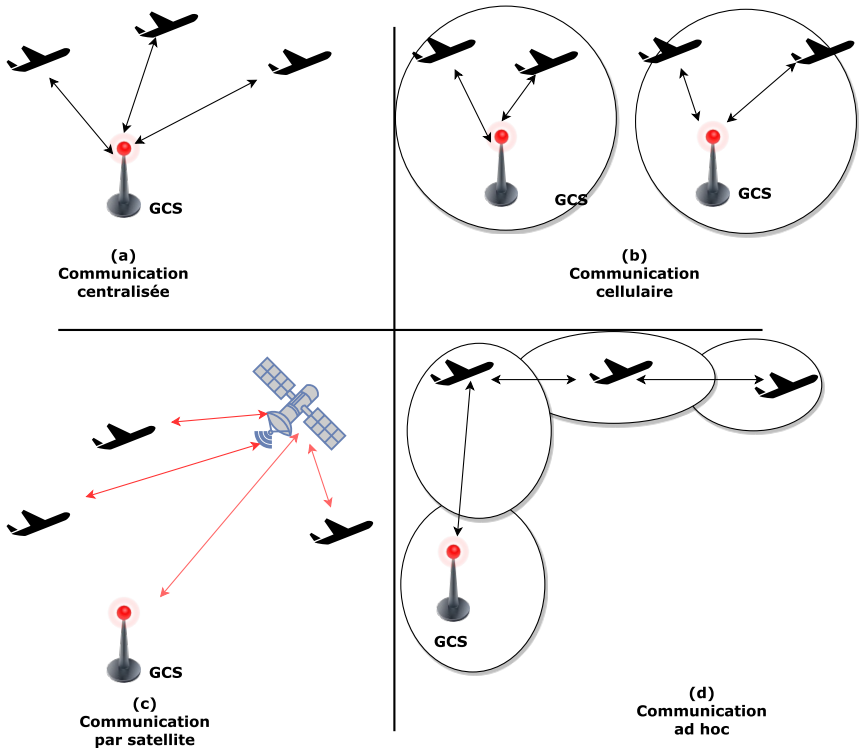

FIGURE 1.5 : Principales architectures de communication permettant la communication pour les flottes de mini-drones

— La présence d'une entité centrale dans le réseau le rend également vulnérable, car, si la station sol est hors service suite à une attaque, la disponibilité des drones n'est plus garantie.

1.3.2 Architecture de communication par satellite

Un autre type d'architecture de communication centralisée, envisageable pour établir une communication entre les différents drones, est le déploiement d'un satellite de communication. Dans cette configuration, le satellite fonctionne comme un relais de communication [FB09]. Ses antennes de réception reçoivent les signaux émis depuis la station sol ; ces signaux sont par la suite transposés en fréquence et amplifiés avant d'être retransmis vers les drones. Il existe deux types de satellites utilisables : le satellite géostationnaire et le satellite orbital. Le satellite géostationnaire est situé dans le plan équatorial. Il tourne à la même vitesse et dans le même sens que la terre ; sa trajectoire est ainsi fixe au-dessus d'un point au sol. Le second est un satellite qui couvre des zones géographiques différentes.

L'utilisation des satellites présente l'avantage d'assurer une couverture plus efficace que celle d'une communication centralisée. Cela permet une amélioration d'interconnexion du réseau de communication des drones, indépendamment de leurs trajectoires. Toutefois,

cette connectivité nécessite toujours un routage vers un système centralisé qui est, ici, le satellite. Compte tenu du caractère temps réel des trafics applicatifs échangés dans le cas d'une flotte de drones, se produisent des latences non négligeables d'échanges entre les nœuds. De plus, en présence d'obstacless autour de la station sol (un immeuble, par exemple), la communication vers le satellite peut être partiellement atténuée ou complètement bloquée.

Une illustration de cette architecture est visible sur la figure 1.5.c.

1.3.3 Architecture de communication réseau cellulaire (Réseau semi-centralisé)

C'est l'architecture utilisée dans le domaine de la téléphonie et qui se fonde sur l'utilisation d'une infrastructure de stations de base. Des cellules sont déployées de façon plus ou moins importante en fonction de la densité du réseau recherchée. Dans chaque cellule, peuvent se trouver un sous-ensemble de drones et une station sol qui gère le groupe [FB09]. La communication entre groupes doit passer par la station sol. Contrairement à l'architecture lien centralisée, la communication directe inter-drones peut être établie, mais seulement à l'intérieur d'une cellule. Il est aussi possible d'étendre la portée de la mission à travers le déploiement de plusieurs stations. Ces stations peuvent offrir plusieurs liens de communication qui permettent, en cas de dégradation de performance sur un lien donné, d'en utiliser un autre.

Cette architecture se heurte à des limitations en matière de coût de déploiement. En effet, la mise en œuvre coûteuse de cette infrastructure constitue un handicap important dans les lieux où aucune infrastructure de couverture n'est encore mise en place. Si l'utilisation des drones n'est pas fréquente, comme c'est le cas pour les applications de type surveillance lors de catastrophes naturelles, le retour sur investissement peut ne pas être garanti [FB09].

Cette architecture est illustrée par la figure 1.5.b.

1.3.4 Architecture de communication ad hoc

Un réseau ad hoc sans fil (illustré par la figure 1.5.d.) est caractérisé par un ensemble d'entités potentiellement mobiles possédant une ou plusieurs interfaces radio qui mettent en place un réseau de communication de courte durée selon les besoins de l'application. Ces nœuds peuvent être amenés à entrer ou sortir du réseau à tout moment. Le réseau ad hoc sans fil est décentralisé et capable de s'auto-organiser sans la nécessité d'une infrastructure fixe. Si un émetteur n'est pas à portée directe de la machine destination, les informations sont transmises de proche en proche, le long d'un chemin établi et maintenu par le réseau en cas de modification de la topologie. Contrairement au réseau sans fil traditionnel, la zone de service du réseau est la zone géographique dans laquelle les nœuds sont distribués.

Le réseau ad hoc sans fil permet la communication entre deux nœuds qui sont hors de portée directe l'un de l'autre.

Analyse

Dans cette section, listons les avantages de l'architecture ad hoc par rapport aux autres types d'architecture pour une flotte de drones :

— La mise à l'échelle et la reconfiguration agile de la mission exécutée par la flotte de drones : avec l'utilisation d'un système à infrastructure centrale ou par satellite, la zone d'opération sera limitée par la zone de couverture de communication du relais. De plus, en cas de présence d'obstacless, la communication entre les drones peut être bloquée. Grâce à l'architecture ad hoc, il est possible de former une chaine de drones, qui viendrait contourner l'obstacle.

— La fiabilité des communications dans un système aéronautique sans pilote : les drones échangent des messages en temps réel concernant les commandes, la configuration et le déroulement de la mission. Ces messages sont échangés dans un environnement dynamique dans lequel les liens de communication fluctuent et peuvent être coupés. De plus, les drones peuvent être amenés (selon les spécificités de leur mission) à entrer et sortir du réseau de communication. En conséquence, l'aptitude d'auto-organisation d'un réseau ad hoc sans fil permet aux drones de chercher un autre chemin en cas de perte d'un lien.

— La non-nécessité d'une infrastructure dédiée : les réseaux ad hoc sans fil se distinguent des autres types de réseaux par l'absence d'infrastructure centralisée. Les nœuds du réseau sont responsables de l'établissement et du maintien de la connectivité du réseau.

— L'utilisation d'un réseau ad hoc sans fil assure la mobilité et la flexibilité [CHD13]. Grâce à la topologie dynamique, les nœuds mobiles du réseau se déplacent librement et arbitrairement en fonction des objectifs de la mission. Puisque les liens de la topologie peuvent être unidirectionnels ou bidirectionnels, les nœuds peuvent accéder au réseau ou en sortir librement.

— La sécurité d'un réseau ad hoc mobile est distribuée entre les nœuds, contrairement au réseau centralisé et cellulaire. De ce fait, le réseau ne possède pas un unique point vulnérable dans le réseau. La responsabilité de maintenir l'intégrité du réseau est déléguée à chaque nœud du réseau qui, à travers un mécanisme de routage sécurisé peut contrôler l'authentification des messages échangés.

Les différences majeures entre ces différentes architectures sont regroupées dans le tableau 1.3.

Références	Terminologies
[BAGL14]	UAANET (UAV Ad hoc Network)
[ZZZ08]	Mobile UAV ad hoc network
[AD10b]	Mobile Ad-Hoc Unmanned Aerial Vehicle Communication Network
[BST13]	Flying ad hoc network
[SH10]	Ad Hoc Networks with UAV Node
[CM12]	Airborne network
[AK07]	Network Aerial Robots
[SSHK+12]	Unmanned Aeronautical Ad-Hoc Network
[RR12]	Aerial Communication Network
[LSHK12]	Networks of UAVs
[RGDW10]	Distributed Aerial Sensor Network
[SL12]	UAV fleet networks
[ST10]	UAV swarm

Tableau 1.2 : Différentes terminologies des réseaux ad hoc de drones

	Centralisé	**Satellite**	**Cellulaire**	**Adhoc**
Avantages	- Découverte de service - Aptitude à contrôler l'entrée et la sortie des nœuds	- Connectivité	- Connectivité - Mise à l'échelle en fonction du nombre de stations de base - Possibilité de choisir le meilleur lien parmi ceuxx situés entre les stations de base	- Coût faible - Facile à déployer - Tient compte de la mobilité des nœuds - Communication autonome sans infrastructure - Communication possible entre voisins - Optimisation de l'utilisation de la bande passante
Inconvénients	- Latence d'échange entre deux nœuds - Blocage possible des signaux - Consommation de bande passante - La sécurité dépend d'un seul point	- Latence d'échange entre deux nœuds - Puissance d'émission importante - Coût élevé - La sécurité dépend d'un seul point	- Coût élevé de déploiement - Indisponibilité des infrastructures dans certaines situations - La sécurité dépend des infrastructures fixes	- Communication intermittente - Intrinsèquement, non-contrôle de l'entrée et de la sortie des nœuds - Nécessite des protocoles de communication dynamiques pour gérer la topologie du réseau

Tableau 1.3 : Comparaison des différentes architectures de communication

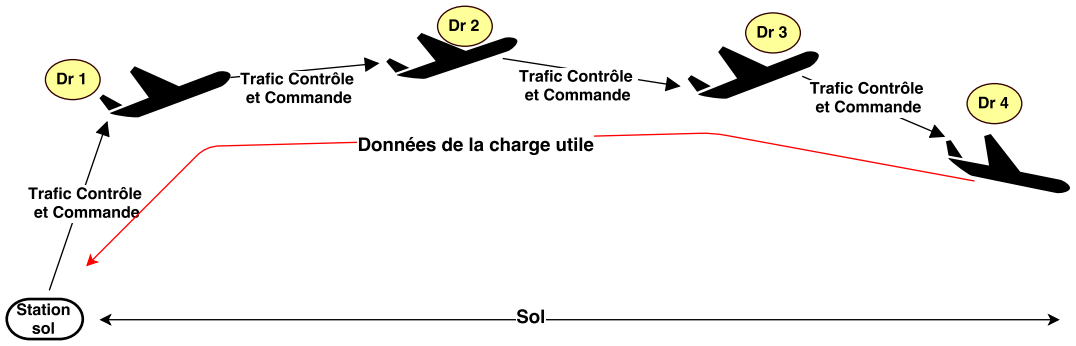

FIGURE 1.6 : Réseau ad hoc de drones

1.3.5 Réseau ad hoc de drones

Le réseau ad hoc de drones, connu sous la dénomination anglaise *UAV ad hoc NETwork* (UAANET) ou encore *Flying Ad hoc NETwork* (FANET), est une sous-catégorie du réseau mobile MANET. Il s'agit du déploiement d'une flotte de drones et de stations sol à travers un réseau ad hoc sans fil. Les drones collaborent entre eux et avec la(les) station(s) sol pour échanger des données qui peuvent être des données propres au routage (des paquets de contrôle) ou des informations propres au système aérien sans pilote.

Plusieurs dénominations (mentionnées dans le tableau 1.2) de ce type de réseau sont employées dans la littérature. La figure 1.6 illustre un réseau UAANET.

Par ailleurs, il existe plusieurs types de messages échangés entre les drones et la station au sol. Ces différents flux applicatifs sont listés dans le tableau 1.4.

Type de flux	**Type de liaison**	**Nature**
Configuration (démarrage autopilote, réglages charge utile	Montant connecté	Critique
Ticks, joystick	Montant non connecté	Non critique
Géoréférencement	Descendant non connecté	Non critique
Données (photos, vidéo)	Descendant non connecté	Non critique
C2	Montant connecté	Critique

Tableau 1.4 : Différents types de flux applicatifs dans un réseau ad hoc de drones

Généralement, ces flux applicatifs comprennent :

— un géoréférencement du drone : ce message est envoyé périodiquement par les drones vers la station de contrôle. Il permet de déterminer la position en temps réel du drone. Ce flux n'exige pas une qualité de service particulière. Il peut supporter quelques pertes ou une latence ;

— un flux tick (ou hearbeat) : c'est un flux de signalement envoyé périodiquement depuis la station sol pour communiquer avec les drones. Il permet de tester l'état de

la liaison entre la station sol et le drone. Ce flux n'exige pas une qualité de service stricte et la perte de paquets par intermittence est acceptable ;

— un flux contrôle et commande (C2), et configuration : ce sont les commandes envoyées vers l'autopilote embarqué à bord ou vers la charge utile. Ce type de flux est considéré comme critique puisque il doit être échangé et traité en temps réel. Il nécessite donc un minimum de délai et un maximum de fiabilité des liens de communication.

— un flux de données : tout comme les flux de configuration, ce type de trafic a des exigences strictes en matière de taux de perte et de délai. Il est traditionnellement envoyé depuis le drone le plus éloigné de la station sol et transféré en relais par les membres de la flotte de drones.

Comme nous le détaillerons dans la chapitre 3, le trafic C2 est régi par des réglementations spécifiques garantissant la sécurité et la sûreté d'une mission. Par exemple, en France, la réglementation exige que chaque drone soit en connexion directe avec la station sol à tout instant, à travers des liens à longue portée ou à courte portée [Dro15]. Cela implique que le trafic C2 doive être envoyé depuis une station sol. Il est possible de le relayer entre drones, mais, aujourd'hui, aucune norme ne prend en compte ce genre de déploiement.

Les réseaux UAANET font généralement partie de la famille des réseaux ad hoc mobiles (MANET). Ils ne nécessitent aucune infrastructure fixe et s'appuient sur la collaboration des nœuds pour échanger des données. Néanmoins, un réseau UAANET possède des caractéristiques spécifiques qui le différencient des autres types de réseau MANET.

Connectivité réseau

La connectivité au sein du réseau UAANET est souvent intermittente en raison du mouvement des drones, qui crée des déconnexions temporaires. En effet, leurs mouvements sont dictés par l'opérateur au sol ou par des plans de vol préchargés. Ainsi, les nœuds risquent d'être en perte de lien de communication avec leurs voisins dès qu'une commande de position est exécutée. Le lien de communication de la source vers la destination peut être indisponible pour une durée indéterminée, ce qui nécessite la réactivité du réseau pour trouver un chemin de secours et éviter des pertes conséquentes aux flux applicatifs.

Densité des nœuds

La densité des nœuds peut être définie comme le nombre moyen de nœuds dans une zone géographique donnée. Dans la littérature, un réseau UAANET ne comporte généralement que quelques drones [DDLW09] [CHJ+10], contrairement au réseau MANET. En effet, grâce à leur vitesse élevée leur permettant ainsi de couvrir une vaste zone, il ne sera plus nécessaire d'en déployer une dizaine, par exemple.

Consommation énergétique

Dans un réseau ad hoc de drones, la durée de vie du réseau dépend de l'autonomie de la batterie et de la consommation d'énergie des différents capteurs. Si l'une des batteries d'un des nœuds est déchargée, le nœud est obligé de se retirer de la mission, ce qui engendre une modification de la topologie du réseau. Selon le type de drones utilisés, ce problème peut survenir dans un réseau UAANET, notamment avec les drones de type Paparazzi [4] qui ont une autonomie de 10 à 15 minutes en vol [Pap03]. Cependant, dans la plupart des cas, notamment pour les drones de Delair-Tech [Man14], le problème se pose moins puisque ils ont une autonomie de vol de 2 heures [MA15]. Dans ce cas, la topologie du réseau peut être maintenue sur une durée plus longue.

Modèle de mobilité

Le modèle de mouvement des nœuds MANET est différent de celui des mini-drones. En effet, les nœuds MANET se déplacent souvent sur une zone se trouvant, dans la plupart des cas, au sol. Par exemple, pour les réseaux ad hoc véhiculaires VANET (Vehicular Ad hoc NETwork), les nœuds se déplacent sur la route ou l'autoroute. En revanche, les mini-drones tout comme les avions (nœuds du réseau AANET : Aeronautical Ad hoc NETwork), se déplacent dans le ciel.

En outre, le réseau MANET utilise généralement un modèle de mobilité *random waypoint* dans lequel la direction et la vitesse des nœuds sont choisies au hasard [AWS06]. Pour le réseau VANET, le modèle de mobilité est hautement prévisible [HFB09].

En ce qui concerne le réseau UAANET, le modèle de mobilité dépend généralement de divers paramètres. Il est le plus souvent prévisible, mais, dans la majorité des cas, il est dynamiquement modifié à cause de la vitesse des drones, des conditions climatiques et de nombreux autres paramètres géographiques et topographiques [CLMG09]. En effet, nous pouvons rencontrer quelques cas d'applications de flottes de drones avec des trajectoires prédéfinies ou préférées. Néanmoins, comme l'environnement est dynamique, le plan de vol est souvent amené à être recalculé par l'autopilote [CCC10]. Quelques modèles de mobilité ont été proposés dans la littérature. Dans [BAGL14], un modèle de mobilité PPRZM (PaPaRaZzi Mobility) a été suggéré. Ce modèle a été défini dans l'objectif de reproduire des mouvements dans un simulateur réseau pour créer un déplacement plus réaliste des nœuds, lequel se compose des mouvements principaux suivants :

— Mouvement rectiligne : déplacement rectiligne d'un point vers une position de destination. Le nœud peut faire des allers-retours rectilignes ;

4. C'est un système complet de logiciels et de matériel libre (open source) permettant de gérer des systèmes aériens sans pilote. Il est composé d'un autopilote pour diriger des aéronefs et d'une station sol contenant des logiciels de planification des missions.

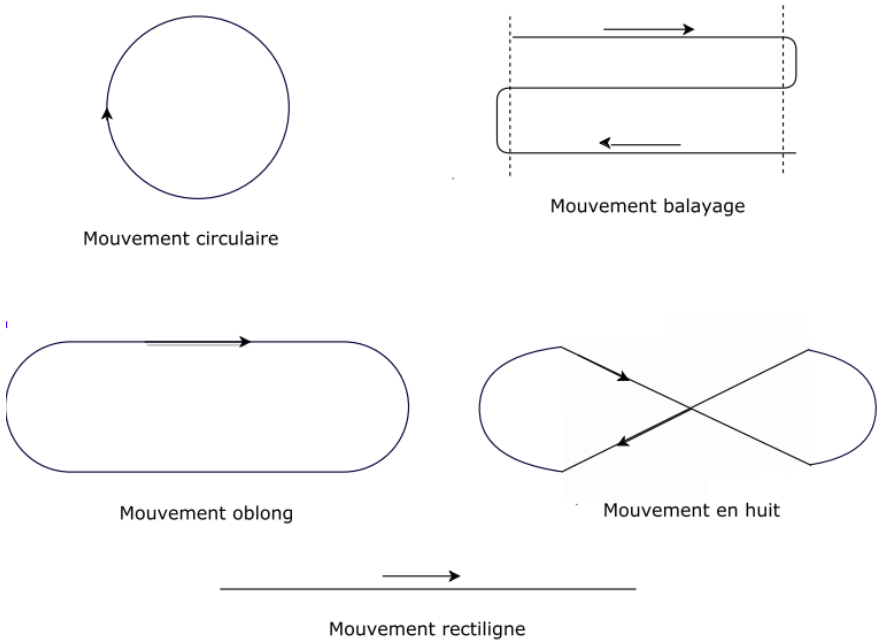

FIGURE 1.7 : Illustration des différents types de mobilité PPRZM

— Mouvement circulaire : le drone survole une zone donnée, suivant une trajectoire circulaire. Il est défini par la position du centre et le rayon qui entoure la zone cible ;

— Mouvement oblong : il est constitué par des allers-retours rectilignes, décalés entre deux points fixes, avec un demi-tour une fois chaque point franchi ;

— Mouvement *huit : il est assez similaire au mouvement oblong. La seule différence se trouve dans le croisement de leurs allers-retours ;*

— *Mouvement balayage : ce mouvement réalise des allers-retours décalés d'une manière continue, permettant de couvrir la surface d'un rectangle désigné par deux de ses angles opposés.*

Ces mouvements sont illustrés par la figure 1.7.

Par ailleurs, dans [WGWW10], un modèle de mobilité SRCM (Semi-Random Circular Movement) a été proposé, dans lequel une fonction de distribution approximative des nœuds est déduite par un disque région à deux dimensions.

Dans [KNT06], deux modèles de mobilité ont également été proposés. Le contexte de mouvement aléatoire a été maintenu. Chaque drone se déplace indépendamment de ses voisins et décide de sa direction en suivant un processus prédéfini et régi par une loi de Markov. Concernant le deuxième modèle de mobilité, les drones maintiennent une carte topographique construite à partir de « phéromones » qui guident leurs mouvements et dont l'action est comparable aux hormones émises par certains animaux : elles agissent en tant que messagers entre les individus d'un même groupe.

En regroupant ces trois approches qui prennent en compte les conditions imprévisibles d'un vrai déploiement d'une flotte de drones [CLMG09], le modèle proposé par Bouachir et al. [BAGL14] offre une meilleure couverture des différents mouvements des drones. En effet, dans un réseau UAANET, le contrôle de la mobilité des nœuds est assuré par le pilote. Suivant la variabilité de la force du signal ou la surface de la zone à couvrir, le pilote peut décider d'ajuster à tout moment le mouvement des drones. De plus, selon le contexte d'application, la mobilité des nœuds du réseau UAANET n'est pas restreinte. Le pilote gère l'ensemble de la mission à partir d'un logiciel de station sol et a la possibilité de modifier son plan de vol en complète autonomie.

Environnement

Dans la plupart des cas des réseaux MANET, les nœuds se déplacent au sol, alors que, pour un réseau UAANET, les drones circulent dans l'air. Par conséquent, le modèle de perte en espace libre est souvent utilisé pour modéliser la couche physique.

Délai strict et contraint

Généralement, les réseaux UAANET sont utilisés pour des applications en temps réel (télésurveillance, par exemple). Par conséquent, les flux de contrôle et de commande doivent arriver et être traités en temps réel par les drones afin d'éviter une perte de contrôle de l'aéronef.

Le tableau 1.5 synthétise ces spécificités du réseau UAANET.

	UAANET	**MANET**
Mobilité des nœuds	très rapide	lente (suivant les nœuds)
Modèle de mobilité	modèles spéciaux comme le PPRZM	généralement aléatoire
Densité des nœuds	faible	forte (suivant le type d'application)
Modification de topologie	rapide	lente (comparé au réseau UAANET)
Modèle de propagation radio	en altitude, généralement en LoS (Line of Sight)	généralement au sol, pas de LoS (suivant la topologie du réseau)
Type de flux	temps réel	non-temps réel

Tableau 1.5 : Comparaison entre les réseaux UAANET et MANET

1.3.6 Projets de réseau ad hoc de drones

Quelques projets ont été lancés pour étudier les réseaux ad hoc de drones. Dans [CLM+11], le projet CARUS (Cooperation Autonomous Reconfigurable UAV Swarm) a pour objectif de permettre la localisation des cibles au sol avec une flotte composée de cinq drones.

Chaque drone est responsable d'une parcelle de zone donnée déterminée par la station de contrôle. Ils sont aussi munis d'un système grâce auquel les collisions sont évitées. Les informations de détection sont échangées entre voisins et sont rapatriées vers la station sol pour ne pas surveiller une zone plusieurs fois. Un protocole de communication qui permet d'établir une communication asynchrone est exécuté par chacun des drones.

Le projet AUGNet (UAV-Ground Network) [BAD+04] a pour objectif de déployer une flotte de drones afin d'apporter un support de communication aux nœuds situés au sol. Chaque drone est donc utilisé comme une passerelle de communication qui assure l'échange d'informations avec les entités au sol. Le protocole Dynamic Source Routing [JMB+01] (DSR) a été retenu pour ce projet de déploiement.

Le projet Airshield [DDLW09] a pour objectif de déployer un réseau ad hoc de drones permettant la supervision d'une zone géographique suite à un désastre. Le cas de la supervision d'un feu de forêt a été spécifiquement étudié. Pour pouvoir suivre convenablement la mission, l'objectif est d'échanger les flux applicatifs et de garantir un niveau acceptable de performance réseau. Pour assurer la fiabilité et la sécurité de la mission, des algorithmes interdépendants sont déployés. Le premier est l'IDL (Inter Drone Links), qui maintient la connectivité entre les drones de la flotte en échangeant des informations sur la topologie du réseau. Le deuxième, DGSL (Drone to Ground Station Links) est utilisé pour la communication descendante vers la station sol afin d'envoyer les informations de télémétrie.

Dans [HLZF15], le projet SMAVNET (Swarming Micro Air Vehicle Network) est présenté. L'objectif est de déployer une flotte de drones dans une région touchée par une catastrophe naturelle. Un réseau ad hoc de drones est donc déployé sur une zone donnée en utilisant le protocole de routage Predictive Optimized Link State Routing Protocol (P-OLSR) [RKT13]. Les drones échangent les informations topologiques entre eux, et envoient directement vers la station sol une vidéo de surveillance en temps réel.

Dans [MRL15], les travaux de cette thèse sont présentés. Ils s'inscrivent dans le cadre du projet SUANET (Secure UAV Ad hoc NETwork) mené en collaboration avec le laboratoire TELECOM de l'ENAC et l'entreprise Delair-Tech [Del15]. L'objectif est de définir et de concevoir une architecture de communication sécurisée pour une flotte de drones. Pour cela, un protocole de routage sécurisé, intitulé SUAP (Secure UAV Ad hoc routing Protocol), a été proposé. Il assure l'authentification des messages entre les drones.

Ces différents projets montrent l'intérêt croissant porté au concept de réseau de communication d'une flotte de drones et donc à l'utilisation d'un réseau ad hoc sans fil pour la communication interdrone. Nous pouvons sougliner la nécessité de développer des protocoles de communication pour des drones partageant des objectifs complémentaires au sein d'une mission. Il existe encore peu d'études consacrées à ce sujet. De plus, la majorité des expérimentations effectuées dans ce domaine relève d'études en simulation, qui restent

théoriques.

Dans la section suivante, nous analyserons les protocoles de communication le plus pertinents pour établir la communication au sein d'une flotte de drones.

1.4 Protocoles de communication d'un réseau ad hoc de drones

1.4.1 Liaison de données (*Medium Access Control*)

Bien que l'objectif de la thèse ne soit pas d'étudier les protocoles de communication sur la couche liaison, il est important d'en connaître leur existence et leur fonction. Le rôle de la couche liaison est de fiabiliser la transmission point à point en vue de satisfaire les exigences de latence des paquets de données possédant différentes priorités. Elle est aussi responsable de la création des trames, du contrôle d'erreur et du contrôle d'admission des différents services pour l'ordonnancement de paquets prioritaires. En outre, à part le problème de qualité de service, la couche MAC est aussi responsable de l'adoption d'une stratégie d'économie d'énergie et de ressources réseau dans la gestion des puissances de transmission et permettre ainsi l'utilisation de la bande passante disponible. Un mécanisme de plage de transmission dynamique et d'attribution de puissance peut être mis en place.

Parmi les protocoles ad hoc de niveau liaison, nous avons la famille 802.11 [CHKV06] de l'IEEE. Ce choix se justifie pour une utilisation simplifiée dans un contexte de communication ad hoc multisaut.

La norme IEEE 802.11 régit le fonctionnement de la couche physique (PHY), de la couche MAC et de la couche *Logical Link Layer* (LLC) de la pile protocolaire OSI. Elle dispose de deux méthodes d'accès : la méthode Point Coordination Function (PCF) et la méthode Distributed Coordination Function (DCF). Avec la méthode d'accès PCF, les stations de base ont la charge de l'accès au médium radio de manière centralisée. Avec la méthode DCF, l'accès est totalement distribué et il n'existe aucune distinction entre les stations de base. Ces particularités justifient le fait que le DCF peut être employé en mode ad hoc et en mode infrastructure alors que le PCF ne peut l'être qu'en mode infrastructure. Dans la suite de ce manuscrit, en faisant référence à la méthode d'accès de 802.11, nous supposerons seulement l'emploi de la méthode d'accès DCF. Cette méthode applique les principes de CSMA/CA [ZA02b] pour l'accès au médium. Avec le CSAMA/CA, les nœuds doivent attendre que le médium soit libre pendant un temps d'attente fixe, appelé DIFS (DCF InterFrame Spacing).

Dans la littérature, la majorité des prototypes d'application des réseaux UAANET et MANET se base principalement sur la famille des protocoles de liaison sans fil, tel IEEE 802.11. Par exemple, le standard IEEE 802.11 a été utilisé pour les communications entre

les drones et la station sol avec les modèles paparazzi [BGLG13]. Toutefois, malgré la popularité du standard 802.11 dans les réseaux ad hoc, il présente quelques inconvénients en matière de qualité de service, notamment quand il s'agit de garantir la bande passante nécessaire pour les trafics importants (par exemple, les flux de contrôle et de commande). Des améliorations ont été proposées à l'image du standard IEEE 802.11b [OP05] qui fournit un débit théorique entre 5.5 et 11 Mbit/s. Il a été utilisé en UAANET dans [BAD+04] [LOBH07] [BLT02]. Une autre amélioration de la technologie IEEE 802.11 est la version IEEE 802.11a [RC05] car elle offre un débit théorique de 54 Mbit/s. Elle a été employée pour un réseau UAANET dans [AD10b]. Par ailleurs, des protocoles MAC, spécifiques aux réseaux UAANET et aux extensions du standard IEEE 802.11, ont été proposés. Nous pouvons citer le protocole AMUAV [AD10a], le protocole LODMAC [TB15] et le protocole Token-MAC [CYL+13].

1.4.2 Niveau réseau : routage dans les réseaux UAANET

Le routage est une méthode contribuant à l'acheminement des données depuis un nœud émetteur jusqu'à son (ses) destinataire(s). Dans les réseaux ad hoc mobiles, le routage se base sur une approche de réémission des paquets. Le problème réside donc dans le choix du chemin optimal. En effet, le routage revient à calculer le meilleur chemin reliant deux nœuds quelconques dans le réseau. Il s'agit alors d'affecter une certaine métrique aux liens, afin que la recherche de route revienne, par exemple, à calculer le plus court chemin entre la source et la destination.

Pour atteindre sa destination, le message peut être relayé par des nœuds intermédiaires, en fonction de la disponibilité des liens de communication. Dans le cas spécifique d'un réseau ad hoc de drones, pour réaliser des missions, les drones doivent relayer les trafics de charge utile (et dans certains cas les trafics de contrôle en fonction de la réglementation) entre eux et la (les) station(s) sol. Pour cela, les nœuds disposent d'une table de routage qui assure la correspondance entre le nœud destination et le chemin pour y parvenir. Selon la technique utilisée, proactive ou réactive, les nœuds mettent en place, respectivement, une route, précédant ou suivant l'apparition de la nécessité d'envoi d'un message au niveau applicatif. Une illustration d'un mécanisme de routage est visible sur la figure 1.8.

Il est important de noter qu'un réseau ad hoc de drones a pour particularité la grande mobilité de ses nœuds. Les changements fréquents de topologie impliquent des changements de route et donc une surcharge de gestion (« overhead ») du mécanisme de routage. Aussi, il est nécessaire de restaurer rapidement une route qui aurait été perdue pour éviter une perte des paquets ou une utilisation sous-optimale de la bande passante en raison des retransmissions consécutives aux pertes. Puisque les trafics échangés sont critiques, il est important d'avoir un mécanisme de secours rapide et efficace. Avec ces contraintes, le protocole de routage doit présenter de bonnes propriétés d'overhead et d'exécution proche

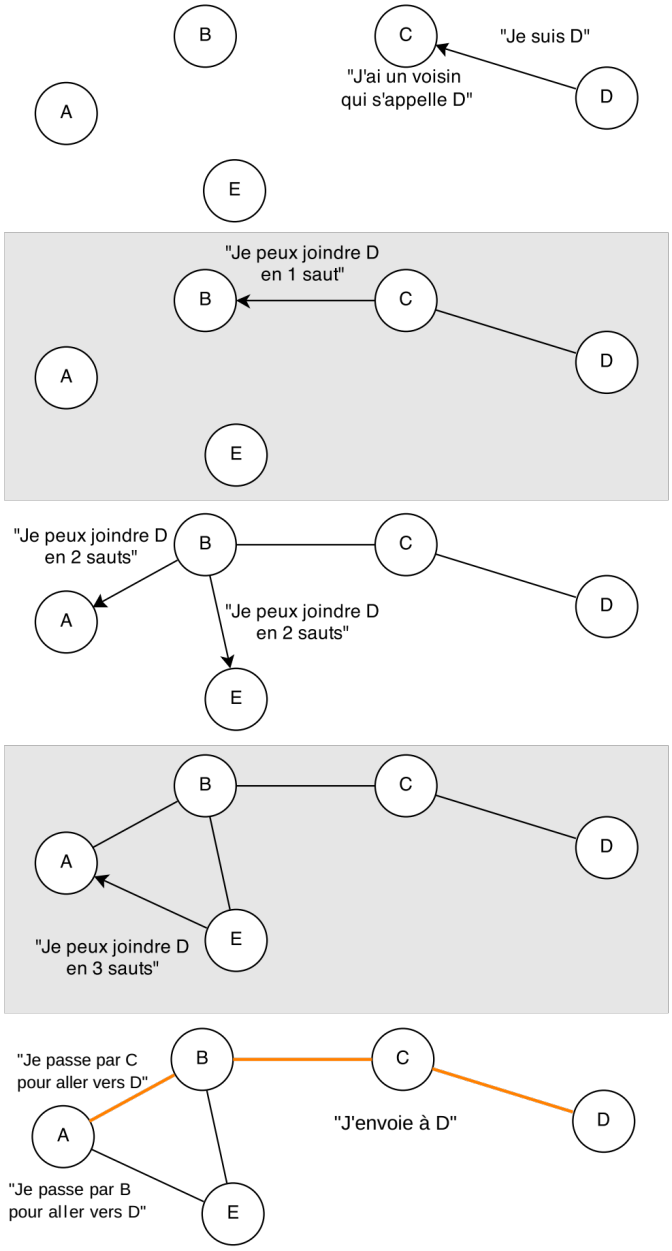

FIGURE 1.8 : Illustration d'un mécanisme de routage ad hoc par l'établissement d'une route de A vers D

du temps réel. Dans ce contexte, de nombreux protocoles de routage pour les réseaux ad hoc de drones ont été proposés dans la littérature. Ce sont, pour la plupart, des extensions de protocoles de routage existant dans le domaine des réseaux MANET [Sah14].

De manière générale, les protocoles de routage ad hoc sont classés suivant diverses métriques de conception : leur méthode de création et de maintenance des routes, ou leur méthode d'acheminement des données. Dans [MML17a], nous avons présenté une étude sur les protocoles de routage UAANET existants et les défis en matière de sécurité associée aux réseaux UAANET. Dans ce qui suit, nous n'allons présenter que les grandes familles de protocoles de routage UAANET, et ce pour ne pas noyer les lecteurs dans les détails techniques de chaque protocole. Aux lecteurs qui seraient intéressés par ces détails, nous leur proposons de lire notre revue sur le protocole de routage et les problématiques de sécurité associées dans [MML17a].

1.4.2.1 Protocole hiérarchique

Les protocoles de routage hiérarchiques fonctionnent en confiant aux nœuds des rôles spécifiques qui varient en fonction du temps. Certains nœuds sont élus et assument des fonctions particulières qui conduisent à une vision hiérarchisée de la topologie du réseau. Par exemple, les nœuds élus peuvent être utilisés comme passerelles pour un certain nombre de nœuds qui se seront rattachés à ce dernier. Ainsi, le routage sera simplifié, puisque il se fera de passerelle à passerelle.

Les protocoles de routage UAANET suivants se basent sur ce principe : Clustering algorithm of UAV networking [KJT08], Mobility Prediction Clustering Algorithm (MPCA) [ZZ11], Multi Meshed Tree (MMT) [MAMS11], Disruption Tolerant Mechanism (DTM)[FD07].

1.4.2.2 Protocole réactif

Le principe du protocole de routage réactif est de ne garder en mémoire que les routes en cours d'utilisation pour le processus de routage. Le processus de construction de route n'est effectué qu'à la demande d'envoi de données (faite par la couche application). La méthode de recherche de route consiste à inonder le réseau avec une requête, afin de trouver la station cible, qui répond en suivant le chemin inverse.

Lorsqu'un nœud source souhaite envoyer un message vers un nœud destinataire, il lui envoie sur le réseau une requête de recherche de chemin, requête nommée souvent RREQ (Route REQuest). Quand un nœud reçoit cette requête, il la transmet à ses voisins, sauf s'il l'a déjà reçue ou qu'il est le destinataire du message. Lors de la transmission d'une requête, selon le type de protocole utilisé, le traitement diffère. Pour un protocole à table de routage, le nœud actualise sa table de routage en ajoutant une entrée qui indique le prochain saut pour atteindre le nœud source indiqué dans le message de requête, créant

ainsi la route inverse pour atteindre la source. Dans le cas d'un protocole de routage à la source, tel que DSR, à la réception d'un message de requête, le nœud intermédiaire ajoute son identifiant au chemin parcouru depuis la source. Le chemin est alors stocké dans sa totalité dans la requête qui est transmise au destinataire. Dans les deux cas, à la réception d'un message requête, le destinataire répond avec un message de réponse RREP (Route REPly). Ce paquet va suivre le chemin inverse parcouru par la requête RREQ, soit à l'aide des tables de routage de chaque nœud intermédiaire, soit à l'aide du chemin contenu dans l'entête du message RREQ. Dans le cas de l'utilisation d'une table de routage, à la réception d'un message RREP, le nœud source ajoute une entrée à sa table pour joindre le destinataire. Le protocole doit également vérifier périodiquement la connectivité de chaque lien.

L'inconvénient est que ce type de protocole doit réagir rapidement en cas de coupure d'une route en cours d'utilisation pour trouver un chemin alternatif. En conséquence, ces mécanismes de maintenance sont lourds à gérer et créent des sursauts du trafic de contrôle à chaque tentative de recherche ou de réparation de route. Ces mécanismes entrainent un délai d'attente et une bufferisation des paquets de données due au défaut de route.

Parmi les protocoles basés sur des principes réactifs, nous pouvons relever les grandes familles suivantes : Ad hoc On Demand Distance Vector (AODV) [PBRD03b], Dynamic Source Routing (DSR) [JMB+01], Temporally Ordered Routing Algorithm (TORA) [GSV10] et leurs déclinaisons : Time slotted AODV [FHS07], Rapid-reestablish Temporally Ordered Routing Algorithm (RTORA) [ZR13] et Reactive-Greedy-Reactive (RGR) [SSHK+12].

Ad hoc On Demand Distance Vector (AODV) : AODV est un protocole de routage réactif utilisant un mécanisme de découverte de route et de maintenance de route qui suit le mécanisme décrit ci-dessus. Une fois la route tracée, les nœuds n'ayant pas été contactés ne participeront pas à l'échange ni à la mise à jour de la route. Pour la maintenance des routes, la détection des ruptures de liens est réalisée à l'aide de messages spécifiques Hello diffusés périodiquement d'un nœud vers ses voisins immédiats ou par l'écoute de la transmission d'un paquet de données sur le lien suivant. Le nœud détectant une rupture de lien diffuse un message d'erreur à l'aide du paquet Route Error (RERR) contenant l'adresse de la (des) destination(s) inaccessible(s).

Le protocole de routage AODV présente l'avantage d'éliminer la surcharge caractérisée par le routage à la source du protocole DSR. Il évite également l'envoi systématique des mises à jour comme dans DSDV. Néanmoins, il présente quelques inconvénients : les liens doivent être symétriques. Cette propriété conduit au fait que le choix de la route la plus récente ne conduit pas forcément à la route optimale ; une seule route par destination est conservée, ce qui crée une latence lors d'une rupture de liens ; de plus une surcharge

importante est générée par le recours fréquent à des phases de découverte de route par la diffusion d'un même RREQ à un nœud donné et par l'utilisation éventuelle de messages Hello.

Dynamic Source Routing (DSR) : DSR est un protocole de routage réactif basé sur le concept du routage à la source. Lors de la phase de découverte, un nœud émetteur regarde dans son cache local si un chemin existe déjà vers la destination. Si c'est le cas, elle sera utilisée en utilisant l'algorithme de routage à la source ; dans le cas contraire, un processus de recherche de route comme décrit précédemment est lancé. Le routage à la source stipule que seule la source peut décider du choix de la route. Aussi, le chemin complet est enregistré dans l'entête des paquets de découverte de route depuis la source vers la destination.

En ce qui concerne la phase de maintenance de route, elle est réalisée en utilisant des paquets d'erreur Route Error (RERR) et d'acquittements ACK. C'est-à-dire qu'un nœud n'ayant pas répondu à plusieurs tentatives de retransmission sera supprimé du chemin à l'aide des paquets de contrôle contenant les adresses des deux extrémités du lien défaillant. Cette absence de réponse peut être détectée par la couche MAC en cas d'acquittement de proche en proche, par acquittement passif si le nœud est capable d'entendre la suite de sa transmission de paquets, ou encore par l'utilisation d'acquittements spécifiques à la couche réseau.

Généralement le routage à la source permet de repérer immédiatement les boucles et de les éliminer. De plus, les nœuds apprennent dynamiquement les routes en scrutant les paquets RREQ et RREP. Néanmoins, DSR introduit une surcharge croissante dans chaque paquet de données qui contient le chemin complet vers la destination ainsi qu'un trafic inutile lié à la diffusion des paquets de contrôle. Ensuite, l'existence d'un lien asymétrique dans le réseau constitue un autre inconvénient majeur de ce protocole, car, dans ce cas, une nouvelle procédure de découverte doit être initiée par la destination.

1.4.2.3 Protocole proactif

Le principe des protocoles de routage proactif repose sur le maintien en mémoire des routes permettant de joindre tous les nœuds du réseau. Cette responsabilité de maintien est laissée au nœud du réseau par l'échange périodique de messages de contrôle. Cette mise à jour périodique des données de routage est diffusée par les différents nœuds du réseau. Cet échange permet donc de récupérer toutes les informations pour calculer les chemins selon des critères prédéfinis. Parmi ceux-ci, nous pouvons citer la technique des plus courts chemins (nombre de sauts séparant les nœuds), le délai de transmission ou encore la bande passante disponible sur le chemin identifié. Nous pouvons aussi associer des coûts au lien de communication ; le coût peut signifier le taux d'utilisation d'un lien, les délais relatifs

de communication ou tout autre facteur qu'il convient de minimiser lors du routage des données.

Les protocoles de routage proactifs utilisent soit un algorithme d'état de lien, soit un algorithme de vecteur de distance. Dans la méthode « état de lien », chaque nœud garde une vision de toute la topologie du réseau, et ce à l'aide des requêtes périodiques portant sur l'état des liaisons avec les nœuds voisins. Pour que cette vision soit à jour, chaque nœud diffuse périodiquement l'état des liens de ses voisins à tous les nœuds du réseau, même quand il y a un changement d'état de liens. Un nœud qui reçoit les informations concernant l'état des liens met à jour sa vision de la topologie du réseau et applique un algorithme de calcul des chemins optimaux afin de choisir le nœud suivant pour une destination donnée. L'un des algorithmes les plus connus appliqués dans le calcul des plus courts chemins est celui de Dijkstra [Ski90]. Notons que le nœud de routage calcule la plus courte distance qui le sépare d'une destination donnée, en se basant sur l'image complète du réseau formée grâce aux liens les plus récents de tous les nœuds de routage.

Dans la méthode « vecteur de distance », chaque nœud diffuse à ses voisins sa vision des distances qui le séparent des autres nœuds du réseau. En se basant sur les informations reçues par tous ses voisins, chaque nœud de routage effectue un certain calcul pour trouver le chemin le plus court vers n'importe quelle destination. Le processus de calcul se répète s'il y a une modification de la distance minimale séparant deux nœuds, et cela jusqu'à ce que le réseau atteigne un état stable. Cette technique est basée sur l'algorithme « Distribué de Bellman-Ford » (DBF) [GR93].

L'inconvénient de ce type de protocoles est le coût de maintien des informations de topologie et de routage lorsqu'il n'y en a pas besoin et en l'absence de trafic de données. Cette propriété réserve ce type de protocoles à des réseaux de taille modeste. En effet, le flux de contrôle induit une consommation permanente de bande passante, qui augmente fortement avec le nombre de nœuds et qui devient problématique pour les réseaux de grandes tailles [VMA14].

Parmi les protocoles basés sur des principes proactifs, nous pouvons citer OLSR [CJ03], DOLSR [AD11], TBRPF [OTBL02], DSDV [PB01], FSR [PGC00], Predictive-OLSR [RKT13], ML-OLSR [ZWL+14] et COLSR [LL12].

Optimized Link State Routing Protocol (OLSR) : le protocole OLSR est basé sur le principe proactif avec un algorithme de type « état de lien ».

Pour éviter l'inondation des paquets de routage qui peut provoquer des saturations, le protocole se base sur l'élection de nœuds spécifiques normés MPR (MultiPoint Relay) chargés de transmettre les informations de topologie. La sélection de ces nœuds MPR se fait à partir de messages Hello pour en déduire la nature des liens, symétriques ou asymétriques, qui les relient. La condition pour pouvoir être MPR est d'atteindre, avec

un lien symétrique, tous les nœuds voisins situés à une distance de deux sauts du nœud initial. L'ensemble des nœuds MPR doit donc couvrir tout le voisinage situé à deux sauts.

OLSR présente l'avantage de réduire la taille des messages de contrôle, car seuls les liens aux MPR sont utilisés. De plus, l'inondation de messages de contrôle est limitée aux nœuds MPR. OLSR est adapté aux réseaux étendus, denses et aux trafics sporadiques. Toutefois, pour des réseaux à faible densité comme UAANET, il engendrera beaucoup de messages de contrôle, ce qui consommera plus de bande passante.

1.4.2.4 Protocole géographique

Dans un protocole de routage géographique, un nœud est supposé connaître sa position géographique, celle de ses voisins et celle du nœud de destination. La connaissance du voisinage est périodiquement mise à jour à l'aide des messages Hello échangées entre nœuds voisins.

Chaque nœud connaît sa position géographique à l'aide d'un système tel que le GPS (Global Positioning System). Les données sont par la suite envoyées en direction de la position géographique du destinataire. Le voisin le plus proche de la destination retransmet le message. La difficulté réside dans le fait que chaque nœud doit connaître la position de tous ses voisins. Les messages Hello, qui sont envoyés à intervalles réguliers (mécanisme proactif) ou à la demande (mécanisme réactif), peuvent être utilisés pour donner des valeurs approximatives. La méthode réactive permet d'envoyer une requête de voisinage uniquement quand un nœud a besoin de la position de ses voisins, tandis que la méthode proactive inonde le réseau pour obtenir la dernière position de chaque nœud. Ces messages envoyés pour découvrir la position des voisins représentent la plus grande partie de l'overhead des protocoles géographiques. Le temps d'envoi et d'exécution de ces informations a aussi des répercussions sur le délai d'acheminement des messages entre une source et une destination du réseau.

Dans une approche « greedy geographic forwarding », le nœud le plus proche de la destination, parmi les nœuds voisins, est choisi comme prochain saut. Comme métrique de proximité, il est possible d'utiliser la distance euclidienne ou la projection sur une ligne entre le nœud courant et la destination.

Les protocoles suivants sont basés sur ce principe : Greedy Perimeter Stateless Routing for Wireless (GPSR) [KK00], Greographic Position Mobility Oriented Routing (GPMOR) [LSLY12], Mobility Prediction Geographic Routing (MPGR) [LSWY12], Location Aware Routing for opportunistic Delay Tolerant Network (LAROD) [KNT11], Reactive-greedy-Reactive (RGR) [SSHK+12].

1.5 Discussions et Conclusions

Dans ce chapitre, nous avons évoqué l'application d'un système de drones et sa convergence vers la flotte de drones. Cette dernière permet une coordination autonome entre plusieurs drones pour améliorer la capacité du système UAS. La communication entre les drones se fait à travers un réseau de communication donné et doit garantir la livraison des données, tout en s'adaptant dynamiquement aux conditions de mise en œuvre d'une flotte de drones. Suite à cela, nous avons détaillé les différents types d'architectures pouvant être utilisés pour une flotte de drones. Grâce à l'aptitude de reconfiguration d'un réseau ad hoc sans fil, l'entrée et la sortie des drones dans le réseau sont plus faciles à gérer par l'intermédiaire des protocoles de routage. Cette architecture convient également à l'échange des différents flux temps réel puisque un routage dynamique permet de trouver un chemin de secours en cas de rupture de liens ou de présence d'obstacles entre la source et le destinataire.

Pour utiliser le réseau UAANET, il est nécessaire de mettre en place un protocole de routage dynamique qui permettra à tous les nœuds de communiquer. Beaucoup de protocoles de routage ont été proposés ces dernières années, chacun ayant sa propre métrique et son propre objectif. Le choix de l'algorithme de routage UAANET est complexe, car il doit répondre aux besoins spécifiques du réseau UAANET, c'est-à-dire :

— l'absence totale d'infrastructure fixe pour distribuer les informations de routage ;

— la topologie du réseau, caractérisée par une faible densité des nœuds, et qui est amenée à évoluer au cours de la mission ;

— la mobilité relative des nœuds. Les drones DT-18 [DT116] de la société Delair-Tech mentionnés dans cette thèse ont une vitesse de 17 m/sec.

— les flux temps réel applicatifs cités précédemment doivent être échangés avec une faible latence ;

— l'intégrité du réseau, c'est-à-dire le rejet des faux paquets ou des paquets modifiés par l'attaquant. Ce dernier point sera approfondi dans le chapitre suivant.

Au regard de ces critères, il semble que les protocoles proactifs ne soient pas appropriés pour des réseaux à faible densité et à forte mobilité comme les réseaux UAANET, car la capacité du réseau n'est utilisée que pour maintenir à jour les informations de routage.

En ce qui concerne les protocoles hiérarchiques, leur utilité n'est pas justifiée pour une flotte de drones puisque une faible densité de nœuds est généralement considérée pour un réseau UAANET. Il n'y aurait donc pas suffisamment de nœuds pour construire une classe de nœuds dans un réseau UAANET.

Pour les protocoles de routage géographique, aucune approche concrète n'est proposée à ce jour pour l'acquisition et le partage de la coordonnée du nœud destinataire dans le réseau. En effet, seules quelques idées ont été partiellement proposées [MWH01]. La

première approche consiste à utiliser un serveur centralisé pour des requêtes/réponses sur la position géographique du nœud destinataire. La deuxième approche consiste à demander de façon réactive ou proactive l'information de localisation. L'inconvénient majeur de ces approches réside dans le fait que la mise en place d'un serveur centralisé dans le réseau nécessite de créer un mécanisme de sécurité du serveur puisque il constitue une vulnérabilité supplémentaire pour le système final. Le temps d'exécution de la requête et sa réponse est aussi non négligeable. Pour la deuxième approche, exécuter une requête réactive, ou proactive, de la position ajouterait un délai qui différencierait la position réelle de la position obtenue.

Les protocoles réactifs sont, quant à eux, adaptés à des réseaux de plus petites tailles, où le délai d'établissement des routes devient moins important. La recherche de route à la demande permet de réagir promptement en cas de perte de lien. Il semble donc que le choix de l'approche réactive est pertinent pour notre application puisque la quantité de paquets de routage échangés est réduite. Les nœuds ne génèrent de paquets de découverte de route qu'en cas de besoin. De plus, même si les protocoles réactifs ne sont généralement pas préconisés pour des applications temps réel (car ils induisent des délais imprévisibles), la faible densité des nœuds UAANET permet de réduire l'impact de l'écart temporel. Des études ont montré des performances similaires en matière de délai de bout en bout entre AODV et OLSR [MRL15] [HMBT07]. Ce type de routage est donc adapté pour des réseaux de topologie dynamiques. Les expérimentations dans la littérature ont montré que le protocole AODV donne de bons résultats en matière de délai de transmission et de débit, quelle que soit la mobilité des nœuds [KJG12]. TORA obtient de moins bonnes performances à cause de la surcharge du réseau.

Toutefois, il est difficile de conclure sur la prédominance d'un protocole par rapport aux autres. Chaque protocole de routage présente des avantages en fonction des caractéristiques de densité, de mobilité, d'exigences du cahier des charges du système visé ou encore du type de mission. De plus, les simulations présentées dans la littérature, que nous avons recensées dans ce chapitre, ont été réalisées à partir d'hypothèses différentes de celles d'un vrai déploiement d'une flotte de drones. Nous pouvons citer, par exemple, les hypothèses faites sur les modèles de mobilité, les exigences temps réel ou encore le comportement des couches inférieures jugées parfaites et qui ne sont pas représentatives des critères réels pris en compte dans les réseaux UAANET. Pour mieux choisir un protocole de routage qui nous servira de base pour notre contribution scientifique, il nous semble donc nécessaire d'effectuer une analyse de performance de chaque famille de protocoles en considérant l'environnement UAANET. Celle-ci sera discutée dans le chapitre 4.

Chapitre 2

Sécurité dans un réseau ad hoc de drones

Les protocoles de routage UAANET ont été élaborés sur le principe selon lequel les nœuds participent automatiquement à la mise en place du réseau de communication. Cela occulte la considération d'existence de nœuds malveillants. La seule configuration de panne prise en compte est le dysfonctionnement physique d'un nœud conduisant à son arrêt de fonctionnement. Toutefois, les réseaux ad hoc mobiles sont généralement vulnérables à différentes attaques. Cela est généralement causé par l'absence d'une entité centrale dans le réseau. Comme le réseau UAANET fait partie de la famille des réseaux MANET et qu'il est souvent impossible d'imaginer l'absence totale d'entités malveillantes, il est impératif d'assurer des mécanismes de sécurité pour protéger les fonctions et les données sensibles du réseau UAANET. La sécurité de ce réseau est importante, car les flux applicatifs échangés sont d'une importance capitale non seulement pour le besoin de la mission, mais aussi pour les civils (le risque critique serait le détournement d'un drone). À cela s'ajoute la difficulté de dissocier l'action d'un attaquant d'une perte de lien produit par la mobilité des drones. Il est aussi fondamental de considérer les limitations de ressources en matière de mémoire, de bande passante et d'énergie, pour l'implémentation des mécanismes de sécurité. Face à ces constats, il est nécessaire de considérer la sécurité d'un réseau UAANET pour fournir une route fiable et sécurisée.

Contents

2.1 Introduction 42
2.2 Vulnérabilité des réseaux UAANET 43
2.3 Attaques dans les réseaux UAANET 45
2.3.1 Description (modèle) des attaquants 45
2.3.2 Analyse de risques des attaques 46
2.3.3 Attaques liées aux protocoles de routage 48
2.3.4 Analyse des différentes attaques au niveau réseau 53
2.4 Solutions existantes et limites 54
2.4.1 Notions et mécanismes de base de la sécurité 54
2.4.2 Techniques cryptographiques 56
2.4.3 Authentification des nœuds 60
2.4.4 Systèmes de réputation 61
2.4.5 Discussions 62
2.5 Protocoles de routage ad hoc sécurisé existants 63
2.6 Discussions 68
2.7 Conclusions 72

2.1 Introduction

Le chapitre précédent a montré que les protocoles de routage existants proposés pour le réseau UAANET n'ont pas de composante sécurité. Les travaux autour de ce réseau sont encore aujourd'hui concentrés sur l'amélioration de la communication sol bord entre un drone et une station sol et l'optimisation de la communication inter-drones. Cette direction est due aux exigences de qualité de service que doit fournir un réseau UAANET. Ces besoins concernent l'occupation de la bande passante par les paquets de routage et le délai d'établissement d'une route.

Cependant, cet environnement spontané pose plusieurs questions de sécurité. Tout d'abord, en prenant en considération l'utilisation des liens sans fil, le réseau est intrinsèquement vulnérable aux attaques d'écoutes illicites ou encore au déni de service. Ensuite, à cause de l'absence d'une entité centralisée, il n'y a pas de service d'authentification des messages et des nœuds. Il est alors possible d'injecter de faux paquets qui peuvent perturber le séquencement de l'algorithme de routage. Ce genre d'attaque peut corrompre la communication ou diminuer la performance du réseau [BLB02]. La définition des mécanismes de sécurité pouvant identifer ces sources de vulnérabilité doit prendre en compte la limitation des ressources en termes d'énergie, de mémoire et de bande passante. Ces contraintes nécessitent une appréhension de la sécurité fondée sur des mécanismes de sécurité préventive et

résiliants face à une compromission des nœuds du réseau. Vu la sensibilité des applications du système UAS, il est impératif qu'un attaquant n'ait pas accès aux flux applicatifs pour corrompre la viabilité du système UAS [ABC+].

Dans ce chapitre, nous allons présenter une étude sur la sécurité des réseaux ad hoc de drones. Dans un premier temps, nous synthétiserons les causes de vulnérabilité du réseau UAANET. Nous effecuerons une analyse de risques pour identifier les attaques qui apparaissent comme étant les plus susceptibles d'être exécutées sur ce réseau. Ensuite, nous détaillerons les besoins en sécurité du réseau en mettant l'accent sur les fonctions et les données à protéger qui sont véhiculées dans le réseau. Dans la dernière partie, nous présenterons les solutions de sécurité qui existent dans la famille des réseaux MANET. Nous étudierons ensuite leurs applications dans un réseau UAANET. Enfin, nous indiquerons les axes de développement qui nous semblent pertinents pour la sécurité des trafics de routage.

2.2 Vulnérabilité des réseaux UAANET

Les causes de vulnérabilité du réseau UAANET sont, d'une part, intrinsèques à l'environnement du système UAS [KWG+12] ; c'est-à-dire qu'elles sont liées à l'architecture physique des nœuds, des différents liens de communication et à la condition de déploiement du système UAS (par exemple, la nécessité de coopération entre les nœuds). D'autre part, la présence de nœuds malveillants dans le réseau peut également être une source de vulnérabilité du réseau UAANET [JSDA12]. Dans ce qui suit, nous allons analyser ces différents points de vulnérabilité.

1. **Canal de communication vulnérable** : des liens sans fil sont utilisés pour l'envoi et la réception des signaux dans un réseau UAANET. Ces liens radio sont soumis à diverses attaques, comme l'écoute illicite ou l'interférence active [KNN+07]. Généralement, l'écoute d'un réseau ad hoc sans fil consiste à placer un nœud malveillant entre deux ou plusieurs nœuds communicants. Étant donné que l'ensemble du trafic passe dans l'air, il suffit alors à l'attaquant de se positionner dans la zone de couverture des nœuds cibles par intercepter les trafics. Pour écouter l'ensemble du réseau, l'attaquant peut également agir sur le protocole de routage en générant de faux paquets ou en modifiant les paquets de routage. Ainsi, il s'assure que les trafics passent par lui. En particulier dans les réseaux UAANET, la présence d'un attaquant qui utilise une antenne à fort gain à portée d'un drone peut permettre à l'attaquant d'écouter illicitement la communication de toute la flotte. De plus, en fonction de la caractéristique des liaisons de données, les liens sont souvent munis d'une faible capacité de bande passante. Un attaquant peut donc exploiter ces caractéristiques en diffusant des paquets pour saturer le réseau et rompre la communication entre

les nœuds [YDZZ05].

2. **Environnement non contrôlé** : l'environnement d'un réseau ad hoc sans fil est dit distribué et dynamique. Cela signifie que la communication est partagée et opportuniste entre les nœuds souhaitant participer au routage. Il est donc difficile de contrôler l'entrée et la sortie des nœuds dans le réseau. Par conséquent, il est possible pour un nœud malveillant de se connecter au réseau et de participer au transfert des paquets. Il peut aussi usurper l'identité d'un nœud légitime et falsifier le mécanisme de routage émettant des informations erronées ou en rejouant des informations non à jour. C'est le cas notamment de l'attaque *rushing* [HPJ03]. Nous arrivons donc au constat qu'il n'existe aucune protection contre l'intrusion des attaquants dans un réseau UAANET, ceci étant généralement le cas de tous les réseaux ad hoc.
3. **Topologie dynamique** : à cause de la vitesse importante des drones (les drones DT-18 ont une vitesse maximale de 80 km/h), la topologie du réseau n'est pas stable et change continuellement (en fonction des trafics C2 envoyés depuis le sol et du plan de vol des drones). Cette caractéristique engendre des problématiques de sécurité puisque, intrinsèquement, un protocole de routage ne peut différencier une coupure de communication (ou de lien) déclenchée par les mouvements des drones et l'action d'un attaquant dans le réseau [EFL+09]. Ces deux situations peuvent provoquer la déconnexion d'un groupe de nœuds dans le réseau et, au niveau applicatif, les représentations sont identiques. De plus, si le nœud malveillant coopérait partiellement dans le transfert des informations tout en créant des incohérences dans le protocole de routage, son identification nécessiterait des mécanismes d'analyse plus approfondis du réseau en appliquant des métriques de réputation et de confiance [Sen10].
4. **Problème de coopération** : intrinsèquement, dans un réseau ad hoc sans fil, les algorithmes de routage ne requièrent pas d'algorithme de pré-association des nœuds avant l'exécution de l'algorithme de routage. Cela est dû à l'hypothèse selon laquelle les nœuds sont de nature coopérative et non malicieuse. Le scénario où un nœud peut compromettre le réseau n'est donc pas pris en compte. Par conséquent, il n'y a aucune garantie quant à l'authenticité de l'identité des nœuds. Les nœuds malveillants peuvent ainsi se présenter facilement dans le réseau, publiant une route avec une meilleure métrique.
5. **Ressources limitées** : en fonction de leur taille, les drones peuvent avoir des capacités de calcul et de mémoire limitées. Cette caractéristique peut être exploitée par des attaquants pour épuiser les ressources des drones. C'est le cas, par exemple, de l'attaque *sleep deprivation attack* [PZV+06] qui consiste à diffuser en illimité des messages de contrôle vers les nœuds du réseau. Une fois que ces ressources sont épuisées, les drones peuvent être, par exemple, capturés ou détournés par un attaquant. Par ailleurs, l'introduction des mécanismes de sécurité dans le réseau pour

se prémunir des attaques peut diminuer la performance du réseau en augmentant la charge des microprocesseurs [YLY+04]. Il y a donc des compromis à trouver entre la fiabilité du réseau de communication et le choix d'une solution de sécurité.

6. **Existence des attaques** : généralement, les réseaux ad hoc sont la cible d'attaques qui peuvent viser un sous-ensemble de la couche protocolaire du modèle OSI [FTS10]. Ces attaques ont des objectifs qui peuvent être catégorisés en deux classes : les attaques rationnelles et irrationnelles. L'attaquant irrationnel a pour objectif de perturber le fonctionnement des services réseau sans tirer profit des résultats, tandis que l'attaquant dit rationnel vise à élaborer son attaque dans l'objectif de tirer un bénéfice direct ou indirect des résultats de ladite attaque. À titre d'exemple, un des objectifs peut être la violation de la politique de sécurité dans le réseau pour fausser les fonctions et données sensibles du réseau. Les données sensibles du réseau sont les informations relatives au routage, aux configurations pour la gestion d'énergie, à la sécurité (clés cryptographiques, certificats, etc.). La violation de l'intégrité de ces informations critiques peut entrainer la variation de la connectivité du réseau et donc l'échec de la mission.

2.3 Attaques dans les réseaux UAANET

2.3.1 Description (modèle) des attaquants

Pour modéliser un attaquant, il est crucial de quantifier la ressources qu'il possède. En général, plus l'attaquant dispose de ressources, plus la défense est difficile à assurer. Ensuite, la conception des mécanismes de sécurité doit aussi prendre en compte les autres capacités de l'attaquant, comme leur nombre, leur coordination, leur capacité technique et leur intérêt d'influence. En effet, plusieurs attaquants peuvent menacer un réseau au même moment de manière autonome ou en coordonnée, afin de réaliser une attaque commune rendant la défense difficile.

Par ailleurs, l'estimation des capacités techniques des attaquants est importante pour connaitre la nature de leur menace. Par exemple, un nœud malveillant peut seulement recevoir la transmission de données, mais aussi se présenter comme un nœud valide et avoir accès à la totalité des services du réseau. L'attaquant peut aussi avoir accès à tout le réseau par diffusion de faux paquets ou juste à une certaine région du réseau.

Le champ d'action de l'attaque a également son importance. En effet, si le champ d'action est la couche basse, l'attaque s'avère plus dangereuse, car elle affecte les couches supérieures (liaison, réseau et au-delà). Plusieurs services réseau tels que la localisation, la synchronisation temporelle et la gestion d'énergie, peuvent être atteints dans ce cas. Dans le réseau UAANET, les services critiques doivent être davantage défendus que les services optionnels, car le mauvais fonctionnement des composants critiques porte atteinte au fonc-

tionnement de l'ensemble du réseau. Dans notre étude, nous allons prendre en compte les différents critères des attaquants qui menacent le réseau.

2.3.2 Analyse de risques des attaques

Dans cette partie du document, nous allons étudier les différents risques issus des vulnérabilités et menaces s'appliquant au réseau UAANET.

2.3.2.1 Attaques au niveau de la couche physique

Les attaques s'exécutant sur la couche physique concernent les matériels utilisés (par exemple, le modem sans fil utilisé). Ce type d'attaque ne nécessite aucune connaissance de la topologie du réseau ni de la technologie utilisée sur les nœuds [GGG16]. Du point de vue de l'attaquant, ce type d'attaque requiert l'utilisation de matériels spécifiques [WCWC07] qui peuvent perturber les signaux échangés (par exemple, une antenne à fort gain).

Typiquement, les attaques observées à ce niveau sont : l'écoute illicite, l'interférence active et l'attaque *jamming*. Ces attaques sont possibles car les nœuds mobiles partagent le même médium sans fil. Par conséquent, les messages transmis peuvent être aisément entendus, et de faux messages peuvent être injectés dans le réseau. Toutefois, ce type d'attaque est difficilement réalisable dans un réseau à forte mobilité [WCWC07]. En effet, l'attaquant doit rester a portée du nœud cible pour rompre sa communication. Nous avons vu dans le chapitre 1 que, dans le cas particulier des réseaux UAANET, les drones ont des mobilités dynamiques en raison de la succession des commandes envoyées par la station ou d'un plan de vol préchargé. Ce type de *pattern* de mobilité rend difficile pour l'attaquant de se mettre à proximité des nœuds cibles. En raison de ce postulat, nous avons décidé de ne pas considérer ce type d'attaque dans notre étude d'architecture de communication sécurisée.

2.3.2.2 Attaques au niveau de la couche liaison

Pour une meilleure compréhension des attaques qui touchent la couche liaison, nous allons rappeler brièvement les différentes fonctions de cette couche. Les protocoles de la couche liaison ont pour mission de coordonner la transmission sur le médium commun. Il est surtout responsable de la communication entre voisins. Pour cela, le protocole IEEE 802.11 [OP05], qui est souvent utilisé, emploie des algorithmes distribués de gestion de contention pour partager le médium. Il existe deux types d'algorithmes DCF ou PCF dont les caractéristiques sont explicitées dans [ZR03].

Les protocoles de routage et MAC s'exécutent généralement dans l'hypothèse que chaque entité respecte les spécifications du protocole et qu'elles coopèrent entre elles. Toutefois, à partir du moment où la topologie est dynamique, cette hypothèse n'est plus réaliste, car

rien n'empêche un nœud malveillant de compromettre le réseau sur les différents niveaux des couches protocolaires du modèle OSI. Nous pouvons donc dire que la couche IEEE 802.11 est potentiellement non sécurisée puisque aucune entité centrale ne peut surveiller la communication entre les participants. Les attaques à ce niveau sont diverses et variées. Cela peut concerner l'augmentation des délais entre les trames (*Shortest InterFrame Space* : SIFS, *Clear to Send* : CTS, DCF et Polls IFS : PFS) au-delà du seuil acceptable [KV05]. Cela peut aussi être l'épuisement des ressources ou la capacité du canal de transmission par des attaques de déni de service [RMA+08] et l'empêchement de l'accès au canal [RHA04]. Ces attaques ont généralement deux motivations : soit par égoïsme, soit malicieusement dans le but de perturber l'opération de l'algorithme de routage. L'intérêt d'égoïsme n'est pas considéré dans notre travail, car nous faisons l'hypothèse que tous les drones et les stations sol utilisés doivent répondre aux besoins du réseau. Les nœuds sont homogènes et il n'y a aucune raison pour qu'à un moment donné, ils décident de ne pas exécuter la fonction de routage en faisant une retransmission de paquets. En ce qui concerne les classes d'attaques qui découlent des nœuds malicieux, elles ont généralement pour objectif de diminuer la performance du réseau en termes de débit réseau et de disponibilité. Par conséquent, elles empêchent les autres nœuds légitimes de participer au réseau. Ces attaques appartiennent donc à la famille des attaques par déni de service (DoS). Les attaques de déni de service se fondent sur diverses techniques comme, par exemple, la manipulation des intervalles *backoff* [GAY07] qui exploite les vulnérabilités du 802.11 [ZWN04] ou encore les attaques *jelly fish* [NN08]. Différentes approches de détection de déni de service ont été proposées dans la littérature [SZ08] [SCZ11]. La plupart d'entre elles se focalisent sur la détection de manipulation de *backoff* en appliquant des mécanismes de mesure de confiance entre les nœuds ; c'est-à-dire que chaque nœud surveille la valeur de la variable *backoff* de son voisin. La formation de cette chaine de confiance permet d'évaluer la variation de la performance du réseau durant la communication [GA05]. Néanmoins, l'application de ce raisonnement se heurte à quelques limitations. L'une d'elles concerne l'hypothèse faite de l'existence des mécanismes de sécurité d'authentification et d'intégrité dans le réseau [YLY+04] alors que la mise en place d'un tel algorithme relève d'une problématique particulière.

Nous pouvons donc dire que les techniques contre une attaque touchant le niveau MAC se basent généralement sur l'utilisation des chaines de confiance entre les nœuds. Mais pour que ce type d'algorithme puisse fonctionner, il est admis de supposer l'existence d'un protocole de routage fiable et sécurisé assurant l'authentification et l'intégrité des messages. Il est donc compréhensible que la proposition d'une sécurité pour la couche MAC ne puisse pas être uniquement traitée au niveau de la couche liaison. En effet, l'ensemble des solutions que nous avons identifiées dans la littérature mettent en avant à la fois une solution au niveau liaison, mais aussi la nécessité d'une solution au niveau réseau. Par conséquent, nous pouvons décider de ne sécuriser que la couche réseau et, ensuite,

d'apporter des briques de sécurité au niveau liaison. Nous pouvons également traiter les deux thématiques conjointement en adoptant une approche de *cross layer* comme expliqué dans [SZ08].

La suite de ce manuscrit s'appuiera sur ce premier principe. Ainsi, nous ne nous focaliserons que sur la proposition d'un mécanisme d'authentification et d'intégrité au niveau réseau. Cela nous permettra également de nous concentrer sur la protection de la communication multisaut. Pour sécuriser la communication au niveau liaison, des propositions mentionnées dans [BAFF16] peuvent être utilisées, mais ne seront pas traitées dans le cadre de cette thèse.

2.3.3 Attaques liées aux protocoles de routage

La couche réseau permet la connectivité de plusieurs nœuds dans une topologie ad hoc. Cette connectivité est atteinte en communication multisaut avec la collaboration de tous les nœuds du réseau. Toutefois, cette hypothèse de collaboration est difficile à maintenir dans un environnement volatile composé de nœuds ayant des rôles différents. En conséquence, diverses attaques ciblant la couche réseau ont été identifiées et étudiées dans la littérature [KNN+07]. Les objectifs de ces attaques sont divers. Elles peuvent avoir pour buts d'absorber les trafics réseau, d'inclure des nœuds malveillants dans le réseau, de dévier et contrôler les flux réseau. Pour atteindre ces objectifs, plusieurs méthodes peuvent être adoptées par les attaquants. Les trafics réseau peuvent être retransmis sur un chemin non optimal qui peut introduire des délais significatifs par rapport au délai normal de communication. Les paquets peuvent également être retransmis sur des chemins non existants et se perdre. Les attaquants peuvent créer des boucles de routage qui vont causer une congestion complète ou partielle du réseau. Le réseau peut également être partitionné par plusieurs attaquants empêchant un nœud source de trouver une route vers une destination et ainsi dégrader la performance du réseau [KNM08].

Par ailleurs, pour mieux représenter les différentes attaques à ce niveau, des catégorisations peuvent être faites. Par exemple, une manière de les classifier est de se baser sur leur position dans le réseau [DLA02]. Un attaquant peut être localisé à l'extérieur (on parle alors d'attaquant externe) ou à l'intérieur du réseau (attaquant interne). Les attaques externes ont pour but de congestionner le réseau en propageant des informations de routage incorrectes tandis que les attaques internes sont difficiles à contrecarrer puisque les nœuds font déjà partie du réseau. Pour identifier les nœuds malhonnêtes du réseau, il est nécessaire de recourir à des mécanismes de sécurité plus avancés, comme la mesure de confiance ou de réputation.

Par ailleurs, nous pouvons aussi les classifier selon l'action de l'attaquant dans le réseau [DLA02], elle peut être passive ou active. Les attaques passives sont difficilement détectables, car les services réseau restent disponibles. Des baisses de performance peuvent

apparaitre partiellement, mais il est parfois difficile de les différencier spécifiquement. Les attaquants se contentent d'écouter les trafics réseau échangés. Dans ce cas spécifique, l'objectif d'une solution de sécurité serait d'assurer la confidentialité des messages. Pour ce qui est des attaques actives, elles concernent les actions qui influent directement sur les trafics échangés. L'objectif d'une attaque active peut être de dégrader la performance réseau en rejetant ou modifiant des messages de contrôle, d'ajouter du délai de transmission, de modifier la topologie du réseau, en insérant des nœuds compromis ou en invalidant des liens valides par l'envoi de faux paquets.

Nous pouvons également les catégoriser par rapport à la fonctionnalité du routage qu'ils invalident. En effet, les attaques peuvent toucher les différentes phases de routage, à savoir la phase de découverte et de maintenance de route [FTS10]. Les attaques faites sur la phase de découverte de routes ont des impacts plus importants sur l'intégrité du réseau, car leur objectif est d'infiltrer le réseau par l'introduction d'un ou de plusieurs nœuds malveillants. Si un nœud arrive à s'introduire dans le réseau, il peut non seulement porter atteinte aux performances du réseau, mais aussi accéder aux flux applicatifs. Il est donc impératif de sécuriser au maximum le processus de découverte de routes pour assurer l'impartialité du réseau. Pour ce qui concerne la maintenance des routes, les attaques sont conduites pour dégrader la performance du réseau ou pour modifier la topologie. Elles peuvent être prévenues par des techniques de sécurité proactive.

Dans la suite du manuscrit, nous considérerons cette catégorisation afin de sélectionner les attaques qu'il nous semble important de prendre en compte et de résoudre par l'ajout de nouveaux mécanismes de sécurité.

2.3.3.1 Attaques pouvant être exécutées durant la phase de découverte de routes

Selon la stratégie de routage, durant le processus de découverte de route, un émetteur cherche généralement une route vers une destination en diffusant des requêtes. Il s'agit dans ce cas d'une diffusion par inondation pour atteindre tous les nœuds actifs. Par la suite, à travers une méthode proactive ou réactive, une route sera trouvée et sera utilisée pour envoyer les données. Au regard de ce raisonnement, la méthodologie de recherche de route doit être sécurisée pour trouver un chemin fiable et sécurisé. Ces chemins conditionnent ensuite toute la survie de la mission. En conséquence, il faudrait donc qu'un nœud attaquant n'arrive pas à exécuter les actions suivantes durant cette phase : modifier la topologie du réseau, ajouter des nœuds malveillants et falsifier des liens ou des paquets de routage. Les attaques qui peuvent interférer pendant cette phase sont expliquées ci-dessous.

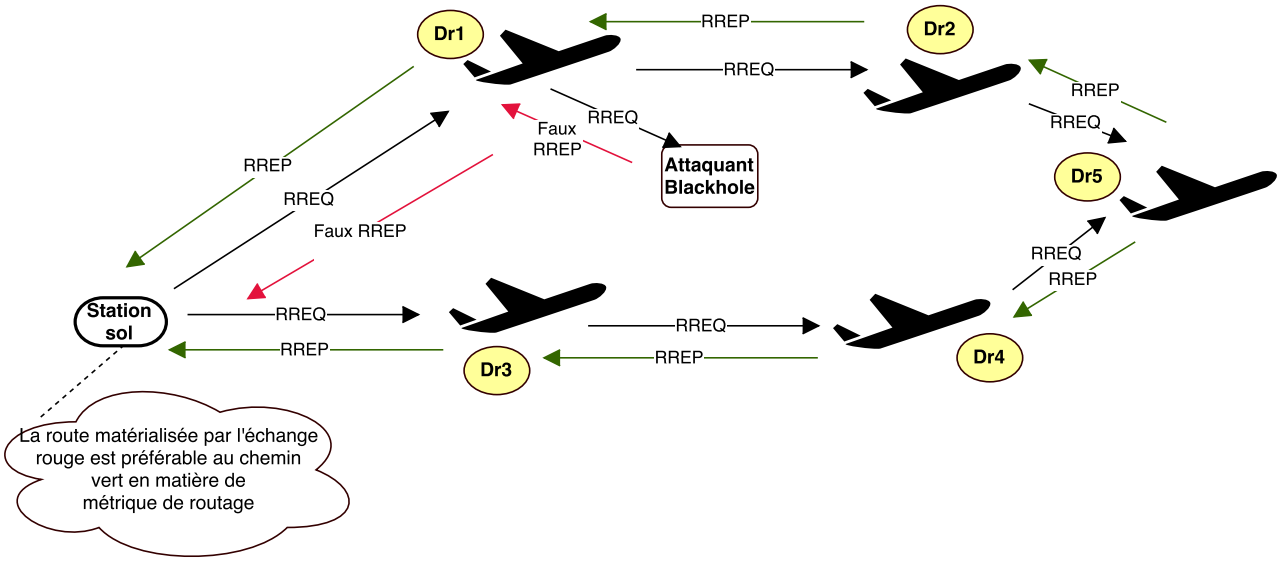

FIGURE 2.1 : Attaque *blackhole*

Attaque *blackhole* : l'attaque *blackhole* [TS07] correspond à une zone formée par un ou plusieurs attaquants dans un réseau. Cette zone fait discrètement disparaitre le trafic sans informer le nœud source. Selon la densité du réseau, cette attaque peut être exécutée par un ou plusieurs nœuds malveillants de manière indépendante (*single blackhole*) ou simultanée (*collaborative blackhole*) [AS16]. Le principe est de faire croire aux nœuds valides qu'il existe une meilleure route vers une destination donnée. Ceci peut être réalisé, par exemple, par injection de faux paquets de contrôle dans le réseau. Si de tels paquets sont traités, un chemin passant par un trou noir sera mis en service. Une autre variante de cette attaque appelée *greyhole* [KS14] existe. Elle consiste à ne pas supprimer tous les paquets, mais à en sélectionner quelques-uns uniquement.

La figure 2.1 illustre un exemple de cette attaque. Sur cette représentation, la station sol essaie d'établir une route vers le nœud Dr5. L'attaquant *blackhole* a la capacité d'usurper l'identité d'un nœud valide du réseau. Il peut alors participer au mécanisme de découverte de route en répondant à la station sol avec un message de type *route reply*. Ce message annonce un chemin, avec un coût minimal, vers le nœud Dr5 (par exemple, avec une valeur de nombre de sauts inférieur aux autres chemins). La station sol mettra alors sa table de routage à jour avec cette fausse route. Les paquets de données transiteront par le nœud malicieux qui pourra tout simplement les ignorer. À cause de cette attaque, les paquets sont captés et absorbés par le nœud malicieux.

L'attaque *wormhole* : l'attaque *wormhole* [MNS08] ou « trou de ver » consiste généralement à placer un tunnel entre deux attaquants (qui se sont entendus) dans le réseau.

Toutefois, il peut exister plus de deux attaquants dans le réseau, créant ainsi plusieurs tunnels en fonction de la densité des nœuds. Dans cette première configuration, le premier attaquant réalise une écoute illicite dans la zone de couverture d'un nœud cible. Ensuite, il retransmet ces paquets à travers le tunnel vers le deuxième attaquant qui peut les réinsérer, avec ou sans modification, dans le réseau. Si les attaquants modifient le contenu des paquets, ils pourront modifier la topologie du réseau si aucun mécanisme d'authentification des messages n'est appliqué. Toutefois, cette attaque est souvent décrite comme exécutée sans modification pour contourner les algorithmes d'authentification. Il faut noter qu'elle est effective contre tout type de protocole de routage, mais elle trouve surtout son efficacité contre les protocoles employant le nombre de sauts en tant que métrique de routage. Une fois que la route passant par les nœuds attaquants est choisie, ils peuvent réaliser d'autres attaques en modifiant ou sélectionnant les paquets. Ce type d'attaque produit une perturbation au niveau du routage des paquets puisque les nœuds distants vont croire qu'ils sont voisins. Les nœuds légitimes, choisissant le chemin le plus court, vont sélectionner le tunnel *wormhole* par défaut, plutôt que d'opter pour un chemin légitime plus long. Cette attaque permet à un attaquant de court-circuiter l'émission normale des messages de routage en créant un tunnel virtuel pour partitionner le réseau. Cette attaque est généralement très difficile à détecter, car elle nécessite l'identification des paquets légitimes ayant été échangés par un lien *wormhole*. Ce type d'information est difficile à obtenir dans un réseau ad hoc mobile.

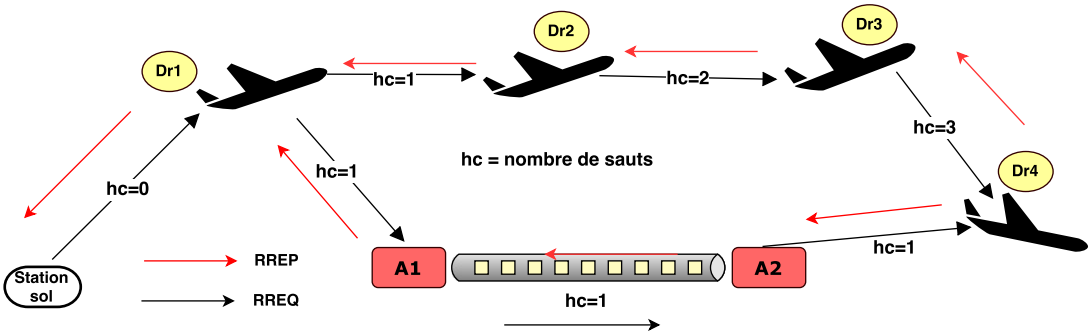

FIGURE 2.2 : Illustration de l'attaque *wormhole*

La figure 2.2 illustre un exemple d'attaque *wormhole* qui donne l'illusion à S et à N3 qu'ils sont voisins. Sur cette figure, A1 va écouter illicitement le nœud S pour recevoir et transférer tous les paquets directement vers A2. En conséquence, N3 va croire que S est à sa portée. Cette attaque va donc transmettre tous les paquets de contrôle échangés (c'est-à-dire les messages Hello, requêtes RREQ, réponses RREP et les messages d'erreurs RERR) depuis le nœud S directement vers N2. Même en n'ayant aucune connaissance des clés cryptographiques ou encore de la fonction de hachage utilisée, les attaquants peuvent dégrader ainsi l'intégrité du réseau en faisant transiter les paquets de contrôle. À la suite

de cette attaque, les attaquants peuvent diminuer la performance du réseau en faisant un tri sélectif des paquets, ou en exécutant des attaques plus complexes, comme l'attaque *rushing*.

Dans la section suivante, nous allons analyser les approches de sécurité existantes pour contrer ce type d'attaque. Le protocole de routage SAODV [Zap02] sera mis en avant pour illustrer les limites du service d'authentification face à l'attaque *wormhole*. Par la suite, une solution de sécurité complète sera apportée au chapitre 4.

Attaque *rushing*

L'attaque *rushing* [HPJ03] vise à augmenter les chances de l'attaquant d'être choisi comme élément d'une route pour joindre une destination donnée. Elle n'est effective que sur les protocoles de routage réactifs, car l'attaque se base sur la diffusion spontanée de paquets de découverte de route.

L'attaque *rushing* consiste à profiter du fait que la plupart des protocoles de routage réactifs se basent sur le principe du « premier arrivé, premier servi » ; c'est-à-dire que lors de la phase de découverte des voisins, la première requête est prioritaire dans le traitement tandis que les autres sont mises en attente ou ignorées. Par exemple, le deuxième paquet arrivé pour une même demande est ignoré automatiquement par l'algorithme de routage réactif, car cela suppose une métrique moins bonne, car plus longue. L'objectif de l'attaquant est donc d'arriver à faire passer ses requêtes avant celles des autres nœuds en se plaçant géographiquement entre deux nœuds communicants. Pour y parvenir, il peut profiter des temporisations avant l'envoi de messages que vont supporter ses concurrents. Ces temporisations concernent : le délai imposé par la couche MAC entre le moment où le paquet est délivré à l'interface pour envoi et le moment où il est effectivement envoyé. Est également concerné le délai généré par le protocole de routage qui permet d'éviter les collisions entre les requêtes.

Attaque de consommation de ressources

L'attaque consiste à consommer les ressources d'un nœud valide. Pour cela, un nœud malveillant diffuse en masse des paquets périmés (non à jour) ou de faux messages. Ces données vont occuper la bande passante. Par exemple, en considérant le protocole de routage AODV, un nœud peut décider de transmettre une grande quantité de requêtes RREQ vers une destination inexistante. Puisque aucun nœud ne va répondre à ces requêtes, ces paquets vont circuler dans le réseau et consommer les ressources réseau en matière de bande passante et d'énergie. Cette attaque peut dégrader la performance du réseau ad hoc en matière de débit de 84 % [DB05]. Une forme de cette attaque est montrée par la figure 2.3. Le nœud intrus envoie des requêtes en continu vers une destination inexistante

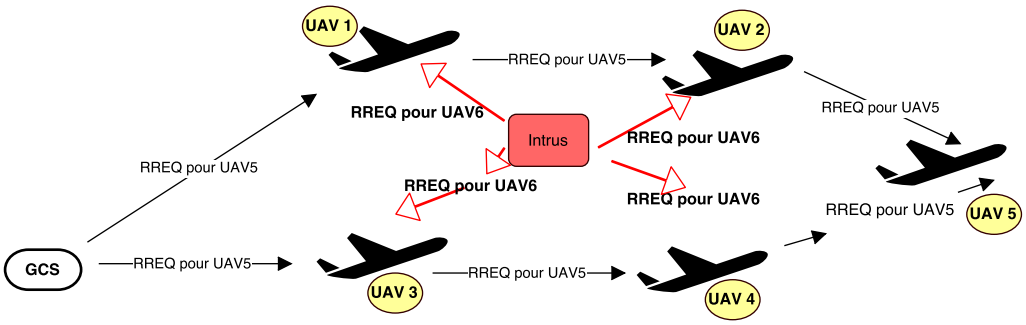

FIGURE 2.3 : Attaque consommation de ressources

vers les nœuds dans sa zone de couverture. Ces nœuds vont ensuite traiter ces messages en transmettant les paquets sans jamais parvenir à un résultat.

Attaque de type rejeu

La topologie d'un réseau ad hoc sans fil est fréquemment modifiée par la mobilité des nœuds. Aussi, dans une attaque de rejeu, les attaquants récupèrent et enregistrent les messages de contrôle échangés valides, et les injectent dans le réseau quand ils ne sont plus à jour. En conséquence, la table de routage ou la table de voisinage (pour les protocoles de routage proactif) sera faussée par des informations non à jour. Cela engendre la création de routes non existantes ou de chemins ayant des métriques inexactes.

2.3.3.2 Attaques pouvant être exécutées durant la phase de maintenance

La phase de maintenance de route consiste en l'échange de messages de contrôle entre les nœuds participants à la formation de la topologie. Les attaques exécutées durant cette étape consistent en la diffusion de faux paquets conduisant à la reconfiguration des routes déjà établies ; par exemple, dans le cas des protocoles de routage réactifs, comme AODV ou DSR, le mécanisme d'échange de messages d'erreur RERR en cas de coupure de lien conduisant à la suppression d'un ou de plusieurs liens. Ce mécanisme peut être exploité par un attaquant en diffusant un faux paquet d'erreurs pour invalider des liens légitimes. Un nœud malveillant peut aussi usurper l'identité d'un nœud valide par son adresse IP ou MAC pour exploiter son identité en diffusant des messages erronés. En outre, l'attaque de type rejeu peut également être réalisée à cette étape pour falsifier le choix d'une route.

2.3.4 Analyse des différentes attaques au niveau réseau

Cette classification des attaques met en évidence la présence de nombreuses attaques dans un réseau UAANET. Pour synthétiser l'objectif et la méthodologie de ces attaques, la

figure 2.4 subdivise les différentes vulnérabilités du routage du réseau UAANET. Nous avons identifié quatre objectifs majeurs (les blocs colorés sont les attaques traitées dans cette thèse) :

1. **La divulgation d'informations de routage** est atteinte en interceptant et en analysant les trafics de routage échangés sur le médium sans fil. Pour résoudre cette vulnérabilité, il faut assurer la confidentialité des paquets de contrôle afin d'en restreindre l'accès. Des approches qui vont dans ce sens ont été proposées dans [AKR07] en utilisant l'association d'une clé privée et d'une clé unique de groupe. Toutefois, les informations de routage en elles-même ne sont pas des informations secrètes. Il est donc secondaire du point de vue de la sécurité de restreindre l'accès aux paquets de routage.
2. **La divulgation d'informations de charge utile** est atteinte lorsque la route qui a été utilisée est corrompue. Cela signifie qu'un attaquant est arrivé à rompre le mécanisme de sécurité mis en place. Il est donc nécessaire d'ajouter une brique de sécurité qui vienne assurer la confidentialité des données. La conception de ce mécanisme passe inévitablement par la protection des données par des moyens cryptographiques et l'éventuelle modification des applications. Ce type de mécanisme de sécurité n'est pas considéré dans ce manuscrit, car il dépasse le contexte des réseaux ad hoc. Nous nous concentrons donc sur la protection de la couche réseau.
3. **La dégradation de performance du réseau** peut être réalisée en rejetant des trafics, en injectant de faux messages, ou en rejouant des messages non à jour.
4. **La modification de la topologie du réseau** est atteinte en invalidant des liens valides (usurpation d'identité) par l'exclusion des nœuds légitimes (attaque *rushing*), en incluant des nœuds malveillants (attaque *wormhole* et *Byzantine*) ou en ajoutant des délais supplémentaires pour l'échange des paquets de contrôle (l'attaque *grey-hole*).

Nous avons donc orienté nos travaux vers les solutions de protection de la couche réseau spécifique, étant donné que la protection des couches inférieures représente des problématiques bien spécifiques qui ont été étudiées dans la littérature. Les solutions passées en revue ci-après concernent les attaques contre le protocole de routage lui-même et/ou les attaques contre la retransmission des paquets de données par les nœuds intermédiaires.

2.4 Solutions existantes et limites

2.4.1 Notions et mécanismes de base de la sécurité

Cette section récapitule les mécanismes de base de la sécurité et les primitives cryptographiques.

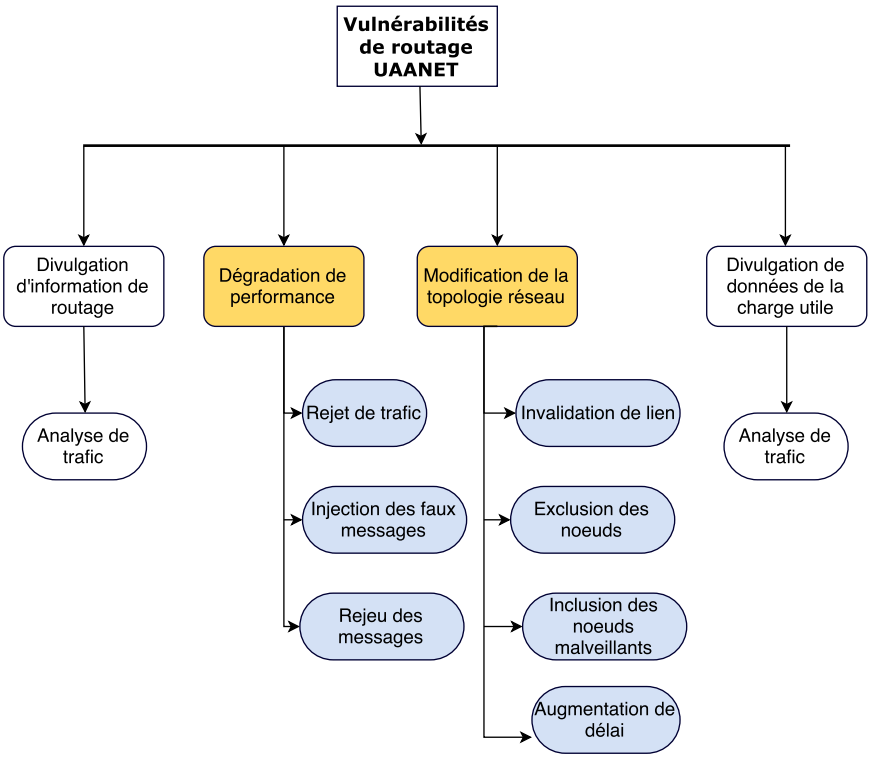

FIGURE 2.4 : Classification des différentes vulnérabilités qui touchent le routage d'un réseau UAANET

Typiquement, les besoins pour la sécurité des réseaux UAANET sont similaires à ceux nécessaires aux autres types de réseaux MANET. Néanmoins, nous pouvons observer une différence au niveau du caractère temps réel et critique des flux applicatifs qui sont échangés. Il faut aussi noter que la mise en place de mécanismes de sécurité qui prendraient en compte un sous-ensemble de ces services de sécurité nécessiterait la prise en compte des besoins applicatifs, notamment en matière de temps, de fiabilité et d'optimisation de l'occupation de la bande passante ; ce qui veut dire que la couche sécurité ne devrait pas pénaliser la performance du réseau. Les longs délais de traitement sont à éviter pour ne pas impacter le délai de transmission. L'utilisation des ressources énergétiques devrait également être optimisée pour éviter des pertes de connectivité. Les services de sécurité nécessaires pour le bon fonctionnement d'un réseau UAANET peuvent donc être définis de la façon suivante :

— Authentification : elle permet de vérifier la véracité de l'identité d'un nœud dans un réseau. Elle est importante lors de la phase de découverte de route pour authentifier les nœuds et les messages échangés. Sans un mécanisme d'authentification, un adversaire peut falsifier les informations de contrôle du routage. Il peut aussi usurper l'identité d'un autre nœud, et ainsi recevoir les flux applicatifs à sa place. Par exemple, un nœud peut se faire passer pour la station de contrôle et recevoir tous

les trafics de la charge utile ;

— Intégrité : elle consiste à vérifier la non-modification des paquets de données durant leur transfert depuis la source vers la destination. Elle assure qu'aucune modification qui pourrait changer la vision qu'ont les nœuds de la topologie du réseau n'a été apportée aux données. Par exemple, en cas d'attaque sur l'intégrité, un nœud attaquant non voisin du nœud cible peut se faire passer pour son voisin direct. La vérification de l'intégrité est faite pour les champs dits mutables, c'est-à-dire des champs qui sont amenés à changer dans le réseau.

— Confidentialité : dans le cas des réseaux UAANET, elle permet de s'assurer que les données temps réel et critique ne sont accessibles qu'aux nœuds éligibles. Il est important de souligner que, généralement, la confidentialité est surtout appliquée pour les trafics applicatifs, contrairement aux informations de routage. En effet, les trafics de signalisations (de routage) ne contiennent pas d'informations secrètes qui nécessitent une grande confidentialité.

— Disponibilité : elle garantit que tous les services réseau du système UAS sont disponibles. C'est un service de sécurité difficile à assumer à cause des contraintes qui pèsent sur les réseaux UAANET. De manière générale, dans un réseau MANET, il est impossible de conserver l'état du réseau comme dans les réseaux filaires. La topologie et l'état du réseau sont amenés à évoluer au cours du temps à cause du mouvement des nœuds.

2.4.2 Techniques cryptographiques

L'objectif fondamental de la cryptographie est de permettre à des entités de communiquer au travers d'un canal peu sûr (par exemple, un réseau de communication) de telle sorte qu'un ou plusieurs attaquants ne puissent interpréter les informations échangées. Elle consiste à appliquer des transformations sur le contenu d'un message à l'aide d'algorithmes de chiffrement (afin de le rendre incompréhensible) et de déchiffrement (afin de reconstruire le message original). Il existe généralement deux techniques de cryptographie : symétrique et asymétrique. Une taxonomie des mécanismes cryptographiques utilisées dans le contexte des réseaux MANET est résumée par la figure 2.5.

2.4.2.1 Cryptographie symétrique et asymétrique

D'une part, la cryptographie symétrique (ou cryptographie à clé secrète) suppose que chaque entité connait la seule clé secrète partagée pour chiffrer et déchiffrer les données. Cette clé identique est partagée préalablement de manière sûre. Le chiffrement symétrique est généralement simple, rapide et efficace, à condition qu'un nœud malveillant ne puisse pas découvrir la clé secrète. Or, avant de pouvoir communiquer ensemble, les deux nœuds

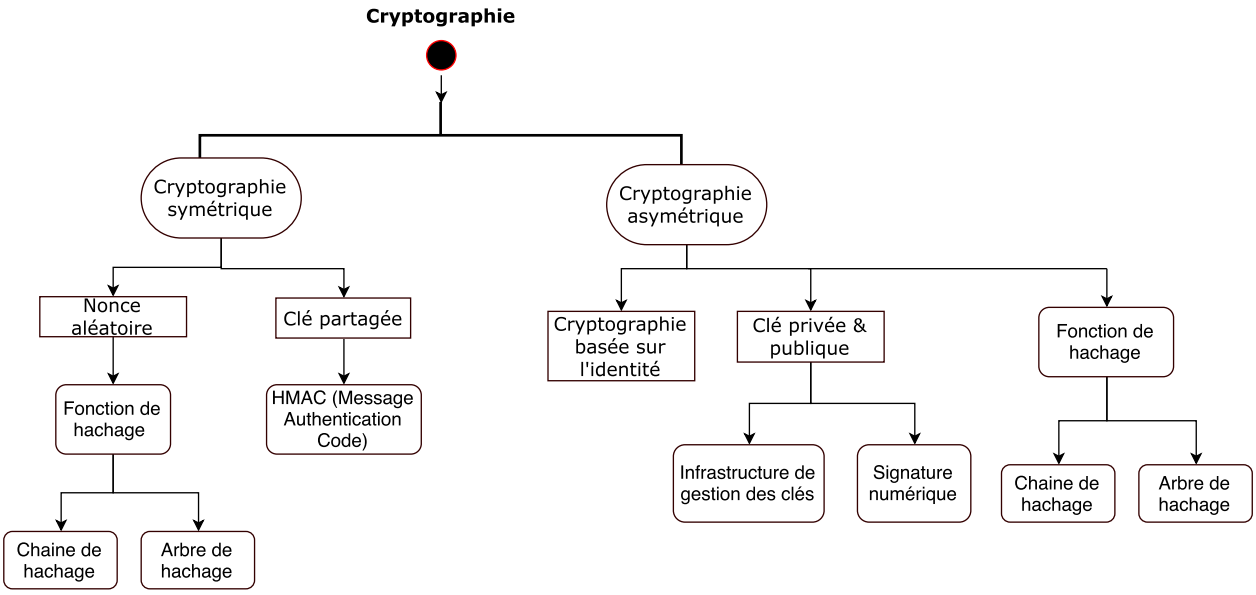

FIGURE 2.5 : Liste des techniques cryptographiques

doivent se mettre d'accord sur la clé. Cet échange initial constitue le principal point faible du cryptage symétrique.

D'autre part, la cryptographie asymétrique (ou cryptographie à clé publique) repose sur l'utilisation d'une clé publique (qui est diffusée) et d'une clé privée (qui est gardée secrète). La première permet de coder le message tandis que la deuxième permet de décoder l'information.

Les protocoles de routage sécurisés sont basés sur des règles d'échange strictes de communication entre les nœuds participants, qui doivent être respectées. La conception des protocoles d'authentification doit être faite avec soin, car des erreurs peuvent conduire à des dysfonctionnements et des conséquences graves sur le niveau de sécurité du réseau.

2.4.2.2 Signatures

La signature numérique est un code numérique associé à un message électronique afin que les nœuds destinataires puissent en authentifier l'origine et en vérifier l'intégrité. Son implémentation fait appel aux fonctions de hachage et de la clé privée du signataire. Les signatures numériques peuvent être vérifiées par un algorithme de vérification à clé publique.

Il existe en pratique de nombreuses catégories d'algorithmes pouvant être utilisés pour procéder à la signature numérique. Par exemple, l'algorithme RSA [NMSK01] est connu pour être robuste. Nous pouvons également citer les algorithmes de Schnorr [Sch91] et ElGamal [ElG84] qui sont basés sur des logarithmes discrets. Bien que tous ces algorithmes

aient des architectures différentes, ils fournissent à peu près tous la même interface d'utilisation. C'est pourquoi, dans ce manuscrit, nous appliquerons l'hypothèse de chiffrement parfait, selon laquelle un texte chiffré ne possède pas d'autre propriété que de pouvoir être déchiffré avec la clé correspondante.

Par ailleurs, il est important de noter que l'utilisation d'une signature numérique dans un réseau spontané comme UAANET nécessite l'utilisation d'une signature de petite taille pour maintenir un *overhead* de communication raisonnable.

2.4.2.3 Authentification des messages

L'authentification des messages consiste à protéger l'intégrité du message et à vérifier que l'information n'a subi aucune modification par un attaquant. Elle permet également de vérifier l'identité de l'émetteur et la non-répudiation de celui-ci. Pour réaliser cette opération, il faut un authentificateur, une signature ou un code d'authentification de message (Message Authentication Code : MAC). Ces condensés doivent être envoyés avec le message. Le MAC est généré à travers un algorithme qui dépend à la fois du message et d'une clé particulière qui peut être privée ou publique et qui est connue seulement de l'émetteur et du récepteur. La taille du message peut être variable, mais, dans la plupart des cas, un MAC possède une taille fixe.

2.4.2.4 Fonctions de hachage

Une fonction de hachage cryptographique est une procédure déterministe qui compresse un ensemble de données numériques de taille arbitraire en une chaîne de bits de taille fixe qu'on appelle l'empreinte, le condensé ou la valeur de hash *(hash value)*.

Nous pouvons illustrer le MAC comme exemple d'une fonction de hachage à sens unique qui produit une empreinte dépendant à la fois du message et de la clé. Le terme « unique » implique la propriété d'évitement de collision entre le condensé et la fonction de hachage. C'est-à-dire qu'en supposant un condensé y, la valeur du message x doit être difficilement déduite telle que $H(x) = y$.

Par conséquent, une fonction de hachage acceptable h doit avoir les propriétés suivantes pour ne pas affaiblir la sécurité du procédé de signature :

— La fonction h doit détruire toute structure homomorphique, par exemple, être incapable de calculer le condensé des deux messages combinés en connaissant leur valeur de hachage individuelle ;

— La fonction h doit être appliquée sur le message en entier ;

— La fonction h doit être à sens unique pour éviter la déduction d'un message à partir de son condensé ;

— La fonction h doit être suffisamment robuste pour résister à l'attaque des anniversaires [Kir11] qui consiste à calculer deux messages avec une même empreinte.

Par conséquent, nous pouvons noter qu'une fonction de hachage se doit d'être faiblement sensible aux collisions et à sens unique. Une fonction de hachage h est faiblement sensible aux collisions si, étant donné un message x, il est difficile par calcul d'obtenir un message $x^{'} \neq x$ tel que $h(x^{'}) = h(x)$.

La structure de conception d'une fonction de hachage est un procédé complexe dont le principe de base dépend de l'utilisation répétitive d'une fonction de compression qui prend en entrée à la fois le bloc de message et la chaine de valeur de l'étape précédente. Typiquement, l'un des obstacles à la création d'une telle fonction est le choix de la fonction de compression pour résister à la collision entre deux messages possédant la même valeur de hachage. En effet, la probabilité d'existence d'une collision est non nulle. C'est donc la robustesse de la fonction contre le calcul de message à partir du condensé qui importe. Différentes fonctions de hachage sont citées dans la littérature. Les familles MDs (MD2, MD4, MD5) [Riv92], le Secure Hash Algorithm (SHA) [ErJ01] et HMAC [MG98].

Par ailleurs, il existe plusieurs manières d'implémenter une architecture de hachage. Par exemple, nous pouvons distinguer l'arbre de hachage ou arbre de Merkle [Szy04] qui est une structure de données utilisée dans les réseaux pair-à-pair pour assurer l'intégrité des données. Elle peut également être utilisée avec les fonctions de hachage pour l'authentification et la distribution de la chaine des clés. Le principe consiste à diviser les données d'entrée en un ensemble de blocs de taille identique. Ces blocs sont alors considérés comme les feuilles de l'arbre. Ils peuvent être additionnés en taille avec des valeurs neutres, comme des zéros, de façon à obtenir des blocs de la taille souhaitée.

Par ailleurs, il est également possible de construire nouvelle fonction de hachage basée sur une composition de fonctions pour augmenter le niveau de robustesse.

2.4.2.5 Certificat

C'est une donnée numérique qui atteste que l'identité d'une entité donnée est liée à une seule clé publique. Elle joue ainsi le rôle de passeport numérique. Les certificats sont signés et transmis de manière sécurisée par un tiers de confiance appelé « Autorité de certification (AC) ». Le certificat utilise deux mécanismes : le premier associe une clé publique à une identité, le deuxième utilise une signature numérique pour garantir l'unicité de ce lien clé publique/identité ainsi que sa validité. Tant que les nœuds ont confiance en cette autorité, ils peuvent être assurés que les utilisateurs de ces clés en sont bel et bien les propriétaires.

En disposant d'un certificat au lieu d'une clé publique, le destinataire peut vérifier un certain nombre d'aspects au sujet de l'émetteur pour s'assurer que le certificat est valide et qu'il appartient bien au nœud légitime. Il peut notamment comparer l'identité du nœud

propriétaire, vérifier que le certificat est toujours valide, vérifier que le certificat a été signé par une AC de confiance, et vérifier l'intégrité de la signature du certificat de l'émetteur.

Par ailleurs, dans un réseau classique où l'autorité possède une position fixe, le certificat X.509 [HPFS02] est utilisé dans le cas où il y a une association entre l'identité du nœud et sa clé publique. Pour vérifier le certificat, l'identité de l'autorité est alors contrôlée avec sa date d'expiration et la signature numérique de l'autorité de certification. Dans un réseau spontané, comme le réseau ad hoc de drones, il n'est pas possible de fournir une architecture hiérarchique et centralisée capable de gérer l'horodatage et l'enregistrement des identités (les certificats X.509 sont donc inexploitables). Par conséquent, d'autres classes de certificats ont été définies dans la littérature (par exemple, MOCA [YK04]). Ils se caractérisent par l'association d'une identité à base d'adresse IP ou d'adresse MAC avec la clé publique. Ce genre de certificat permet une validation de l'identité sans un tiers de confiance. Les nœuds du réseau sont donc responsables de l'authentification de leurs voisins.

2.4.2.6 Infrastructure à clés publiques

C'est une combinaison d'algorithmes offrant une méthodologie de gestion efficace des clés et des certificats. Cette infrastructure doit notamment gérer la création des clés et des certificats, la protection des clés privées, la révocation de certificat (par exemple la gestion de la situation quand la clé privée d'un nœud est compromise), l'enregistrement et la récupération des clés, la mise à jour des clés et des certificats,la gestion de l'historique des clés et la gestion d'accès aux certificats. Dans la suite de ce manuscrit, nous supposerons l'existence d'un système de gestion de clés dans le réseau. En effet, notre contribution principale se porte sur la sécurisation du routage ad hoc qui nous permet de garantir l'authentification et l'intégrité des messages. Cette problématique sera examinée dans la partie perspective du manuscrit.

2.4.3 Authentification des nœuds

Pour vérifier l'identité d'un nœud dans un réseau MANET, il est possible d'utiliser trois méthodes :

1. L'utilisation du protocole de Key agreement : c'est-à-dire la distribution de la clé de manière dynamique durant l'échange. Il se base sur le concept de la cryptographie asymétrique où tous les participants sont dotés d'un couple de clé publique/clé privée. Parmi les protocoles utilisant ce principe, nous pouvons citer le modèle de Diffie-Helman (DH) [BCPQ01] et le modèle Encrypted Key Exchange (EKE) [AP05].
2. L'utilisation du modèle *Duckling Security* pour établir une relation de type maitre-esclave : dans ce modèle, un nœud est marqué *imprinting*, par une entité centralisée.

Une clé secrète est échangée entre les deux entités à travers un canal supposé sûr. La gestion de clés se fait par l'entité centrale qui liste les paramètres sous son contrôle associés à l'identité de son propriétaire et de la clé correspondante. Cette solution a l'avantage de donner l'autorisation aux nœuds de gérer le cycle de vie des clés, néanmoins, elle implique un fort risque de violation des différents types d'informations applicatives en cas d'attaque sur l'entité centrale.

3. L'utilisation d'une infrastructure à clé publique auto-organisée : dans cette méthode, chaque nœud établit des certificats pour les nœuds en qui il a confiance. Lorsque deux entités du réseau veulent communiquer sans connaissance préalable, ils s'échangent leur liste de certificats et créent une chaine de confiance. Ce troisième axe est souvent utilisé pour l'authentification des nœuds au sein d'un réseau MANET, car il s'affranchit du besoin d'une entité centrale de certification.

2.4.4 Systèmes de réputation

Les objectifs des systèmes de réputation peuvent être résumés par les trois points suivants :

— Fournir des informations permettant de reconnaitre les nœuds de confiance ;
— Donner des points positifs aux nœuds qui ont un comportement vertueux pour les encourager à agir de manière fiable ;
— Déterminer et isoler les nœuds non fiables.

Chaque nœud calcule les degrés de confiance de chaque nœud rencontré à partir des techniques de détection d'intrusion. Ce degré de confiance est recoupé avec la réputation captée au cours du temps. Ce degré de confiance reflète le comportement global des nœuds.

Typiquement, pour sécuriser un réseau MANET, il est possible de choisir entre un algorithme de prévention ou de détection. L'approche de prévention consiste généralement à proposer des mécanismes de sécurité garantissant l'authentification, la confidentialité, l'intégrité et la disponibilité. En ce qui concerne l'approche de détection, elle se base sur la surveillance de comportement des nœuds et la mesure de leur réputation. Ce type d'approche s'applique si un environnement composé de nœuds hétérogènes est considéré. Le but est alors de s'assurer que les nœuds participent convenablement aux règles du réseau et qu'ils ne se contentent pas de l'exploiter, voire d'y nuire. Généralement, nous distinguons deux comportements pouvant compromettre le réseau : l'égoïsme et la malveillance. L'égoïste est celui qui se contente de préserver des ressources (en bande passante, en énergie, etc.) sans rien partager alors que le nœud malveillant injecte de faux paquets pour exclure des nœuds légitimes ou porter atteinte à la performance du réseau. Ces deux actions sont appliquées à condition que les nœuds égoïstes ou malveillants fassent déjà partie du réseau. Dans notre cas d'étude, nous faisons l'hypothèse que les nœuds sont homogènes et qu'ils n'ont pas un comportement égoïste. Nous supposons qu'ils sont conformes aux

comportements pour lesquels ils ont été définis. Par conséquent, un nœud légitime ne peut donc devenir égoïste ou malveillant. Dans ce cas, les systèmes de prévention peuvent suffire pour empêcher un attaquant de se joindre au réseau.

Également, les systèmes de réputation sont difficilement applicables dans les réseaux ad hoc à forte mobilité puisque le mouvement des nœuds rend le temps d'interaction (nécessaire pour l'échange d'observation) très court et les interactions répétées très rares [MGFM06].

2.4.5 Discussions

Nous allons synthétiser les différentes méthodes de sécurité qui ont été mentionnées dans cette section. Tout d'abord, en ce qui concerne les moyens pour assurer l'intégrité et l'authentification des messages, nous pouvons identifier deux approches : l'utilisation des signatures numériques ou l'utilisation des codes d'authentification des messages (MAC). Le MAC est une fonction de hachage à sens unique qui dépend d'une clé secrète. L'inconvénient du MAC est le partage préalable des clés secrètes entre les nœuds mobiles susceptibles de vouloir communiquer. Notre choix s'est donc tourné vers l'utilisation d'une signature numérique qui s'appuie sur une approche de cryptographie à clé publique. Bien que l'utilisation de ce type de mécanisme demande plus de temps et d'exigences en termes de charge de traitement, l'utilisation des drones ayant des puissances de calcul confortables (les valeurs précises seront abordées dans le chapitre 4 qui suit) nous permet de ne pas considérer cette limitation. Il est aussi intéressant dans certains cas d'utiliser un autre mécanisme pour vérifier l'intégrité des messages qui sont modifiées fréquemment. Par exemple, le nombre de sauts, comme nous l'expliquerons dans la section suivante.

En ce qui concerne la confidentialité des paquets de routage, nous ne pouvons pas la considérer comme un critère de sécurité important puisque ce type de donnée n'est significatif qu'en temps réel et leur réinsertion dans le réseau est déjà examinée par le système d'horodatage existant dans le protocole de routage. Toutefois, il est important d'assurer la confidentialité des flux applicatifs qui peuvent être victimes d'attaques de type interception. Ce type de mécanisme n'est pas dépendant de l'environnement ad hoc ni du réseau UAANET, et des solutions existantes peuvent être adaptées.

Par ailleurs, en ce qui concerne la définition d'une infrastructure à clés publiques, ce type de système est clairement fondamental pour la génération et la gestion des clés. Des propositions d'architectures employées dans le domaine MANET ont été analysées dans [CSC11] et nécessitent des études plus approfondies. Dans la suite de ce manuscrit, nous supposerons que ce genre de système existe de façon fiable dans le réseau UAANET. En effet, le cœur de la contribution de cette thèse (qui sera développé dans le chapitre 4) porte sur la définition d'un protocole de routage sécurisé. Pour mener à bien ce travail, il nous est nécessaire de faire un certain nombre d'hypothèses sur certains mécanismes de sécurité qui doivent être initialement présents dans le réseau UAANET, l'infrastructure

à clés publiques en fait partie et ne fera pas l'objet d'une contribution spécifique dans le chapitre suivant.

2.5 Protocoles de routage ad hoc sécurisé existants

Dans ce qui suit, nous allons étudier les principaux protocoles de routage sécurisés en environnement ad hoc de la littérature. Nous nous focaliserons sur les algorithmes proposant des méthodologies de prévention empêchant un nœud malveillant de rejoindre le réseau.

Les travaux de recherche existants autour de ce sujet s'intéressent généralement à l'établissement des clés cryptographiques afin d'assurer l'authentification des messages. Plusieurs propositions jugées commes pionnières (à cause de leur standardisation) ont été proposées dans la littérature. Nous pouvons citer les trois algorithmes de routage MANET qui ont servi de base à la plupart des travaux de routage sécurisé : OLSR, AODV et DSR.

Protocole SAODV

Le protocole SAODV (Secure AODV) [Zap02] a pour objectif de garantir l'authentification des messages. Les champs non mutables des paquets de contrôle (par exemple, l'adresse IP source et destination) sont signés successivement par les nœuds intermédiaires et les signatures sont authentifiées par tous les nœuds. Les parties mutables (par exemple, le nombre de sauts) qui sont considérées comme plus sensibles ne peuvent pas être vérifiées par une signature à cause du temps de traitement que cela rajouterait. Il faut aussi prendre en compte le concept de charge réseau puisque vérifier une signature à chaque retransmission augmente l'*overhead* en sécurité. Par conséquent, le principe de chaîne de hachage [HP04] est utilisé pour s'assurer que le nombre de sauts n'a pas été modifié par un attaquant.

Il est à noter que le protocole SAODV est utilisé avec la présence d'une infrastructure de gestion de clés pour gérer le cycle de vie des clés entre les nœuds. L'algorithme de signature des messages nécessite que les nœuds disposent d'une paire de clés pour appliquer un chiffrement asymétrique. De plus, les nœuds du réseau doivent aussi connaitre la clé publique des autres nœuds du réseau.

Le protocole SAODV est robuste contre plusieurs attaques. Ces attaques consistent à modifier des paquets légitimes et à fabriquer des paquets non conformes aux règles de routage. Il peut également faire face aux attaques de vol d'identité. L'objectif principal de SAODV étant de s'assurer que des nœuds malveillants n'arrivent pas à se faire passer pour des nœuds valides et puissent ainsi interagir dans le réseau avec des identités usurpées. Les nœuds intermédiaires ne doivent pas également pouvoir modifier les paquets en transit. Cette propriété permet de contrer un certain nombre d'attaques, dont l'attaque *blackhole* expliquée précédemment.

Cette protection n'est toutefois pas totale et il est clair qu'un attaquant peut ne pas incrémenter le nombre de sauts. La valeur du champ *hash* reste alors inchangée, ce qui ne permet pas au nœud légitime de détecter le comportement malicieux. C'est le cas notamment de l'attaque *wormhole*.

Dans le chapitre 4, nous ferons une étude analytique du protocole SAODV qui permettra de mettre en évidence formellement les lacunes de ce protocole et proposerons une solution originale de protocole de routage sécurisé.

Protocole SPREAD

Le protocole SPREAD (Secure Protocol for REliable dAta Delivery) [LLF04] a pour objectif de transformer un message entier en plusieurs parties. Ces parties sont ensuite envoyées sur différents chemins. Cela permet de considérer le cas où un groupe de nœuds est compromis et, dans ce cas, l'information dans son intégrité ne sera pas compromise. Le mécanisme de SPREAD est montré par la figure 2.6. Sur cette figure, les trafics de routage sont envoyés sur trois chemins différents. Le nœud destination décide si les différents bouts de message qui sont arrivés sont recevables. Pour cela, il se base, par exemple, sur le niveau de sécurité spécifié et la disponibilité de plusieurs chemins. De plus, les chemins peuvent être modifiés pour éviter l'identification de *pattern* de ces chemins. Pour que l'algorithme SPREAD puisse fonctionner, il faut que la source et la destination soient sécurisées et de confiance. Ce genre de scénario peut ne pas fonctionner dans le cas où les deux protagonistes (source et destination) sont eux aussi des nœuds mobiles et qu'ils sont localisés dans une zone non sécurisée (ce qui est le cas du réseau UAANET). En conséquence, les messages peuvent encore être divulgués par attaque sur la destination. La confidentialité n'est donc pas totalement assurée.

Protocole SRP

Le protocole SRP (Secure Routing Protocol) [PH02] est un protocole de routage sécurisé réactif qui se base sur le protocole DSR. Il sécurise principalement la phase de découverte de route en contrant les fautes byzantines isolées [ACH+04]. Les fautes byzantines comprennent l'ensemble des actions provoquées par le non-respect de la spécification du réseau et qui provoquent des résultats non conformes à ce qui était attendu. Les pannes visent à créer des ruptures temporaires du réseau, à traiter des messages corrompus ou encore des arrêts inopinés des services fournis par le réseau. Le protocole SRP suppose que les nœuds réalisent une association de sécurité préalable entre eux en générant une clé de chiffrement symétrique pour chaque flux de communication. Cette clé est ensuite utilisée pour signer le message en générant un HMAC. Les paquets de découverte de route ne sont signés que par la source et la destination pour authentification. Les nœuds intermédiaires ne peuvent

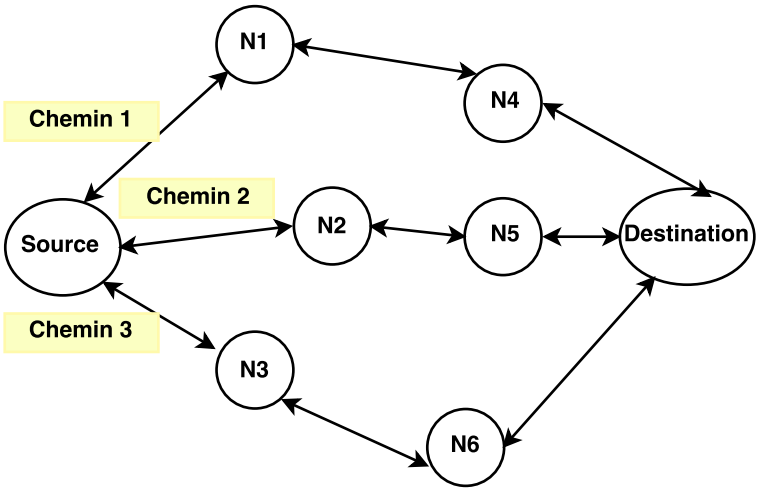

FIGURE 2.6 : Envoi de données sur plusieurs chemins

donc pas signer les messages. Ils ne peuvent qu'ajouter leur adresse ou leur identité dans la liste des nœuds ayant traité le paquet. Toute signature est vérifiée par la source et la destination et tout échec de vérification entraine automatiquement la destruction du message correspondant.

Toutefois, les adresses des nœuds ne sont pas protégées contre des manipulations frauduleuses. Pour traiter ce point de vulnérabilité, le protocole SRP utilise la multiplicité des routes entre la source et la destination pour trouver au moins une route non malveillante selon un critère donné. Toutefois, ce mécanisme exclut l'établissement de l'information dans la mémoire cache des nœuds intermédiaires à des fins d'optimisation (comme c'est déjà le cas pour les protocoles DSR et AODV). Les paquets de requêtes doivent arriver à destination avant que des réponses puissent être générées (puisque elles doivent être signées). Par ailleurs, il est aussi important de noter que, dans le protocole SRP, les nœuds échangent des paquets de maintenance de route pour signaler une perte de lien. Or, ce type de paquet n'est pas protégé. Ainsi, un attaquant peut créer un faux paquet publiant une rupture de lien fictive.

Protocole SEAD

Le protocole SEAD (Secure Efficient Ad hoc Distance vector) [HJP03] se base sur le protocole DSDV. Il protège contre les attaques par modification du numéro de séquence et du nombre de sauts dans les messages de mise à jour. Pour cela, il se base sur l'utilisation d'une fonction de hachage à sens unique. À chaque saut, chaque nœud calcule une suite

de hachés (h0, h1, h2, . . ., hn) en appliquant une fonction de hachage H. Le protocole SEAD vérifie l'intégrité des messages pour différencier les mises à jour provenant des nœuds légitimes des paquets venant des attaquants. Cela permet de minimiser l'impact de l'attaque de type consommation de ressources.

Cependant, SEAD ne considère pas la modification d'autres champs, comme le prochain saut ou la destination. Il ne protège pas non plus contre la fabrication et l'envoi d'un nouveau message de mise à jour à destination d'un autre nœud en utilisant la même métrique et le même numéro de séquence qu'un message de mise à jour récent. Ainsi, un nœud malhonnête peut modifier les champs qui ne sont pas protégés ou fabriqués, et injecter de faux messages.

Protocole ARIADNE

Le protocole ARIADNE [HPJ05] est une extension du protocole DSR. Il assure une authentification point à point des messages de routage en utilisant les fonctions de hachage à clé secrète (HMAC Hash-based Message Authentication Code). Pour assurer une authentification sécurisée, ARIADNE se base sur TESLA [PT03] qui est un protocole permettant l'authentification sécurisée lors des diffusions. Chaque nœud a une clé secrète qui lui permet de calculer une chaine de hachés qui sera utilisée de la manière suivante : lorsqu'il transmet un message de demande de route (Route Request), le nœud lui ajoute un HMAC calculé sur l'ensemble du message avec le dernier haché encore non utilisé de la chaine. Si un message de réponse de route (Route Reply) passe par le nœud, celui-ci révèle la pré-image de la valeur qu'il avait utilisée dans le message Route Request. Lorsque tous les nœuds sur le chemin réalisent cette opération, le chemin est authentifié. Pour fonctionner, ARIADNE requiert que tous les nœuds du réseau soient synchronisés (suite à l'utilisation de TESLA) et que chacun d'entre eux connaisse la dernière valeur de la chaine de hachés de tous les autres. Cette extension sécurise le protocole contre les attaques par modification et par fabrication, mais reste encore vulnérable aux comportements égoïstes et à l'attaque *wormhole*.

Protocole SEAR

Le protocole SEAR (Secure Efficient Ad hoc Routing) [LZW+09] se base sur le protocole AODV. Il utilise la cryptographie symétrique durant la communication et la cryptographie à clé publique à l'initialisation pour partager les clés utilisées. Il garantit l'authentification des messages à travers un ensemble de fonctions de hachage à sens unique pour construire une chaine de hachés appelés authentificateurs (en anglais *authenticators*). Ces valeurs de hash sont par la suite associées à chacun des nœuds dans le réseau pour assurer l'authenticité des messages de contrôle.

Dans SEAR, les nœuds intermédiaires peuvent vérifier les authentificateurs en leur appliquant la fonction de hachage à sens unique utilisée. Par ailleurs, pour éviter qu'un nœud malveillant diffuse un message sans accroitre le nombre de sauts, l'identité du nœud est prise en compte dans l'encodage du hash.

Ce protocole est robuste contre les attaques de modification des paquets et de génération de faux paquets. Toutefois, il reste vulnérable aux comportements égoïstes et à l'attaque *wormhole*. En effet, deux ou plusieurs attaquants peuvent créer un tunnel dans le réseau pour falsifier le choix d'une route et capturer les trafics.

Protocole SOLSR

Le protocole SOLSR (Secure OLSR) [ACJ+03] est une extension sécurisée du protocole OLSR. Il a pour objectif de proposer un mécanisme d'authentification de paquets de contrôle pour protéger contre l'attaque de rejeu. Pour cela, un mécanisme HMAC est utilisé de saut en saut. Les messages de contrôle sont aussi horodatés pour éviter le rejeu d'anciens messages. L'hypothèse faite ici est que tous les nœuds légitimes du réseau ont accès à la même clé pour signer et vérifier l'intégrité des messages.

Ce protocole a pour inconvénient d'ajouter des *overhead* supplémentaires au protocole OLSR. Également, en cas de forte mobilité, la performance de SOLSR tend à décroitre à cause de l'incohérence entre la position des nœuds et l'information inscrite dans le paquet. Toutefois, son avantage réside dans l'utilisation d'une cryptographie symétrique qui engendre moins de calcul processeur.

Protocole ARAN

Le protocole ARAN (Authenticated Routing for Ad hoc Networks) [SLD+05] est un protocole réactif proposant l'authentification des paquets de découverte et de maintenance de route. Il propose également un service de non-répudiation en utilisant des certificats pré-établis distribués par un serveur de confiance. Pour participer aux processus de routage, un nœud doit demander un certificat à ce serveur contenant son adresse IP, sa clé publique et un temps d'expiration. Ce certificat est utilisé pour s'authentifier auprès des autres nœuds. Quand un nœud souhaite initier une communication vers une destination, il crée et envoie un paquet de découverte de route (Route Discovery Packet : RDP) contenant son certificat, un nonce et une variable de temps d'expiration. Il signe ensuite avec sa clé privée et l'envoie aux nœuds intermédiaires. Chaque nœud transmettant un message de demande de route doit le signer, ce qui augmente la taille des messages de routage à chaque saut.

Ce protocole a l'avantage de prévenir contre les attaques de modification des paquets de routage. Il permet également de contrer les attaques de type rejeu. Toutefois, l'utilisation

d'un serveur centralisé de certificats constitue une menace en cas de vulnérabilité du serveur. Dans une telle situation, un nœud attaquant peut acquérir une clé qui est valide et qui est acceptée par les nœuds du réseau. Ce type de système est également difficile à déployer lors de la mission d'une flotte de drones.

Un inconvénient de cette méthode est également qu'elle utilise l'authentification saut par saut en vérifiant à chaque fois le certificat, ce qui augmente considérablement le calcul au niveau de chaque nœud ainsi que la taille des messages, ce qui consomme plus de bande passante.

2.6 Discussions

Protocole	Service de sécurité	Mécanismes de sécurité	Avantages	Inconvénients
SAODV	- Authentification - Intégrité - Non-répudiation	- chaîne de hachage - Signature numérique	- Sécurise les différents paquets de contrôle : RREQ, RREP, RERR, Hello - Robuste contre les attaques de type modification (*blackhole, greyhole et sinkhole*) - Sécurise les champs mutables et non mutables	- vulnérable contre l'attaque *wormhole* et *rushing*
SRP	- Authentification - Intégrité	- HMAC	- Sécurise les modifications et les falsifications des messages	- Ne sécurise pas les paquets de maintenance de route - vulnérable contre l'attaque *wormhole* et *rushing*
ARAN	- Authentification	Serveur de certificats	- Contre l'attaque spoofing, rejeu et modification des messages de routage	- Se base sur l'utilisation d'un serveur centralisé qui n'est pas possible en UAANET - Vulnérable contre l'attaque *wormhole* et *rushing*.
ARIADNE	- Authentification - Intégrité	- chaîne de hachage - TESLA - Signature numérique	- Authentification des nœuds sur la route - Contre l'attaque de type modification	- La synchronisation par TESLA peut être non fiable en UAANET - Vulnérable contre l'attaque *wormhole* et *rushing*
SOLSR	- Authentification - Intégrité	- HMAC	- Contre l'attaque rejeu (*replay attack*)	- Rajoute de l'*overhead* en cas de forte mobilité. - Incohérence entre la position des nœuds et l'information dans le paquet.
SEAR	- Authentification - Intégrité	- chaîne de hachage	- Sécurise les modifications et falsification des messages	- vulnérable contre l'attaque *wormhole*
SEAD	- Authentification	- Fonction de hachage à sens unique	- Contre l'attaque de déni de service et la consommation de ressources	- vulnérable en présence de plusieurs attaquants : par exemple, l'attaque *wormhole*, *blackhole* collaboratif
SPREAD	- Authentification - Intégrité - Confidentialité	- Métrique de confiance - Routage multi-chemins	- Si la source et la destination sont sécurisées, SPREAD permet de sécuriser les données - Partage de données sur plusieurs liens	- Vulnérabilité de tout le réseau si la destination est attaquée. - vulnérable contre l'attaque *blackhole*, *wormhole* et *rushing*

Tableau 2.1 : Synthèse des protocoles de routage sécurisé existants

Les protocoles de routage sécurisés cités précédemment sont synthétisés et comparés dans le tableau 2.1. Pour distinguer un protocole, il est judicieux de considérer les différents paramètres suivantes.

— Adaptation au changement de topologie : comme énoncé dans le chapitre précédent, le réseau UAANET se distingue en particulier par la fréquence de changement de topologie induite par la vitesse de déplacement des drones. Il est donc préférable d'adopter un protocole ayant une bonne adaptation aux différents scénarios de mobilité et qui soit, en même temps, capable de maintenir une performance acceptable (taux de perte de paquets, délai de transmission, délai de rétablissement d'une route et overhead).

— *Overhead* généré : la question porte sur l'occupation en bande passante de l'entête de la partie sécurité du protocole de routage, l'objectif étant que l'*overhead* généré ne pénalise pas l'échange des flux applicatifs.

— Capacité de mémoire et de traitement : cette métrique permet d'associer l'approche choisie avec l'exigence induite en matière de capacité de mémoire et de traitement par les processeurs. Le but est d'adopter une approche non gourmande de ces ressources.

Pour considérer l'application de ces approches de sécurité de routage ad hoc existant dans un réseau UAANET, il est important de les comparer en fonction des métriques énumérées ci-dessus. Nous posons donc les questions suivantes :

1. L'objectif de sécurité du réseau UAANET correspond-il à celui des protocoles de routage sécurisé existants ?
2. Les mécanismes de sécurité proposés pour les réseaux MANETs sont-ils appropriés pour les réseaux UAANET ?
3. Quels sont les défis spécifiques au réseau UAANET qui ne sont pas examinés par les mécanismes de sécurité ad hoc existants ?

À partir de la comparaison des approches existantes, nous pouvons affirmer que le réseau UAANET partage le même objectif de sécurité que le réseau MANET. Il s'agit en effet d'assurer l'authentification des messages de routage, l'intégrité des champs mutables et l'authentification des nœuds qui, pour sa part, est traitée par la proposition d'une infrastructure adaptative de gestion de clé. Nous avons décidé d'écarter la question de la confidentialité des messages de routage, car les informations de routage ne contiennent pas d'informations secrètes susceptibles de mettre à mal le réseau. En effet, seule l'authentification des messages est vraiment significative pour détecter la modification non autorisée des paquets de routage ou encore la création de faux paquets. Ce constat nous permet également d'ignorer les services de sécurité « secrets et anonymats » qui peuvent être trouvés dans les réseaux VANET [EBPQ14].

Par ailleurs, pour tenir compte de la forte mobilité des drones et du changement fréquent de topologie, nous ne considérerons pas les algorithmes ayant un problème de mise à l'échelle, comme les protocoles SOLSR et SRP. Ces protocoles génèrent une quantité importante de paquets d'information, que chaque nœud doit échanger ou maintenir. Ce genre d'état

n'est pas approprié pour un réseau à topologie dynamique comme le réseau UAANET, car le délai et la quantité importante d'*overhead* peut dégrader la performance du réseau.

En outre, il est important de préciser que, dans un réseau UAANET, la métrique de mise à l'échelle n'a pas d'importance significative, en raison, principalement, de la faible densité des nœuds du réseau UAANET (cf. chapitre 1). Cela favorise ainsi l'utilisation de la technique cryptographique symétrique. Néanmoins, ce type d'approche (le partage des clés) peut devenir rapidement complexe dès que le nombre de nœuds devient trop élevé. Ce problème peut être amoindri en utilisant une architecture d'authentification par diffusion telle que TESLA. Toutefois, le problème peut persister en raison de la caractéristique temps réel des trafics applicatifs qui peut ne pas toujours être respectée en appliquant l'algorithme TESLA. En effet, avec TESLA, l'authentification nécessite un minium de délai d'aller-retour pour réaliser la procédure de divulgation de la clé. Une raison supplémentaire de ne pas utiliser la cryptographie symétrique est le manque de coopération des nœuds UAANET avec une entité centrale pour identifier et écarter les nœuds malveillants. Pour atteindre ces objectifs, un service de non-répudiation, qui n'est disponible qu'avec la cryptographie asymétrique, est nécessaire. Néanmoins, il est important de souligner que, pour des raisons de fiabilité, il est possible de combiner la cryptographie asymétrique et la cryptographie symétrique. Nous concluons donc que l'approche de cryptographie symétrique n'est pas adaptée au réseau UAANET. Ainsi, nous pouvons ne pas considérer les protocoles SRP, SEAR et SAR.

L'inconvénient majeur de l'approche asymétrique est la forte consommation en capacité de traitement et de stockage mémoire, ce qui peut être un handicap pour les drones à faible capacité de mémoire et d'énergie [Sal13].

Toutefois, dans notre cas spécifique, nous utilisons des drones qui ont des capacités de mémoire et de calcul bien supérieures aux nœuds mobiles dans les réseaux MANET. Par conséquent, les problèmes liés à la consommation de ressources des cryptographies asymétriques dans le réseau MANET peuvent ne pas être considérés pour l'étude de sécurité associée aux réseaux UAANET.

Par ailleurs, nous choisissons d'écarter de notre analyse les solutions qui exigent la présence d'un serveur de certificats, comme le protocole ARAN. Ce genre de système n'existe pas encore dans les architectures disponibles [GJV15]. En outre, ces solutions peuvent engendrer des vulnérabilités supplémentaires en cas d'attaque sur le système centralisé.

En ce qui concerne le protocole SPREAD, le fait qu'il envoie une partie du trafic sur plusieurs liens nécessite l'existence de plusieurs chemins et donc de plusieurs nœuds. Dans un réseau UAANET, il est souvent commun de déployer trois ou quatre drones. Il est ainsi peu fréquent d'avoir plusieurs chemins allant d'une source vers une destination.

Suite à ces constats, notre choix s'est donc tourné vers le protocole SAODV. Il garantit l'authentification et l'intégrité des messages. Il permet également de sécuriser à la fois les

champs mutables et les champs non mutables. Les champs mutables sont protégés par une fonction de hachage pour alléger l'effort de calcul demandé à chaque saut. L'utilisation de la cryptographie asymétrique est justifiée par notre système UAS qui possède des puissances énergétiques suffisantes. Ce choix est synthétisé dans le tableau 2.2.

Toutefois, il est important de souligner que le protocole SAODV est vulnérable aux différentes variantes de l'attaque *wormhole*, notamment lorsqu'un ou plusieurs attaquants transfèrent des paquets de routage sans modifier le nombre de sauts. Ces attaques ne peuvent pas être détectées par SAODV. Elles peuvent être aussi exécutées sans nécessairement besoin de posséder les clés cryptographiques utilisées par les nœuds légitimes. Nous nous sommes penchés sur cette problématique et détaillons dans le chapitre 4 une proposition pour contrer cette attaque.

Par ailleurs, une autre problématique nous intéresse dans cette thèse : il s'agit de la contribution à la certification de notre architecture de communication sécurisée. Ce besoin s'explique surtout par la nécessité d'introduire notre système UAS composé d'une architecture de communication ad hoc dans l'espace aérien national. Il s'agit aussi d'un besoin exprimé par l'entreprise Delair-Tech pour valider le système UAS composé de l'architecture de communication ad hoc. En ce qui concerne les standards existants, aucune norme n'a été à ce jour proposée pour réguler le déploiement d'une flotte de drones. Toutefois, nous pouvons tirer profit des standards dans le secteur aéronautique comme le DO-178 *Software Considerations in Airborne Systems and Equipment Certification* [BH07]. Dans ce secteur, les logiciels embarqués à bord d'un avion sont soumis à des contraintes de certification qui consistent à s'assurer qu'un processus de développement conforme au secteur aéronautique a été suivi ; par exemple, par un enchainement de taches bien spécifiques (selon le cycle de conception en V). Dans cette thèse, nous nous intéressons donc à la mise pratique d'une méthodologie de prototypage rapide pour le développement logiciel de notre protocole de routage sécurisé. Pour cela, nous considérerons une approche de développement orientée modèle que nous détaillerons dans le chapitre suivant.

Solutions techniques	**Algorithmes**	**Services de sécurité**	**Attaques traitées**
Signature numérique (SAODV)	RSA	Authentification	- Dégradation de performance - Modification de la topologie du réseau
Fonction de hachage (SAODV)	SHA [GH04]	Intégrité	

Tableau 2.2 : Ensemble des mécanismes retenus

2.7 Conclusions

Dans ce chapitre, nous avons étudié la sécurité au sein d'un réseau ad hoc de drones. Nous savons que des attaques dites intelligentes existent et peuvent être exécutées contre un réseau UAANET. Nous avons vu que les solutions de sécurité ne sont pas encore au point alors que le déploiement des réseaux UAANET va sans doute s'intensifier dans les années à venir en raison de la prolifération des mini-drones. Ainsi, il est important de définir et concevoir des mécanismes de sécurité adaptés au contexte de déploiement d'une flotte de drones. Cela nécessite le chiffrement de la communication et la vérification de l'identité des messages et des nœuds. Ces mécanismes doivent aussi prendre en considération les ressources limitées en bande passante et en capacité de traitement des mini-drones.

La classification des attaques que nous avons présentées met en évidence le fait que la seule protection du protocole de routage ne peut apporter de solution à la vulnérabilité des réseaux ad hoc en général. Cette protection est efficace contre les attaques au niveau de la couche réseau pour permettre le bon acheminement des paquets de données. Nous avons vu que les attaques aux couches inférieures dépendent des mécanismes d'accès au médium pour la couche liaison et des techniques de transmission pour la couche physique. Quant aux attaques au niveau de la couche application, elles dépassent largement le domaine des réseaux ad hoc. Nous concentrerons donc notre étude sur la protection de la couche réseau pour fiabiliser la recherche et l'utilisation des routes pour l'échange des flux applicatifs.

Parmi les protocoles existants ayant été étudiés, notre attention s'est portée particulièrement sur le protocole SAODV qui permet d'assurer l'authentification des messages. Toutefois, ce protocole reste vulnérable à des attaquants *wormhole* qui peuvent interférer avec l'échange des données de charge utile, ou affecter sa performance en sélectionnant les paquets à retransmettre. Il nous semble donc important d'étendre la propriété du protocole SAODV à l'aide d'un mécanisme qui puisse prévenir et détecter cette attaque. Dans le chapitre 4, nous présenterons une approche de détection de cette attaque. Enfin, nous nous intéresserons dans le chapitre 3, à la méthodologie employée pour le développement d'un logiciel embarqué dans un drone civil. Cette méthodologie est utilisée pour contribuer à la certification de notre système multi-drones et va permettre de tester sa mise en œuvre.

Chapitre 3

Méthodologie de développement d'un logiciel embarqué pour un système de drones

Les différents cas d'application proposés par les systèmes UAS sont prometteurs, car ils permettent de réaliser des missions flexibles et évolutives. Ces missions sont pour la plupart, réalisées sur des zones peuplées en occupant l'espace aérien national. Il est donc primordial d'assurer la bonne conduite de la mission en couvrant tous les cas possibles d'exécution des différents modules du système. Ceci dans le but d'éviter des problèmes de dysfonctionnement imprévu qui pourraient avoir des conséquences lourdes (par exemple des pertes humaines). Dans le cas spécifique des aéronefs sans pilote, aucune norme de sûreté de fonctionnement n'a été à ce jour définie. Si elle existait, elle aurait permis de définir le cycle de vie et le niveau de sécurité des systèmes embarqués composant le système UAS. Le domaine de l'aviation civile qui est relativement proche du domaine des systèmes de drones possède un standard de sécurité (sécurité de conception) pour les logiciels, le document DO-178 avec ses variantes. Dans ce sens, pour valider son système de drones, il peut être judicieux de suivre la recommandation de cette norme. Dans cette thèse, nous nous concentrons sur ce type de problème en nous penchant sur l'ingénierie de développement des systèmes complexes et en proposant une méthodologie de prototypage rapide pour le développement de notre protocole de routage sécurisé.

Contents

3.1 Introduction 74
3.2 Conception de systèmes embarqués critiques 76
3.2.1 Norme de sûreté 76
3.2.2 Cycle de vie de développement logiciel 77
3.2.3 Processus de validation de logiciel dans la norme DO-178B . . . 79
3.3 Méthodologie orientée modèle 80
3.3.1 Présentation des différentes étapes 81
3.3.2 Analyse des modèles 83
3.3.3 Avantages de l'utilisation de la méthodologie MDD (Model Driven Development) 85
3.4 Certification d'un système UAS 86
3.4.1 Apport de l'approche MDD dans la certification UAS 88
3.5 Choix d'outils pour l'application de la méthodologie MDD . . . 90
3.5.1 Synthèse sur les chaînes d'outils 94
3.6 Conclusions 95

3.1 Introduction

Un système de drones est généralement composé de plusieurs modules logiciels responsables chacun d'un ensemble de fonctions spécifiques. Ces modules sont amenés à échanger dynamiquement des informations relatives à leur environnement afin d'offrir différents types de services lors d'une mission donnée. Pour tenir compte de cet environnement critique, le développement d'un logiciel s'exécutant dans le système UAS nécessite la garantie d'une sécurité fonctionnelle en fournissant un minimum d'écart entre la spécification du cahier des charges et l'implémentation du code source. Il est important de noter que dans le cas des systèmes embarqués critiques, nous parlons de la sécurité fonctionnelle, c'est-à-dire la sécurité dépendante du bon fonctionnement du système en réponse à ses entrées. Elle est associée au terme sûreté (safety) de fonctionnement et non à la sécurité des informations. Nous utilisons ce terme puisque il est souvent employé dans la littérature pour les études de conception des logiciels critiques.

Par ailleurs, la plupart des méthodes de conception de logiciels sont basées sur le langage UML [RJB04]. Toutefois, ces méthodes nécessitent une adaptation particulière pour prendre en considération l'environnement d'exécution du système final. C'est-à-dire qu'ils (les méthodes basées sur UML) ne permettent que des descriptions haut niveau d'un système sans tenir compte des contraintes d'implémentation et d'exécution matérielle. Également, le langage UML ne répond pas aux exigences de conception lorsqu'il est utilisé

dans le contexte aéronautique ou dans le contexte de système de drones. Ceci est causé par le fait qu'il ne possède pas les chaines d'outils nécessaires pour contribuer à la validation d'un système critique. Dans le contexte de conception d'un logiciel embarqué s'exécutant sur un aéronef piloté ou non piloté, il est nécessaire de prendre en compte la certification du logiciel durant sa phase de conception. Cette considération se traduit par l'utilisation de chaines d'outils de conception qui vont contribuer à la certification du système final.

En outre, les drones civils sont actuellement déployés sur des zones aériennes spécifiques et séparées de l'espace occupé par les avions (de tourisme, moyens ou longs courriers). Cependant, afin d'être utilisés à des fins commerciales, les drones civils peuvent être amenés à partager le même espace aérien que les avions civils. Pour atteindre cet objectif, il est nécessaire de valider le fonctionnement du système autopiloté. Cette étude consiste à parcourir toutes les fonctions constituant le système UAS et d'étudier leurs exécutions dans tous les scénarios possibles. Pour contribuer à cette validation, nous avons décidé d'utiliser la solution qui a été adoptée dans le domaine de l'aéronautique ; cette approche consiste à utiliser une méthodologie de prototypage rapide afin de procéder à des vérifications de la sécurité fonctionnelle du logiciel pendant sa conception. Cette méthode se base sur une architecture orientée modèle et l'utilisation d'une chaîne d'outils de vérification formelle et de génération de code. Elle garantit une traçabilité complète des exigences à partir de la spécification jusqu'à leur implémentation dans le code source. Cette méthode permet la portabilité et la réutilisabilité des codes sources générés à travers la séparation des aspects dépendant de la plateforme et les aspects abstraits décrivant le système.

Notre objectif dans cette thèse est de mettre en œuvre une architecture de communication sécurisée au sein d'une flotte de drones. Cette architecture va s'ajouter avec l'architecture logicielle du système UAS de Delair-Tech déjà en place. Bien que l'architecture actuelle permette déjà de déployer un unique drone dans l'espace aérien civil français [MA15], le nouveau système composé d'une flotte de drones doit passer par un nouvel examen de validation. En effet, le déploiement d'une flotte de drones introduit de nouveaux comportements (surtout si les trafics de commande et de contrôle sont échangés à travers la flotte), qui n'existent pas dans les sous-systèmes actuels, ce qui nécessite une nouvelle validation globale.

Nous nous intéressons donc dans cette thèse à la contribution de validation de notre protocole de routage sécurisé par l'utilisation d'une méthodologie orientée modèle. L'utilisation des modèles permet de décrire sur plusieurs niveaux d'abstraction et de manière concise et claire, la complexité des logiciels du système de drones. Les modèles permettent également d'avoir un système modulaire, facile à comprendre et réutilisable. Ces modèles sont ensuite transformés en code source par des outils de génération de code préconisés dans le domaine de l'aéronautique. Les modèles aussi bien que les codes sont vérifiés par des outils de vérification formelle pour attester la correspondance entre la spécification (par les modèles) et l'implémentation.

Généralement, la conception et la validation des systèmes embarqués critiques dans le contexte aéronautique sont définies par la norme DO-178B [MLD+13]. Cette norme à été complété en 2012 par le DO-178C qui introduit la programmation orientée objet et l'utilisation des méthodes de modélisation et des méthodes formelles. Un sous-ensemble des recommandations de cette norme (l'utilisation des modèles et des méthodes formelles) a donc été appliqué pour assurer la sécurité fonctionnelle de notre système final.

Dans ce chapitre, nous verrons d'abord les motivations qui ont poussé à l'utilisation de la démarche orientée modèle. Puis, nous aborderons les principes de cette méthode en présentant les différentes étapes et les outils pouvant être utilisés. Enfin, nous étudierons l'avantage de cette méthodologie dans l'effort de conception et dans le contexte de certification. Pour finir, nous examinerons les chaines d'outils qui ont été utilisés pour concevoir notre logiciel embarqué.

3.2 Conception de systèmes embarqués critiques

L'utilisation des approches de vérification et de validation est justifiée par les exigences de conception des systèmes embarqués critiques. En effet, des questions se posent lorsqu'il s'agit de faire cohabiter les contraintes de l'exigence de spécification et les contraintes sécuritaires (au niveau fonctionnel). L'utilisation d'outils automatiques de vérification peut être une solution à ce problème. Dans ce qui suit, l'influence des normes sur la conception des logiciels embarqués sera étudiée.

3.2.1 Norme de sûreté

Les normes de sûreté qui orientent la conception des systèmes embarqués critiques se basent sur un cycle de vie et un niveau de sûreté [1]. Le cycle de vie détermine les étapes de vie du logiciel embarqué, en incluant les exigences de sécurité fonctionnelle de la conception. Le niveau de sûreté quant à lui permet d'assigner chaque sous-système à son exigence en matière de niveau de criticité.

Nous présentons dans cette section, la norme DO-178B. Cette norme préconise certaines méthodes pour la conception et la validation des systèmes embarqués critiques pour le domaine avionique. Elle définit un ensemble de règles, techniques et méthodes à appliquer pour un niveau de sûreté ciblée. À titre d'exemple, la FAA (Federal Aviation Administration) a défini 5 niveaux de panne sur lesquelles la norme DO-178B se base. Les niveaux de criticité sont illustrés dans le tableau 3.1. Le niveau A, qui correspond à la catégorie des pannes catastrophiques susceptibles d'empêcher la poursuite d'un vol, est le plus critique

1. La sûreté concerne l'ensemble des moyens humains, organisationnels et techniques réunis pour faire face aux actes spontanés ou réfléchis ayant pour but de nuire, ou de porter atteinte dans un but de profit psychique ou financier.

alors que le niveau E représente les parties non sécuritaires du système. Il est à noter que c'est l'autorité d'évaluation de risque qui détermine le niveau correspondant au système. Il y a donc un ensemble de règles de conception à suivre sur chaque niveau qui va permettre de certifier un logiciel embarqué. En utilisant cette norme dans le domaine des systèmes UAS, l'application des recommandations de sécurité fonctionnelle va permettre d'accroître la confiance des autorités de certification au système final.

Catégorie de panne	**Condition**
Niveau A	Catastrophique : pannes susceptibles d'empêcher la poursuite en toute sécurité d'un vol ou d'un atterrissage.
Niveau B	Dangereuse : pannes susceptibles de réduire les possibilités de l'aéronef.
Niveau C	Majeure : pannes susceptibles de réduire les possibilités de l'aéronef se traduisant par une réduction significative des marges de sécurité ou des capacités fonctionnelles
Niveau D	Mineure : pannes susceptibles d'entraîner pour l'équipage des actions se situant tout à fait dans le domaine de leurs capacités ; par exemple, une légère réduction des marges de sécurité ou des capacités fonctionnelles.
Niveau E	Sans effet : pannes n'affectant pas la capacité opérationnelle de l'aéronef.

Tableau 3.1 : Classification des pannes dans la norme DO-178B

3.2.2 Cycle de vie de développement logiciel

Le développement d'un logiciel embarqué critique doit suivre un ensemble de processus au cours de son cycle de vie. Chaque processus comporte un ou plusieurs critères de transition permettant de passer au processus suivant. Généralement, il y a plusieurs cycles de développement de logiciels qui existent [Pon08], mais il est souvent préconisé d'utiliser celui qui est recommandé par la norme. Dans notre contexte d'étude, aucune norme n'a été à ce jour définie pour régir la conception d'un logiciel embarqué au sein d'un système UAS. Néanmoins, le domaine aéronautique qui est le plus proche de notre domaine d'étude en matière de sécurité fonctionnelle [BTJBL12], se base sur la norme DO-178B qui utilise le modèle en V. Nous avons donc décidé de suivre les recommandations de cette norme en appliquant le modèle en V comme la montre la figure 3.1.

Le processus du cycle de vie en V d'un logiciel suivant la norme DO-178B est généralement composé des étapes suivantes :

— La planification du logiciel qui a pour objectif de définir et de coordonner les activités de développement et les processus du projet ;

— Le developpement du logiciel qui régroupe les phases des spécifications, de conception, du codage et de l'intégration ;

— La vérification du logiciel qui a pour objectif de détecter les erreurs d'exigences qui peuvent avoir été introduites au cours de la définition des spécifications.

— La gestion de configuration qui a pour objectif d'identifier la configuration du logiciel et de produire des références et traçabilités. Elle fournit également des comptes rendus des anomalies ;

— L'assurance qualité qui a pour but de garantir que le cycle de vie produit un logiciel conforme à ses spécifications. Dans cette étape, il y a également la revue de conformité du produit final.

— La coordination pour la certification a pour objectif de fournir les moyens de conformités du logiciel et une preuve de la planification de sa conception. Les données suivantes sont préparées et soumises à l'autorité de certification : le plan des aspects logiciels de la certification, le dossier de la configuration du logiciel et le résumé des travaux réalisés.

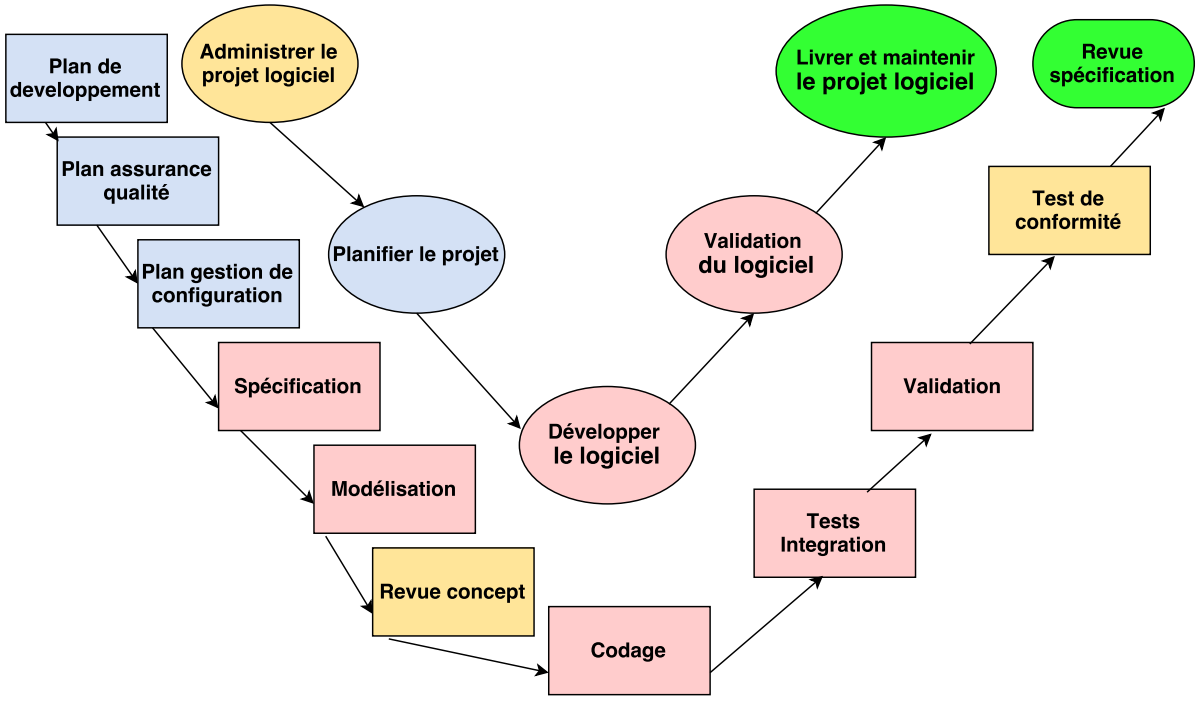

FIGURE 3.1 : Cycle de développement en V

Il est important de noter que pendant la phase de développement du logiciel, les spécifications du logiciel utilisent les produits du cycle de vie du système pour développer les exigences de haut niveau du logiciel. Cela comprend les exigences fonctionnelles, de performance, d'interface et de sécurité fonctionnelle. Ensuite, la modélisation de ses spécifications est faite par plusieurs itérations. L'ajout de code dit de bas niveau caractérisant l'interaction avec le système d'exploitation est traditionnellement réalisé à cette étape [Bac05]. Par la suite, le codage du logiciel est réalisé par la combinaison du code généré et l'ajout des codes de collage. Cette association permet ensuite de passer à l'étape d'intégration du

logiciel pour charger le code exécutable dans la machine cible.

Par ailleurs, la norme DO-178 suggère des techniques à utiliser qui sont en accord avec le niveau de sûreté visé. Par exemple, il est recommandé d'utiliser soit un compilateur certifié (par exemple, le compilateur Ada [2]), soit un compilateur dont la confiance résulte de l'utilisation (par exemple, le compilateur gcc [3]). L'utilisation de ces compilateurs contribue à la validation du code source à cause de leur spécificité technique dans n'importe quelle catégorie de certification (A, B,C, D ou E).

Il est aussi possible que les autorités de certification préconisent l'utilisation des langages fortement typés ou de règles de codage. Cela est dû au fait que les contraintes associées au développement d'un logiciel embarqué critique engendrent des catégories de programmes spécifiques. Par exemple, il est fréquent de recommander la suppression de l'instruction *goto* dans les logiciels critiques, car jugés trop difficiles à étudier et à vérifier [Dij68]. Ces règles sont des exemples parmi tant d'autres et ils peuvent s'appliquer à n'importe quel niveau de certification.

3.2.3 Processus de validation de logiciel dans la norme DO-178B

L'utilisation des jeux de test est la méthode de validation recommandée dans la norme DO-178B. Ils ont pour objectif de démontrer la capacité du logiciel à répondre à des données d'entrée et des conditions normales. Ces jeux de test sont constitués des opérations suivantes :

— Pour les fonctions liées au temps, telles que filtres, intégrateurs et retard, elles sont réalisées par des itérations multiples afin de vérifier les caractéristiques de la fonction dans son contexte.

— Pour les transitions d'états, des jeux de test sont effectués afin de mettre en œuvre les transitions possibles en fonctionnement normal.

— Pour les spécifications de logiciel exprimées par des équations logiques, les jeux de test doivent vérifier l'utilisation des variables et les opérateurs booléens.

La norme préconise également les jeux de test de robustesse qui ont pour objectif de démontrer la capacité du logiciel à répondre à des données d'entrée et des conditions anormales. Il est important de préciser que les conditions anormales évoquées ici ne sont pas les événements attendus, mais les arrivées imprévues durant l'exécution du logiciel.

Par ailleurs, les jeux de test et les jeux de test de robustesse peuvent ne pas être suffisants car ils n'assurent pas à 100 % l'absence d'erreurs [BBF+00]. C'est pour cela que la méthode formelle est introduite. C'est une méthode de vérification basée sur des équations mathématiques. Alors que les jeux de test vérifient si le système garde son comportement correct

2. http ://adacore.com
3. http ://gcc.gnu.org

par rapport à une situation particulière, la méthode formelle vérifie l'adéquation du système à sa spécification pour n'importe quelle situation connue ou inconnue. L'application des méthodes formelles est alors particulièrement importante pour valider la propriété d'un logiciel aéronautique et d'un système UAS. Des résultats satisfaisants en rapport avec la vérification de programme par des techniques de vérification de modèle (model checking) [BBF+13] ou encore l'analyse statique par interprétation abstraite [BCC+03] montrent l'intérêt de l'utilisation des méthodes formelles dans un cycle de développement. Dans notre cas d'étude, nous avons utilisé des outils automatiques de vérification formelle qui sont intégrés dans le toolkit de Mathworks. Parmi ces outils, il y a l'outil Simulink Design Verifier qui utilise des méthodes formelles pour identifier les erreurs de conceptions dans les modèles. Il vérifie les blocs dans le modèle qui génèrent des erreurs telles que le dépassement d'entier, la logique morte, les violations d'accès aux tableaux et la division par zéro. Pour chaque erreur rencontrée, il produit un cas de test de simulation permettant la correction.

3.3 Méthodologie orientée modèle

L'ingénierie logicielle s'oriente actuellement vers l'ingénierie des modèles. Cette approche de développement se concentre sur la création et l'exploitation de modèles abstraits. Ce type d'approche a été introduite et définit par l'OMG (Object Management Group) qui a voulu faire évoluer l'approche orientée objet en augmentant le niveau d'abstraction en un niveau où une autre représentation des concepts et des relations issue de la spécification initiale est utilisée (c'est-à-dire le modèle). Un modèle est une représentation abstraite des connaissances et des activités qui régissent un domaine applicatif facilitant ainsi la compréhension du système final. Cette technique de développement permet au concepteur de se concentrer sur le comportement souhaité du système, et non sur son implémentation. La génération partielle du code à partir des spécifications des modèles permet entre autres de réduire le coût de développement.

Par ailleurs, l'utilisation d'outils de conception tels que Matlab/Simulink [SN93] ou encore Lustre/SCADE [LP09] permet de comprendre la complexité d'un système en proposant un niveau d'abstraction haut niveau. Elle amène de nouveaux panoramas sur l'application de méthode formelle dans le cycle de développement des logiciels embarqués. En effet, ces outils rendent possible la modélisation de l'environnement d'exécution du logiciel. Le concept général est de dériver le logiciel embarqué critique directement à partir des modèles afin de limiter les erreurs de conception et de gagner en temps de développement. Bien que le langage UML soit souvent utilisé pour la modélisation de logiciels, Matlab/Simulink possède un avantage clé, car il permet d'apporter une vue globale du système logiciel et de son environnement avec suffisamment d'information pour valider ses comportements vis-à-vis de sa spécification. Cet outil est souvent utilisé par les industriels qui conçoivent

et testent des logiciels avioniques [Rie13]. Des retours d'expériences ont montré l'utilité de cet outil dans le processus de certification [ESD13].

Dans ce qui suit, nous allons lister les différentes étapes de cette méthodologie se basant sur la recommandation de la norme DO-178B.

3.3.1 Présentation des différentes étapes

La méthodologie que nous proposons est basée sur une adaptation du cycle de développement en V. Nous utilisons simultanément un transformateur de modèle en code pour accélérer la phase de codage et un ensemble d'outils de vérification formelle pour vérifier les propriétés des modèles et du code. Afin de présenter les différentes étapes, nous supposons que la phase de spécification du protocole a été effectuée au préalable de l'application de la méthodologie. Ainsi, nous faisons l'hypothèse que nous disposons d'un ensemble d'exigences claires et valides définissant l'ensemble des services que le protocole de routage sécurisé doit fournir, ainsi que les dépendances qu'il doit considérer. Dans le cas de notre protocole de routage sécurisé, cet ensemble d'exigences comprend notamment des besoins concernant le routage des trafics échangés, leur sécurisation et la performance du système final.

La méthodologie peut être résumée par l'application successive des sept étapes comme le montre la figure 3.2.

La première étape de la méthodologie est le partitionnement qui consiste à diviser le futur logiciel en différentes classes de partitions. Chaque partition est associée à un sous-ensemble des fonctionnalités du système (et donc à un sous-ensemble des exigences fonctionnelles). Cette étape est faite manuellement et est très significative pour la suite de la conception. La finalité de cette étape est de pouvoir diviser le cahier des charges en un groupe de sous-ensembles disjoints composés de procédures. Le logiciel final exécutera un ensemble d'instances de ces procédures.

Dans la seconde étape, nous passons à l'étape de modélisation qui consiste à décrire sous forme de bloc Simulink et des machines à états avec Stateflow, le comportement de chaque partition logicielle. Ces modèles sont faciles à lire et permettent de tester et valider le fonctionnement du logiciel dès la phase de conception. Pour notre cas, nous avons modélisé notre protocole de routage avec les outils Simulink et Stateflow de la version R2014B [Gui14].

La troisième étape de la méthodologie consiste à transformer chaque modèle de classe de partition en code source à l'aide d'un générateur automatique de code. C'est l'étape de transformation. Nous pouvons générer du code en différents langages tant que le générateur le supporte. Dans notre cas, nous avons généré du code C à l'aide de l'outil Embedded Coder [RBB09]. C'est un outil propriétaire de Mathworks.

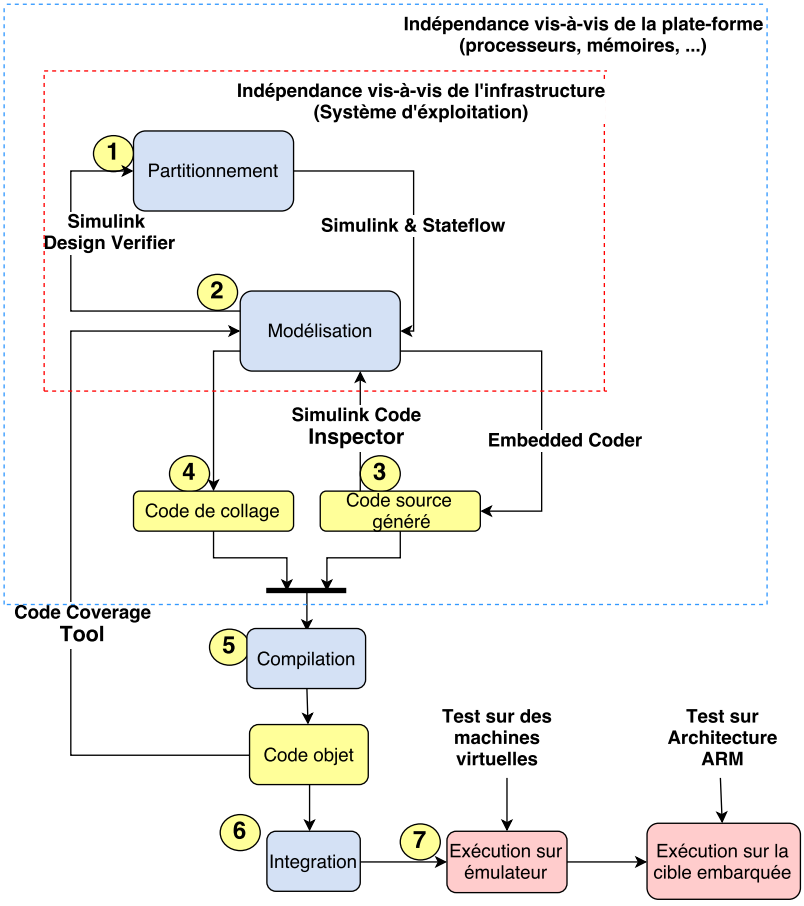

FIGURE 3.2 : Méthodologie de développement orienté modèle pour les systèmes embarqués complexes

Par la suite, durant la quatrième étape, pour que le code généré précédemment puisse fonctionner au sein du système final, il doit être complété par des codes écrits à la main, que l'on appelle « code de collage ». Il est à noter que ce code de collage peut également être automatisé dans le cas d'une importante production [VL12]. Ce code additionnel contient la spécification des liaisons entre les entrées/sorties du modèle et celles de la partition logicielle du système d'exploitation de la cible matérielle. Il spécifie également les groupes d'informations d'ordonnancement des différentes partitions.

La cinquième étape est l'étape de compilation qui consiste à compiler ensemble les codes générés et de collage pour avoir un code binaire de toutes les classes de partition. Ensuite, dans la sixième étape, la structure globale de l'algorithme est décrite en y précisant pour chaque partition le nombre d'instances s'exécutant en parallèle, leur configuration et leur ordonnancement. Ces paramètres sont combinés avec le noyau du système d'exploitation et l'ensemble des codes binaires de partition pour produire une image binaire finale. C'est l'étape d'intégration (la sixième étape).

Par la suite, la septième étape consiste à tester l'image binaire finale dans une ou plusieurs architectures cibles pour vérifier le fonctionnement correct et voulu du système. Dans notre cas, l'image binaire finale a été testée sur un émulateur et sur une cible embarquée utilisée par le drone DT18. L'objectif final est d'utiliser l'image binaire finale pour réaliser des tests en vol avec les drones. Les détails de ces cibles seront présentés dans les chapitres 4 et 5. En particulier, l'émulateur a été développé pour faciliter la migration du code développé sur la machine locale vers l'architecture logicielle et matérielle utilisée par les drones de Delair-Tech.

Les sept étapes sont résumées ci-dessous :

1. Partitionnement (fait manuellement)
2. Modélisation
3. Transformation (des modèles aux codes source de cœur)
4. Collage (du partitionnement aux codes source de collage)
5. Compilation (des codes sources de cœur et de collage aux codes binaires des partitions)
6. Intégration (des codes binaires des partitions à une image binaire)
7. Exécution (de l'image binaire sur émulateur ou cible réelle)

3.3.2 Analyse des modèles

Il est fondamental de vérifier la conception du modèle, notamment pour assurer la qualité de conception des différents blocs Simulink et des diagrammes Stateflow où de potentielles erreurs (par exemple, une division par zéro) peuvent exister. Une bonne méthode de modélisation doit fournir un outil de modélisation avec des critères et des lignes directrices pour développer des modèles de qualité. Ces guides peuvent être exprimés sous forme de règles de conception ayant été validées ou des exemples de modèle.

Plusieurs travaux se sont penchés sur le développement d'une technique d'analyse statique pour vérifier le comportement d'un modèle. Des exemples des travaux relatifs à ce type d'approche sont détaillés dans [LP99]. L'approche le plus utilisée est le *Model checking* qui consiste à lister l'ensemble des états par lequel peut passer le programme, puis à parcourir cet ensemble pour valider que la propriété a été vérifiée. Nous pouvons également recourir à des outils systématiques de test de modèle. Ce genre d'outil implique l'exécution des programmes en prenant en entrée des variables pour satisfaire le critère de test. L'idée étant d'évaluer le comportement du modèle. Dans notre cas d'étude, nous nous sommes basés sur le *Model checking* en utilisant des outils de vérification formelle dans Matlab/Simulink. Ces outils se basent sur des approches de preuves formelles pour tester le modèle avant de passer la main à Embedded Coder pour la génération de code en cas de succès ou la

génération de rapports d'erreurs en cas d'échec. Ces étapes sont illustrées par les figures 3.3 et 3.4.

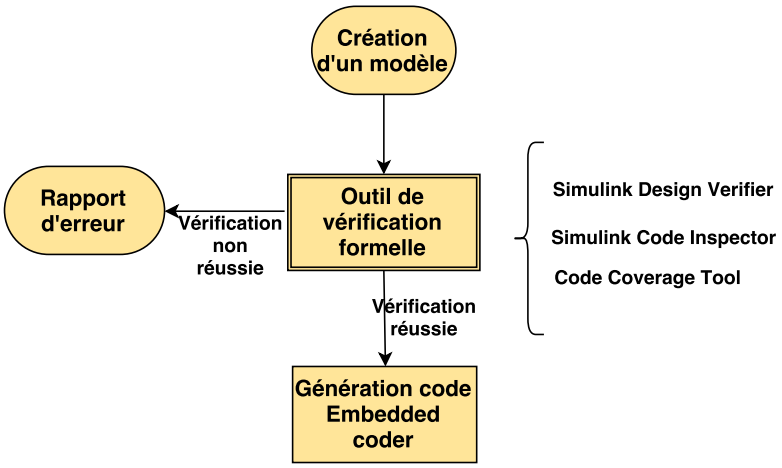

FIGURE 3.3 : Illustration de l'enchainement de la vérification du modèle jusqu'à la génération de code

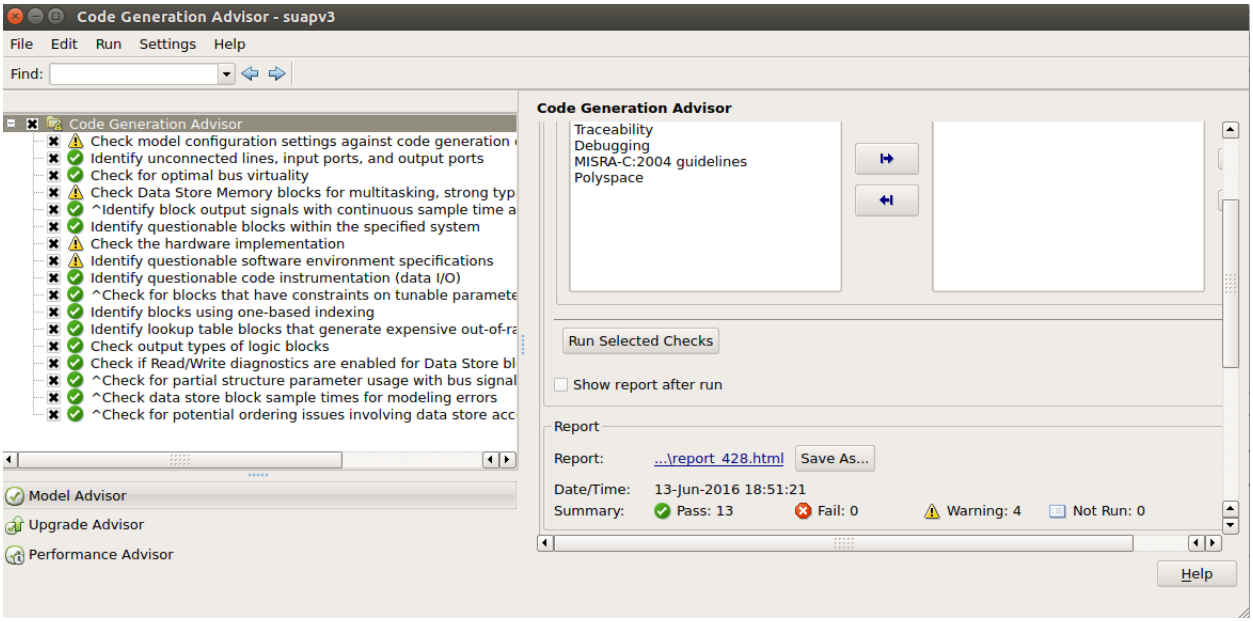

FIGURE 3.4 : Illustration des rapports de vérification de notre modèle

Nous pouvons voir à la figure 3.4 que notre modèle passe avec succès les vérifications réalisées par les outils de vérification formelle. Nous pouvons constater que le rapport contient quatre avertissements qui sont liés à la portée des variables. Ces avertissements sont ensuite corrigés manuellement dans le code généré. La figure 3.4 illustre une copie d'écran d'un exemple de vérification d'un modèle (vérification effectuée par l'outil Simulink

Design Verifier) pour suggérer des améliorations.

3.3.3 Avantages de l'utilisation de la méthodologie MDD (Model Driven Development)

Cette méthodologie rend service au concepteur logiciel de différentes manières.

— La méthodologie permet d'accélérer la phase de développement, car les modèles sont utilisés comme les principaux artefacts de l'étape de développement. Ce modèle est ensuite transformé en un code fonctionnel de haut niveau. Le fait d'utiliser des modèles permet de ne pas travailler directement sur les différentes lignes de code. Il est donc possible d'avoir une représentation rapide et fonctionnelle du code, ce qui permet d'accélérer le processus de développement ;

— En appliquant le concept des modèles, il y a abstraction du logiciel par rapport aux approches traditionnelles. Chaque élément du modèle représente plusieurs lignes de code. Il est donc plus facile d'avoir une vue fonctionnelle de l'architecture logicielle. Ces modèles de haut niveau permettent aux concepteurs de vérifier et de corriger le comportement du système au plus tôt, minimisant ainsi les risques d'incompatibilité par rapport au produit final attendu et limitant ainsi les coûts liés aux corrections ;

— La décomposition du logiciel embarqué en plusieurs partitions facilite la conception. Elle isole les fonctionnalités incorrectes et trop imprécises. Ce fractionnement simplifie également les phases d'évaluation et de validation, car l'assurance de sûreté est plus simple à valider pour des systèmes modulaires que pour les systèmes indissociables [AFOTH06]. Ainsi, le logiciel est à la fois modulaire et réutilisable ;

— La caractéristique importante de l'approche MDD est que le processus de développement se focalise sur les modèles plutôt que les codes sources. L'avantage notable de cette approche est l'expression en modèle utilisant des concepts non liés au problème de technologie d'implémentation et qui sont plus proches de l'exigence du cahier des charges. Cela rend les modèles simples à spécifier, à comprendre et à maintenir. Cela rend aussi les modèles moins sensibles aux modifications de la technologie bas niveau. En effet, le modèle est à la fois indépendant vis-à-vis de la plateforme matérielle, processeur et mémoire et aussi vis-à-vis de l'infrastructure logique et le système d'exploitation ;

— La génération automatique de code contribue à la productivité et à la fiabilité du système final. Le code généré est dans la plupart des cas prêt à être utilisé ne nécessitant aucune modification. Le processus de vérification est fait par des outils de vérification formelle qui s'assurent de l'absence des propriétés indésirables. Cette vérification est réalisée durant la phase de conception des modèles de manière itérative pour corriger au plus tôt des erreurs de conception.

3.4 Certification d'un système UAS

Dans cette partie, nous allons analyser les besoins et l'état de l'art de la certification d'un système UAS, pour montrer le lien avec le choix de la méthodologie orientée modèle. Généralement, le besoin de certification est expliqué par la nécessité d'intégrer les drones civils dans l'espace aérien national. Certifier un système, c'est lui attribuer un certain degré de confiance. Pour la partie logicielle, il s'agit de prouver à un organisme de certification le respect d'un processus de développement rigoureux. Cet objectif nécessite la spécification des règles et des procédures pour la conception physique et logique des drones civils. Ces restrictions dictent notamment le niveau de sûreté qui doit être respecté par les constructeurs des drones. À cela s'ajoute le besoin de fiabilité et de tolérance aux fautes que doit garantir le système. Pour répondre à ces exigences, ce qui est généralement fait par les acteurs dans ce domaine est de faire l'équivalence de ce qui se fait en aviation civile dans le domaine des drones civils.

Généralement, afin qu'un aéronef puisse légalement voler, il doit être au préalable validé et certifié par des autorités de certification comme le FAA (Federal Aviation Admnistration) ou encore le EASA (European Aviation Safety Agency). La certification de navigabilité couvre un certain nombre d'aspects concernant la conception et l'opération de l'aéronef. Ces aspects à vérifier concernent notamment :

— le vol : à savoir sa performance, la caractéristique de vol, la stabilité, le contrôle et la maniabilité ;

— La structure : à savoir, la charge utile, la surface de contrôle, la surface de balancement et de stabilisation l'évolution d'usure ;

— Conception : les ailes, les surfaces de contrôle, le système de contrôle ;

— Source d'énergie : à savoir le système d'alimentation ;

— Équipement : à savoir son installation, le système électrique, l'équipement de sûreté, etc.

Pour que l'aéronef soit certifié, il doit être conforme à son type de certificat c'est-à-dire que les éléments cités ci-dessus sont sous contrôle total du système pour assurer la sûreté de l'opération. Ces règles ont été appliquées à l'aviation civile pour réguler la navigabilité des avions commerciaux dans l'espace aérien. Ils peuvent être appliqués au domaine des drones civils selon l'annexe 2 de l'*International Civil Aviation Organization* (ICAO) [Mar09] qui stipule que peu importe la nature de la navigabilité d'un aéronef, des standards d'opération doivent être respectés.

Particulièrement en France, l'utilisation des drones est encadrée par l'arrêté du 11 avril 2012 de la Direction Générale de l'Aviation Civile (DGAC). La mise en place de ce dispositif s'inspire de la réglementation de l'aéromodélisme. Cette réglementation se décline dans

Scénario	Caractéristiques	Type de mission
Scénario dit «à vue» (S1)	- Drone moins de 25 kg - Distance d'évolution maximale de 200 m du télépilote - Altitude maximale de 150 m	- Inspection d'ouvrage de dimension limitée - Vol hors zone peuplée - Vol à vue
Scénario dit «hors vue» (S2)	- Altitude maximale de 50m - Drone moins de 25kg - Distance d'évolution allant jusqu'à 1km	- Vol hors zone peuplée - Vol sans visibilité - Inspection d'ouvrage de dimension surfacique ou linéaire
Scénario dit «a vue» (S3)	- Drone moins de 4kg - Drone équipé d'un dispositif de protection des tiers au sol(parachute, airbag...) - Distance d'évolution maximale de 100 m du pilote - Altitude maximale de 150m	- Évolution en zone peuplée (avec autorisation) - Inspection d'une zone de dimension connue - Vol à vue
Scénario dit «à vue» (S4)	- Altitude maximale de 150 m - Drone moins de 2 kg - Distance d'évolution illimitée	- Inspection d'ouvrage de dimension surfacique ou linéaire - Vol hors zone peuplée - Vol sans visibilité

Tableau 3.2 : Différents scénarios d'utilisation selon la réglementation française

quatre scénarios mettant en œuvre certains types d'aéronefs dans un environnement au sol déterminé, à une altitude donnée, et à vue ou non télépiloté. Ces quatre scénarios sont montrés dans le tableau 3.2. La réglementation porte sur une analyse de risques sur plusieurs critères : la masse de l'appareil, l'éloignement possible du drone par rapport au pilote, le survol d'une zone peuplée, la dangerosité du vol par rapport aux autres usagers de l'espace aérien. Cette démarche est régie par deux arrêtés en avril 2012 :

— Arrêté du 11 avril 2012 relatif à la conception des aéronefs civils sans pilote, aux conditions de leur utilisation, et sur les capacités requises pour leur pilotage à distance

— Arrêté du 11 avril 2012 relatif à l'utilisation de l'espace aérien

Pour répondre à ces critères, un drone doit obtenir une certification associée à chaque scénario d'utilisation. Notre objectif est donc de contribuer à éliminer les risques qui peuvent être provoqués par notre protocole de routage sécurisé dans le nouveau système UAS dans son scénario d'utilisation.

3.4.1 Apport de l'approche MDD dans la certification UAS

Le processus de certification d'un logiciel embarqué pour les drones civils fait encore l'objet de recherche et de réflexion par les autorités de certification. Au moment de rédaction de ce manuscrit, aucune norme n'a été définie. Toutefois dans le monde aéronautique, quelques standards (DO-254 [BH07], DO-178B, ARP-4761 [4], ARP-4754 [ARP96]) existent et ont été utilisés par les industriels. Nous avons vu précédemment que la majorité des exigences appliquées dans le monde aéronautique subsistent dans le domaine des drones. Ce constat nous permet dans cette thèse de considérer le standard le plus proche de notre domaine pour la conception des logiciels embarqués aéronautiques. L'idée est de suivre la recommandation de cette norme pour contribuer à la certification de notre algorithme de routage quand un tel standard sera créé. Il s'agit de la famille DO-178 et plus récemment le standard DO-178C [MLD+13].

Cette norme régit les activités de développement et de test des logiciels embarqués à bord des avions et aéronefs commerciaux. Elle a été écrite conjointement par des représentants des constructeurs aéronautiques, des éditeurs de logiciels et des organismes de certification. Ils imposent notamment l'utilisation de l'approche orientée modèle ainsi que l'utilisation de certaines méthodes d'analyses de code. L'application de cette approche peut nous donner les avantages suivants :

— Tout d'abord, il y a la garantie de la procédure de vérification à chaque étape du développement et aussi l'assurance d'une traçabilité complète des exigences, à partir de la spécification jusqu'à l'implémentation dans le code source.

— Ensuite, l'utilisation de la méthode orientée modèle va permettre au logiciel d'être modulaire et réutilisable. Cette propriété de modularité est appréciée d'ores et déjà par les autorités de certification dans le contexte aéronautique. En effet, la garantie de sûreté et de sécurité est plus facile à apporter pour des systèmes modulaires que pour les systèmes monolithiques [AFOTH06].

— Grâce à l'utilisation des méthodes formelles, nous pouvons démontrer l'absence d'erreur de conception du système final. Il est important de noter que les méthodes formelles sont également considérées dans les normes de certifications par la norme DO-331 [Pot12]. Le DO-331 a été créé en 2012 pour être une annexe au DO-178C relatif à l'usage des méthodes formelles pour la certification des logiciels aéronautiques. Le DO-178C se focalise sur le concept de développement orienté modèle tandis que le DO-331 introduit l'analyse par simulation et par couverture du modèle.

  La simulation du modèle permet de valider son objectif en analysant le comportement dynamique du système. La valeur et le comportement obtenus en sortie de la simulation doivent correspondre à ce qui était attendu. En ce qui concerne la

4. http ://standards.sae.org/arp4761/

couverture du modèle, elle est faite durant la phase de simulation du modèle pour comparer les spécifications et les modèles. Les outils Simulink Design verifier et Simulink Code Inspector peuvent être utilisés à ce sujet. La spécificité de chacune de ces deux normes DO-178C et DO-331 est montrée par la table 3.3. Nous avons également inscrit dans le tableau des exemples d'outils qui répondent au besoin de cette norme.

— La norme DO-178C comporte un document annexe consacré à la qualification d'outils. Cette notion de qualification d'outils stipule que les autorités de certification acceptent l'utilisation par les concepteurs logiciels des outils qu'ils ont validés. Ils savent, par exemple, que tel générateur automatique de code ne va pas rajouter d'erreurs par rapport au modèle. En plus de diminuer le temps de développement, et de tests en outils automatisés, ces outils peuvent également diminuer les efforts de certification. Dans notre cas d'étude, plusieurs outils de la gamme de produits Mathworks entrent dans ce cadre. Nous pouvons citer Polyspace, qui est un outil d'analyse statique et de vérification de règles de codage, qui correspond à un des objectifs de la norme. Il y a également Simulink Code Inspector qui répond à pas moins de cinq objectifs de la DO-178C à savoir la conformité du code source avec les exigences bas niveau, conformité du code source avec l'architecture logicielle, lisibilité du code source, traçabilité entre le code source et les exigences et la précision et cohérence du code source.

	DO-178C	**DO-331**
Contenu	- Modélisation - Génération de code source - Obtention de code objet exécutable - Test unitaire sur les partitions logicielles - Traçabilité complète des exigences	- Méthode formelle - Vérification de modèle - Analyse par simulation du modèle - Analyse de couverture du modèle
Exemple d'outils	- Model based design Mathworks - Polyspace - Simulink Code Inspector	- Model coverage - Simulink Design verifier

Tableau 3.3 : Apport des normes DO-178C et DO-331

Pour conclure cette section, nous pouvons donc affirmer que pour assurer la sûreté de notre architecture de communication sécurisée, sa conception doit être réalisée en suivant l'ensemble des recommandations introduites par la norme DO-178C. Cela dans le but de contribuer à la certification du système UAS. Également, le développement du logiciel embarqué critique peut profiter de la productivité, de la réduction de coût et de la qualité du développement orienté modèle. Dans le chapitre suivant, nous présenterons l'utilisation des modèles et des méthodes formelles pour la conception de notre protocole de routage sécurisé.

3.5 Choix d'outils pour l'application de la méthodologie MDD

Nous avons vu précédemment les contenus des sept étapes de la méthodologie. Dans cette section, nous allons associer des outils que nous avons utilisés pour chaque étape de la méthode.

Étape 1 : organisation de l'architecture de communication sécurisée

Comme nous l'avons évoqué précédemment, la première étape de la méthodologie consiste à partitionner les fonctionnalités de notre protocole de routage sécurisé. Cette étape est faite à la main pour échelonner les spécifications en blocs destinés à représenter les classes de partitions. Le diagramme des classes est montré par la figure 3.5 et commenté dans le chapitre suivant concernant la proposition d'un protocole de routage sécurisé.

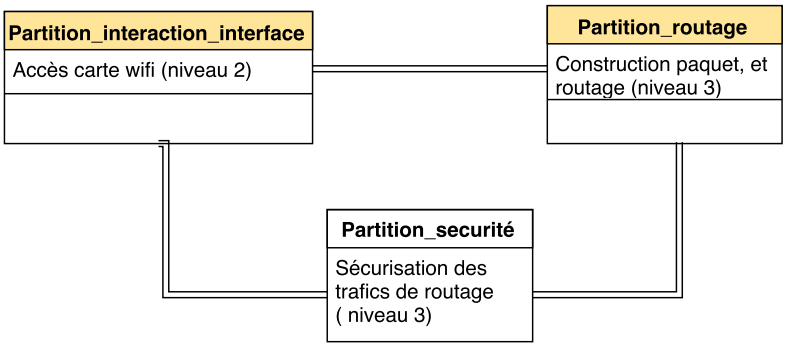

FIGURE 3.5 : Illustration des classes de partition de notre protocole de routage sécurisé

La classe de partition de routage est responsable du routage des paquets. La classe de partition de sécurisation a pour objectif de sécuriser les trafics de routage en garantissant leurs authentifications. La classe de partition d'interfaçage est consacrée à la gestion des interactions des classes gérant les données du routage aux entrées/sorties matérielles. Cette liaison consiste en pratique à réceptionner les trames reçues, à désencapsuler les données de ces trames et à les transmettre à la classe de la partition de sécurisation et de routage. Chaque classe de partition est composée de blocs fonctionnels. Par exemple, le bloc de routage est composé de plusieurs sous-partitions comme le montre la figure 3.6. Dans cette classe de partition de routage, la sous-partition « Demux » est responsable de récupérer les paquets entrants dans le système, ensuite, le système procède à un filtrage suivant une métrique bien particulière (par exemple, l'adresse IP destination, le numéro de séquence) pour transférer le paquet au bloc correspondant. Pour l'instant, nous n'allons pas détailler le contenu de ce bloc de routage. Il sera évoquée au chapitre suivant pour parler des mécanismes de découverte et de maintenance de route de notre protocole de routage.

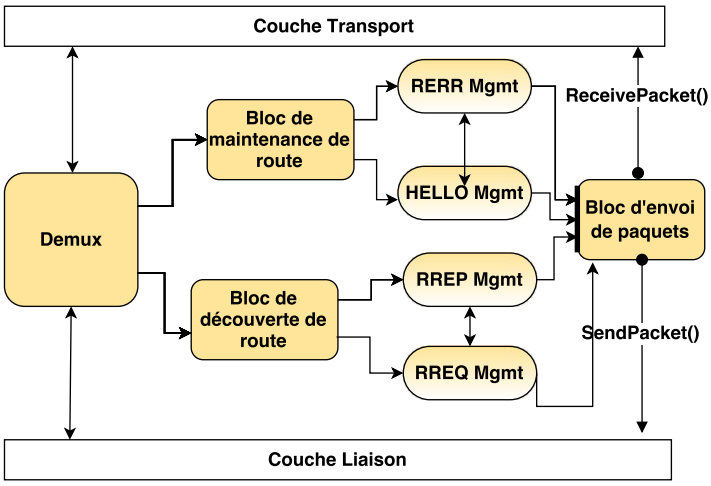

FIGURE 3.6 : Conception de la classe de partition de routage

Étape 2 : modélisation avec Simulink et Stateflow

Les outils de conception tels que Lustre/SCADE ou Matlab/Simulink permettent de contenir la complexité croissante d'abstraction. Notre choix s'est tourné vers Simulink et Stateflow, car ils sont préconisés par les industriels conjointement avec la norme DO-178C. Généralement lorsque les partitionnements des exigences sont faits, le concepteur peut passer à la modélisation. En utilisant Simulink, nous avons pu représenter graphiquement par un diagramme en blocs notre système. Ces blocs sont reliés entre eux par des signaux. Les blocs Simulink sont également formés des opérations arithmétiques ou des générateurs de signaux aléatoires en passant par des opérations écrites en Matlab, C/C++, Fortran ou encore Ada. Simulink permet aussi de simuler le fonctionnement des programmes ainsi que les preuves formelles sur les modèles.

Étape 3 : transformation du modèle

La troisième étape consiste à transformer les modèles en un langage le plus proche de la cible matérielle. Pour cela, nous avons généré du code C avec l'outil Embedded Coder de Matlab/Simulink. Embedded Coder génère un code C et C++ qui peut être utilisé sur les processeurs embarqués. Ce générateur de code permet des optimisations perfectionnées pour contrôler finement les fonctions, fichiers et données du code généré. Il fournit aussi des rapports de traçabilité, une documentation d'interface de code et une vérification logicielle automatisée qui permet de s'intégrer dans une démarque d'un développement logiciel basé sur la norme DO-178B. Avant de générer le code, Embedded Coder lance des outils de vérification du modèle, soit pour remonter des erreurs de conception, soit pour optimiser la configuration du modèle. Les codes sources générés peuvent également être optimisés

spécifiquement par rapport au type de processeur cible qui sera utilisé dans le système embarqué.

Dans le cas de notre protocole de routage sécurisé, les modèles Simulink et Stateflow des classes de partitions ont été convertis en 64 fichiers dont 1 fichier main (5891 lignes de code source en langage C). La figure 3.7 illustre la sortie obtenue par Embedded Coder.

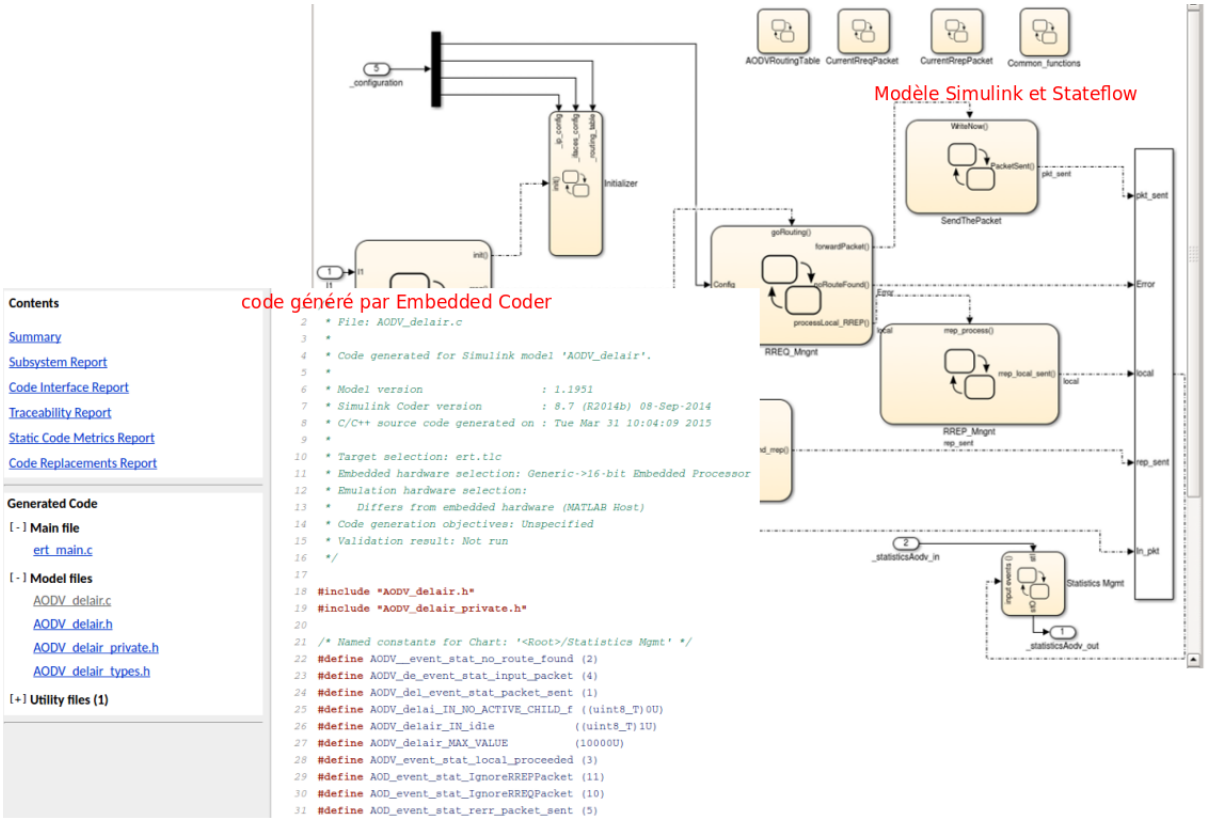

FIGURE 3.7 : Utilisation de Embedded Coder pour transformer les modèles en code source

Étape 4 : code de collage manuel

Le code obtenu à l'étape précédente nécessite d'être lié aux entrées-sorties du système. La première procédure est de gérer le code de communication entre partitions qui sont représentées que par des signaux dans Matlab/Simulink. L'idée est donc de lister ces fils et d'ajouter le code correspondant à l'entrée et à la sortie de chaque partition.

Par ailleurs, nous avons également fourni au système d'exploitation embarqué, un point d'entrée pour chaque partition. C'est principalement la classe de partition de routage qui interagit avec l'interface bas niveau.

En outre, les codes de collage ajoutent également les différentes configurations du système, comme le montre la figure 3.8. Nous avons d'abord la configuration des sockets netlink pour

faire communiquer l'espace utilisateur avec le niveau kernel. Ce paramètre est nécessaire pour appliquer le mécanisme de sécurité adéquat correspondant au type de paquet. Il y a également les codes de mise à jour du noyau du système d'exploitation suivant l'état du *daemon* de routage sécurisé au niveau applicatif. L'encapsulation de paquet IP et le filtrage netfilter sont des approches basiques pour rediriger chaque paquet entrant vers le bloc de sortie correspondant. Ces différentes configurations sont analysées plus en détail dans le chapitre suivant.

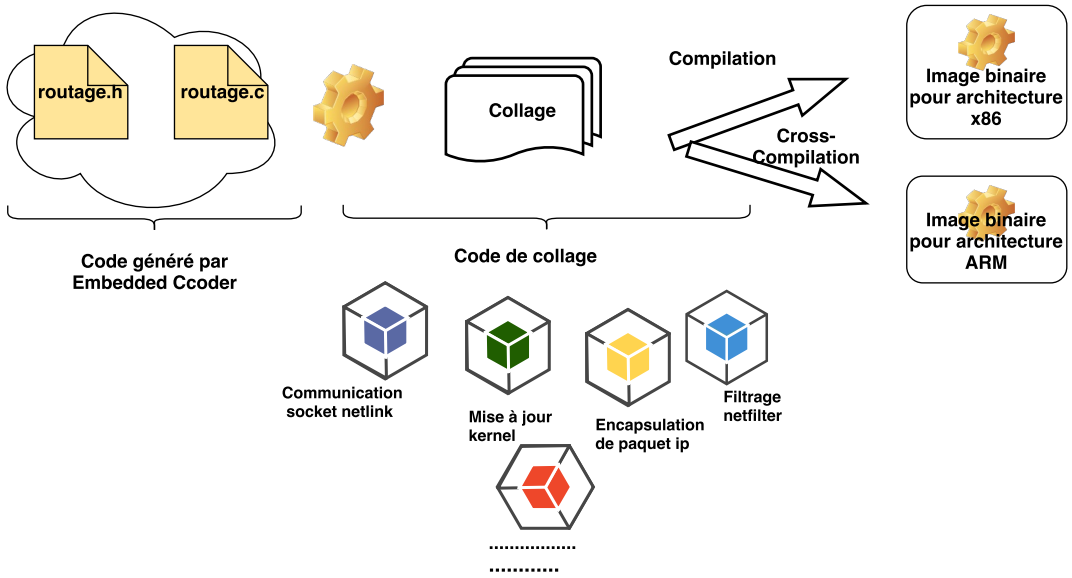

FIGURE 3.8 : Code de collage liant le système d'exploitation au modèle

Étape 5 : compilation avec gcc et arm-gcc du code en langage C

Les codes générés par Embedded Coder et de collage sont indépendants du processeur associé à la plateforme matérielle. La dernière étape consiste donc à obtenir les fichiers binaires exécutables qui seront employés pour chaque cible matérielle. Pour cela, nous avons utilisé à la fois le compilateur gcc (version 4.6.3 ubuntu/Linaro) et un compilateur embarqué de notre cible matérielle. Un compilateur est souvent associé à un *linker*. Ils sont responsables de produire une image binaire correspondant au code source en utilisant les configurations du processeur et la plateforme matérielle et en partageant le code binaire aux adresses mémoires adaptées.

Pour faire fonctionner le code binaire sur notre cible matérielle qui est la carte Phytec [5] utilisé par les drones, il est nécessaire de procéder à un processus de cross-compilation. Le principe du cross-compilation est de compiler des programmes sur une machine ayant

5. http ://www.phytec.fr/produit/single-board-computer/phyboard-mira/

une architecture différente de la cible matérielle qui va utiliser le programme. La raison principale de cette démarche est de favoriser le développement local du logiciel, d'apporter des modifications locales jusqu'à obtenir une binaire d'exécution. Une des raisons aussi est que dans notre cas, la machine locale de développement est plus puissante que la cible. Dans notre cas, le système cible dispose d'un calculateur et de ressources mémoires limitées. Pour cela, nous avons utilisé le toolchain (une série d'instructions permettant au compilateur de connaitre l'architecture cible) se basant sur openembedded [6] car les fichiers sources de notre système d'exploitation embarqué sont générés par openembedded. Ce dernier est un framework libre de compilation de composants logiciels destinés à être déployés sur des systèmes embarqués. Il a été utilisé pour produire l'image Linux utilisée sur notre cible matérielle.

À la fin de cette étape, nous disposons d'un fichier binaire pouvant se lancer dans une machine virtuelle avec l'architecture ARM.

3.5.1 Synthèse sur les chaînes d'outils

Le tableau 3.4 résume les différents outils utilisés pour appliquer la méthode MDD dans cette thèse. Ces outils permettent d'obtenir des gains en matière d'évaluation et de certification. L'évaluation de la sécurité fonctionnelle est simplifiée par l'approche modulaire qui est faite à l'étape de partitionnement et de modélisation. Quant à la certification, la qualification des outils permet de réduire voir d'éliminer les tâches de processus de certification [Var13]. Par ailleurs, ces outils participent également à l'assurance de la qualité du logiciel. Ils assurent que le développement du logiciel s'exécutant dans le système UAS et les différents processus sont conformes aux plans et règles approuvés initialement dans la phase de spécification et de conception du système. Ils garantissent également que les critères de transition pour chaque processus du cycle de vie du logiciel sont satisfaits. La garantie de la conformité entre l'exigence du cahier des charges et du produit logiciel est également atteinte.

Étape	**Outils utilisés**
Outils de modélisation	Simulink & Stateflow
Outils de transformation	Embedded Coder
Langage du code source	C
Compilation et édition de liens	gcc, compilateur embarqué
Système d'exploitation embarqué	Linux généré par openembedded
Emulation/plateforme matérielle	Virtualbox, Architecture ARM

Tableau 3.4 : Outils applicables dans le cadre de notre méthodologie

6. http ://www.openembedded.org/wiki/Main_Page

3.6 Conclusions

Le système UAS est composé de différents modules logiciels qui proposent des services pour la réalisation d'une mission donnée. Pour pouvoir être commercialisé et utilisé en grand public, ce système doit répondre à des exigences de conception et offrir la sûreté de sa navigabilité dans l'espace aérien national. Ces exigences sont gérées par le processus de certification que doivent passer les constructeurs des drones.

Notre cas d'étude se réfère à l'intégration des drones de l'entreprise Delair-Tech dans l'espace aérien français. Bien que le système actuel permette déjà de faire un vol sans visibilité (catégorie des vols *out of line of sight*) entre le pilote et l'aéronef, l'ajout de notre module d'architecture de communication sécurisé vient s'ajouter dans la hiérarchie et nécessite une nouvelle phase de validation du système dans sa globalité. Pour cela, nous devons répondre aux exigences de l'autorité de certification. L'une de ces exigences est la garantie de la sûreté de la mission durant l'opération. Il y a donc un besoin fort d'assurer la correspondance entre les différentes partitions de l'algorithme et son implémentation en code source. Pour cela, nous avons étudié durant cette thèse, l'application d'une approche flexible et cohérente qui supporte le développement de notre protocole de routage sécurisé. L'approche orientée par les modèles nous a permis de raisonner sur les différents besoins qui composent notre cahier des charges, et leur interaction. Elle nous a permis également d'exprimer d'une manière explicite la façon de réaliser ces interactions à travers l'utilisation des modèles afin d'avoir une représentation concise du système. Cela est un gain significatif pour la procédure de certification future de notre système. En outre, cette méthodologie nous a permis de couvrir les différentes phases du cycle de développement en adoptant une démarche basée sur la transformation de modèles en code source. Ce type de pratique permet d'assurer la cohérence du système durant les différentes phases du cycle de vie en garantissant la modularité et la réutilisabilité des différentes fonctions du logiciel.

Chapitre 4

Protocole SUAP (Secure UAV Ad hoc routing Protocol)

Dans ce chapitre, nous présenterons les mécanismes de sécurité du protocole Secure Uav Ad hoc routing Protocol (SUAP), et leur mise en œuvre. Pour concevoir ce protocole, le choix a été fait de s'appuyer sur un algorithme de routage existant. Pour appuyer ce choix, il était nécessaire d'évaluer, dans un environnement UAANET, les performances des algorithmes de routage utilisés pour les réseaux MANET. Cette étude de performance a été réalisée avec un outil hybride de test, qui combine l'approche de simulation (utilisation du simulateur OMNET++), et d'émulation (utilisation des machines virtuelles). L'outil qui a été créé prend en compte un sous-ensemble de l'environnement réel d'exécution d'un réseau UAANET réaliste.

Cette évaluation nous a permis par la suite de distinguer le protocole AODV, qui offre des qualités de services acceptables au regard de notre cahier de charge. Cela a conduit à la création d'une autre instance de ce protocole, avec la méthodologie orientée modèle présentée dans le chapitre 3. Ensuite, nous avons ajouté des mécanismes de sécurité garantissant l'authentification et l'intégrité des messages. Ces services de sécurité sont assurés respectivement par un algorithme de signature, et d'une chaine de hachage. Ensuite, pour contrer l'attaque wormhole, deux mécanismes complémentaires de détection ont été proposés. Le premier se base sur la relation entre le nombre de sauts et la distance géographique relative entre deux nœuds voisins. Le deuxième mécanisme qui garantit la mise en place d'une route sécurisée s'appuie sur une chaine de hachage imbriquée en prenant en compte l'identité des nœuds communicants authentifiés.

Enfin, pour vérifier l'utilité des fonctions de sécurité du protocole SUAP, nous avons vérifié formellement les propriétés de sécurité proposée avec l'outil Automated Validation of Internet Security Protocols and Applications (AVISPA).

Contents

4.1 Introduction 98
4.2 Choix d'un protocole de routage 99
4.2.1 Protocole expérimental et outil de test de protocole de routage UAANET 99
4.2.2 Architecture de l'outil de test 100
4.2.3 Résultats d'évaluation des performances 103
4.2.4 Délai moyen de ré-établissement d'une route après une perte de connectivité 105
4.3 Protocole SUAP 108
4.3.1 Modèle réseau et d'attaques considérées dans la conception du protocole SUAP 108
4.3.2 Description du protocole SUAP 110
4.3.3 Analyse des solutions existantes 111
4.3.4 Protocole SAODV 112
4.3.5 Analyse des vulnérabilités de SAODV 116
4.3.6 Proposition d'un mécanisme pour détecter et contrer l'attaque *wormhole* 123
4.3.7 Proposition d'un mécanisme pour contrer l'attaque *wormhole* réalisé par un seul attaquant 127
4.3.8 Limites du protocole SUAP 129
4.3.9 Différents formats des paquets de contrôle dans SUAP 131
4.3.10 Analyse de sécurité du protocole SUAP 134
4.4 Vérification formelle des propriétés de sécurité du protocole SUAP 135
4.4.1 Nécessité des procédures de vérification 136
4.4.2 Choix de l'outil AVISPA 137
4.4.3 Utilisation de l'outil AVISPA 138
4.4.4 Cas d'application du protocole SUAP 139
4.5 Mise en œuvre du protocole SUAP 142
4.5.1 Architecture de l'algorithme SUAP 142
4.5.2 Modélisation du protocole SUAP 143
4.5.3 Apport de l'approche orientée modèle dans la conception du protocole SUAP 151
4.5.4 Implémentation du protocole SUAP 152
4.6 Conclusions 153

4.1 Introduction

Pour mettre en place un réseau UAANET, quelques protocoles de routage ont été proposés [MML17b] dans la littérature. Le point commun des algorithmes existants est l'absence des mécanismes de sécurité qui permettent de trouver des routes fiables et sécurisées. En effet, différents types d'attaques peuvent être exécutés, comme évoqués dans le chapitre 2. Ces attaques peuvent modifier les paquets de routage, ou injecter de faux messages pour dégrader la performance du réseau. Par conséquent, il est primordial de proposer une architecture de communication sécurisée qui puisse prendre en compte la sensibilité des applications potentielles. En particulier, la sécurité du routage émerge comme une question difficile en raison de la spécificité du réseau en matière de mobilité, de connectivité et de ressources disponibles (énergie, mémoire, bande passante, etc.). De plus, dans le chapitre 2, il a été évoqué que la plupart des protocoles de routage MANET sécurisés existants sont vulnérables contre l'attaque *wormhole*. L'attaque *wormhole* fait partie des attaques les plus réputées qui menacent le routage ad hoc.

Pour répondre à cette problématique, nous avons proposé un protocole de routage sécurisé pour les réseaux UAANET qui s'intitule SUAP (Secure Uav Ad hoc routing Protocol [MML16a], [MML16b]). Ce protocole se base sur le protocole SAODV et sécurise l'étape de découverte et de maintenance des routes. SUAP assure l'authentification et l'intégrité des paquets de contrôle par l'intermédiaire d'un mécanisme de signature et de fonction de hachage. Ensuite, des mécanismes de sécurité supplémentaires ont été ajoutés, pour la détection de l'attaque *wormhole*. Le premier volet de ce mécanisme est dédié aux paquets de découverte de voisin et qui consiste à établir une correspondance entre la distance relative séparant deux nœuds voisins et le nombre des sauts associés au paquet. Le deuxième volet protège les paquets de découverte de route en s'appuyant sur l'utilisation de l'identité du nœud suivant dans la chaîne de la route dans le calcul d'une empreinte (Hash) à chaque retransmission.

Afin de présenter SUAP, ce chapitre sera structuré comme suit. Dans un premier temps, nous détaillerons les hypothèses concernant le modèle réseau et le modèle d'attaque que nous proposons de prendre en compte. Nous analyserons les choix de départ qui ont été faits pour construire la base du protocole SUAP. Par la suite, nous décrirons les différents mécanismes de sécurité du protocole SUAP en détaillant les vulnérabilités qui ont été considérées pendant la phase de conception. Ensuite, nous montrerons l'analyse de vérification formelle que l'on a réalisée pour vérifier les services de sécurité de SUAP avec l'outil AVISPA [ABB+05]. Enfin, dans la dernière partie, nous entrerons dans le détail de conception du protocole SUAP avec la méthodologie de prototypage rapide qui a été présentée dans le chapitre précédent.

4.2 Choix d'un protocole de routage

Pour construire notre architecture de communication sécurisée, il était nécessaire de choisir un protocole de routage de départ. Nous avons donc décidé dans cette thèse d'utiliser un protocole existant dans l'environnement MANET et ensuite d'y ajouter des mécanismes de sécurité. Ce choix est justifié d'une part par la maturité des protocoles de routage MANET (notamment les protocoles AODV, DSR et OLSR) qui sont reconnus dans la littérature à travers des études d'évaluation de performances [Mac99] [PD12] et de l'existence des standards à leur nom. Nous pouvons, par exemple, citer le RFC [1] du protocole AODV [PBRD+03a], de DSR [JHM07] et d'OLSR [CJ03].

Nous nous intéressons également aux métriques de performance réseau pour pouvoir véhiculer les données de la charge utile avec une qualité de service acceptable. Bien que l'objectif principal de notre étude est de mettre en place un réseau de communication sécurisé entre les drones, il est aussi fondamental que le réseau puisse permettre d'échanger des données applicatives (temps réel).

Ainsi, pour pouvoir sélectionner un tel protocole, il est nécessaire de faire une étude d'évaluation des grandes familles de protocole de routage en essayant de se rapprocher au mieux de la condition de déploiement d'un réseau UAANET. Il convient donc de considérer l'environnement d'exécution des algorithmes de routage pour mieux représenter les conditions réelles de déploiement. Les conditions de déploiement évoqué ici concernent : le système d'exploitation Linux utilisé par les drones, l'exécution en temps réel des protocoles de routage et des modèles de mobilité réaliste. Pour répondre à ces exigences, nous avons décidé de mettre en place un protocole expérimental qui va permettre d'évaluer des protocoles de routage suivant des paramètres bien définis propre au réseau UAANET. C'est pourquoi nous avons créé un outil spécifique de test des protocoles de routage UAANET qui combine à la fois le paradigme de simulation et d'émulation. Cet outil [2] de comparaison de protocole peut également être utilisé pour d'autres types de réseaux [MRL15] comme le réseau VANET ou le réseau AANET (Aeronautical Ad hoc NETwork) en ajoutant les modules appropriés.

4.2.1 Protocole expérimental et outil de test de protocole de routage UAANET

Cette étude a pour objectif de justifier quel protocole de routage serait le plus performant dans un contexte d'utilisation réaliste d'une flotte de drone. Le choix doit être fait de manière à maximiser la qualité du service pouvant être offerte aux utilisateurs, par exemple, pour de la vidéo en temps réel dans un contexte de télésurveillance, en considérant entre

1. Request For Comment
2. Il peut être téléchargé sur ce lien : https://dl.recherche.enac.fr/getfile?fileid=42a9e9797239a56118d9c4973516ae1c

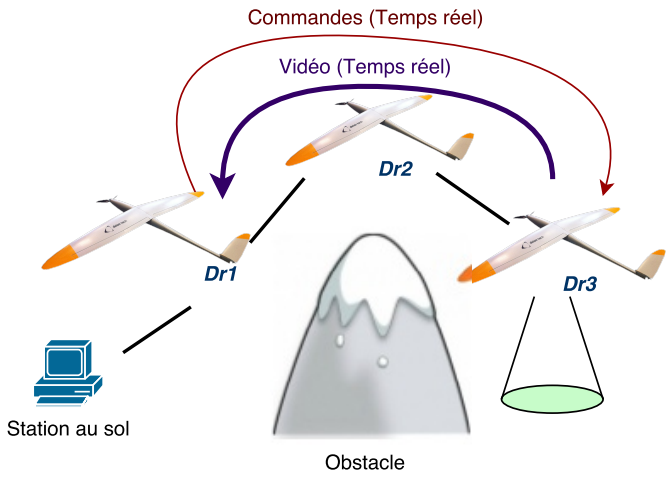

FIGURE 4.1 : Scénario pour lequel notre protocole expérimental doit être adapté

autres la latence d'affichage, la qualité de la vidéo et la disponibilité du service.

Durant notre étude d'état de l'art des protocoles réseaux MANET existants, nous avons remarqué la différence des hypothèses considérées dans les études de simulations dans un environnement MANET et les conditions réelles de déploiement d'un réseau UAANET. Par exemple, beaucoup des tests effectués sur les protocoles de routage MANET supposent un grand nombre de nœuds, de 50 à 200, ce qui ne correspond généralement pas aux situations que nous envisageons. De plus, peu d'études ont été menées spécialement pour les réseaux de drones, dont les caractéristiques en matière de mobilité sont très spécifiques. Enfin, beaucoup d'entre elles se basent sur des résultats issus de simulations et font par conséquent abstraction de nombreux phénomènes pouvant se produire sur un banc de test.

Le but de notre démarche est d'évaluer les performances de différents protocoles de routage dans le cadre d'un scénario proposé. Ce scénario imagine trois drones en mission de télésurveillance. Si les trois drones circulent suivant leur plan de vol, on suppose qu'un obstacle ou une portée limitée puisse venir empêcher la connectivité directe d'un drone avec la station au sol. Dans ce cas, les autres drones doivent pouvoir jouer le rôle de relais vers la station au sol. Ce scénario basique est illustré dans la figure 4.1.

4.2.2 Architecture de l'outil de test

La vue globale de notre outil hybride destiné à tester les protocoles de routage est montrée par la figure 4.2.

Cet outil est détaillé dans l'annexe A.1 et a été publié dans [MRL15]. Dans cette section, nous nous contenterons de citer les principaux modules de l'outil.

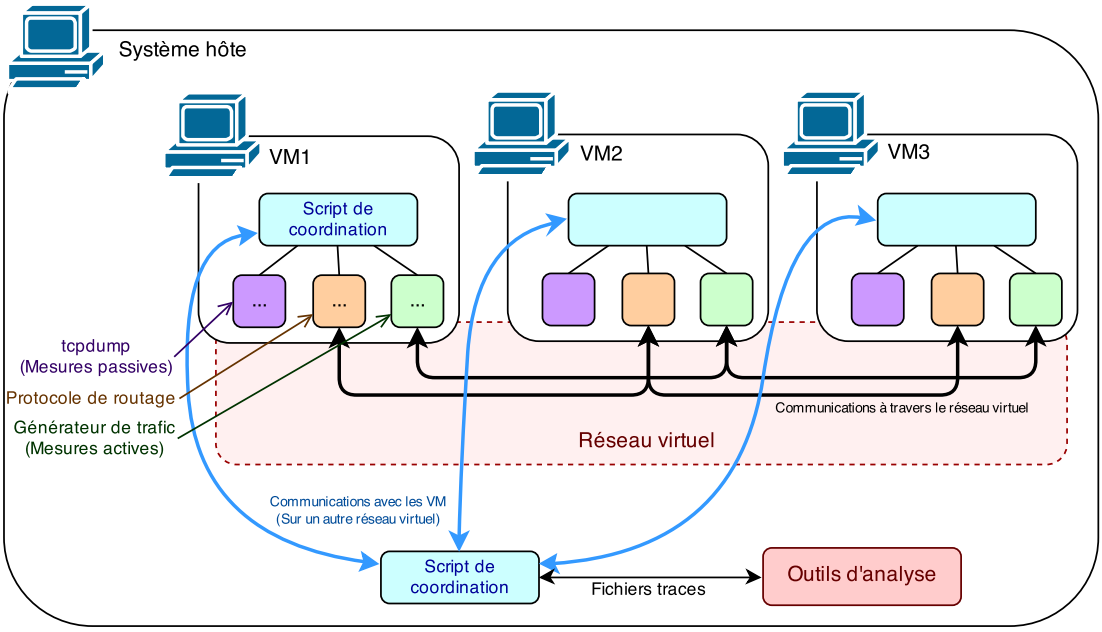

FIGURE 4.2 : Vue globale du système destiné à tester les protocoles de routage par émulation

— Hyperviseur : le choix de l'hyperviseur a été principalement poussé par les possibilités réseau pouvant être obtenues. En effet, nous avons décidé d'implémenter notre solution de routage dans des machines virtuelles pour l'exécution en temps réel dans un système Linux. Notre choix s'est tourné vers *Virtualbox* [Wat08] qui propose un outil simple d'exécution permettant de mettre en communication plusieurs machines virtuelles à travers un réseau virtuel.

— Virtualisation réseau : la mise en place du réseau à nécessité que chacune des machines virtuelles ait une unique interface dans un réseau commun, à l'image d'un réseau sans fil. *Virtualbox* nous a permis de réaliser cette connexion grâce à son système de réseau interne permettant de connecter plusieurs machines virtuelles entre elles. Pour répondre à la question concernant la simulation des pertes de liens qui existent dans une implémentation réaliste d'un réseau UAANET, nous avons inclus la mobilité au protocole expérimental en utilisant le framework virtualmesh [SGB11]. Ce framework permet d'interfacer un système Linux avec une simulation OMNeT++ [3].

— Mobilité : de manière à obtenir un scénario le plus réaliste possible, nous avons utilisé des traces fournies par Delair-Tech pour recréer un scénario de mobilité correspondant au scan d'une zone. À partir d'une trace réelle et grâce à des rotations et changements d'échelle, nous avons recréé un scénario avec 3 drones en scannant une zone en collaboration. Une illustration de ces traces est montrée en annexe A.1.

3. C'est un simulateur réseau qui permet de simuler un réseau ad hoc mobile https ://omnetpp.org/

— Accès au médium : elle est réalisée par le framework Virtualmesh, qui est associé à l'outil OMNET++ et le module INET. Virtualmesh permet ainsi de simuler un accès au médium réaliste (avec collisions, atténuation du signal...).

— Implémentations des protocoles sélectionnés : les protocoles sélectionnés pour une première série d'évaluation sont : OLSR, AODV et DSR. Ces choix ont été motivés parce qu'ils sont des protocoles historiques et traités par de nombreuses publications, et à la base de nombreuses évolutions [Sah14]. Le choix a aussi été justifié par le fait qu'il a été possible d'en trouver leurs implémentations sur Linux.

Par ailleurs, il est important de noter que la troisième grande famille de protocoles de routage qui aurait pu être testée est le routage géographique, avec notamment GPSR. Cette famille de protocoles utilise les positions géographiques pour relayer les paquets. L'information de localisation permet de déterminer le prochain saut pour joindre la destination.

Le choix du prochain saut peut se faire de manière différentes :

— En mode *greedy forwarding* [JMLW08], le nœud envoie le paquet vers le nœud le plus proche de la destination (au sens géométrique du terme) et à sa portée.

— Lorsque ce mode n'est plus possible (situation illustrée dans la figure 4.3) un nouveau mécanisme doit être utilisé. Le *face routing* [LML05] par exemple, consiste à construire un graphe planaire des connections, puis à utiliser la règle de la main droite pour choisir le prochain saut.

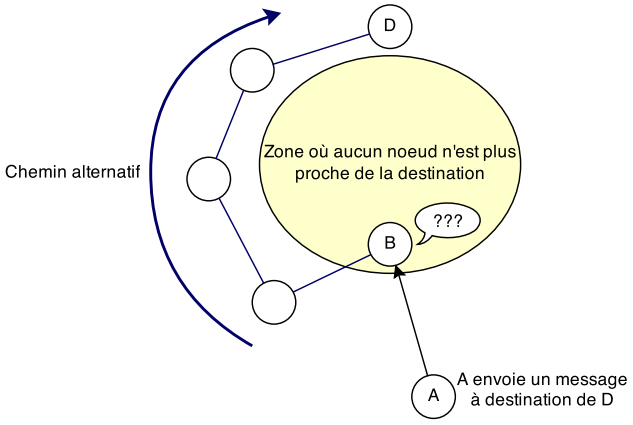

FIGURE 4.3 : Cas où le mécanisme *greedy forwarding* est inefficace

Pour spécifier la position, il y a deux cas à distinguer : lorsque la destination est à une position fixe (comme la station au sol), on pourrait imaginer utiliser un protocole de routage géographique, la donnée pouvant être chargée avant le vol. En revanche, lorsque la destination est mobile, un mécanisme d'échange des positions devient nécessaire.

Dans notre cas, le seul médium qui est disponible est le réseau ad hoc mobile. L'échange des positions nécessite donc une information de routage qui ne peut être fournie que par un protocole de routage. Utiliser un protocole de routage géographique devient donc redondant, car si un mécanisme de routage est nécessaire pour échanger les positions des nœuds, il semble plus pertinent d'utiliser directement les routes proposées par ce mécanisme que d'ajouter un protocole de routage géographique. Nos investigations ont aussi montré que les quelques implémentations disponibles pour Linux sont anciennes et peu nombreuses. Enfin, le routage géographique nécessite que la source du message connaisse la position de la destination, ce qui, dans le cas de notre scénario, n'est envisageable que si un mécanisme d'échange des positions est mis en place. Considérant tous ces éléments, nous avons décidé d'écarter les protocoles de routage géographiques de notre étude comparative.

— Profils des trafics applicatifs générés : nous avons utilisé des traces réelles dont nous avons extrait les principales caractéristiques. Celles-ci correspondaient à des trafics émis lors d'une mission en liaison directe. En extrapolant, nous avons ainsi pu générer un trafic similaire avec nos 3 nœuds, dont les caractéristiques sont exposées dans le tableau 4.1.

Le trafic considéré pour les données utiles correspond à de la vidéo HD, pouvant être généré dans le cas d'une application de vidéosurveillance. Le codec utilisé est du H264, un codec très répandu pour ce type de qualité vidéo. H264 étant un codec à débit variable, nous avons considéré ici un débit moyen de 4 Mbit environ pour une image full HD (1920x1080 pixels) à 30 images/sec. Ces caractéristiques ont été extraites d'une vidéo de promotion produite par Delair-Tech, comprenant essentiellement des images de vol capturées par un drone. Une variabilité uniforme d'environ 50% pour la taille de chaque image a été introduite, cette valeur a été choisie de manière arbitraire. Afin d'éviter une fragmentation des paquets, l'équivalent d'une image est envoyé découpé par paquet de 1000 octets.

Type	Source→Destination	Taille du paquet	Fréquence
Tick	Dr1→Dr2, Dr1→Dr3	64 octets	1.0 paquet/s
Georef	Dr2→Dr1, Dr3→Dr1	64 octets	1.8 paquet/s
Command	Dr1→Dr2, Dr1→Dr3	64 octets	0.034 paquet/s
Vidéo	Dr2→Dr1, Dr3→Dr1	4 Mbit/s	

Tableau 4.1 : Trafics générés

4.2.3 Résultats d'évaluation des performances

Nous avons décidé de mesurer 3 paramètres importants. Ce sont des paramètres utilisés fréquemment pour caractériser les protocoles de routage. En premier lieu, nous avons dé-

cidé d'évaluer le trafic supplémentaire ou *overhead* provoqué par chacun des protocoles. L'*overhead* correspond à la quantité de données additionnelle envoyée pour le fonctionnement du protocole (paquets de contrôle et entêtes dans les paquets de données). C'est un paramètre déterminant quant à la bande passante qui pourra être allouée aux applications. Moins le protocole émet de messages, plus il laisse de place aux applications sur un médium limitée.

Ensuite, nous avons choisi d'évaluer le délai de bout en bout pour chacun des protocoles. Étant donné que la bande passante est largement supérieure au trafic généré, ce paramètre ne devrait pas beaucoup influer. Cependant, il est possible que certains mécanismes des protocoles de routage entrainent des délais supplémentaires. C'est essentiellement e délai maximal qui nous intéresse ici, notamment pour les applications en temps réel.

Enfin, le dernier paramètre est le temps de réparation d'une route. Il caractérise la capacité d'un protocole à modifier rapidement une route si une perte de lien est détectée sur celle-ci. C'est un paramètre important surtout dans un contexte de forte mobilité.

Par ailleurs, il est à noter que nous avons utilisé une couche MAC idéale, où chaque interface est caractérisée par un débit et une portée. Ce choix est justifié par le fait qu'après quelques tests, nous nous sommes aperçus qu'un modèle réaliste au niveau MAC imposait de nombreuses pertes, rendant l'évaluation du protocole de routage difficile. En effet, les pertes occasionnelles causées par la couche MAC permettaient difficilement d'évaluer les pertes causées par le protocole de routage. Nous avons donc remplacé la couche MAC par une couche MAC idéale.

4.2.3.1 Résultats de connectivité

La caractéristique qui nous intéresse concerne la connectivité entre les différents nœuds du réseau. Ici, la connectivité est définie comme la stabilité du réseau qui dépend de la fluctuation des liens utilisés. Si le pourcentage de connectivité est élevé, cela suppose que le protocole de routage offre un bon rendement de stabilité des routes qui sont disponibles dans le réseau. Les résultats que nous avons obtenus concernant la connectivité sont exposés dans les tableaux 4.2. Nous avons choisi les trafics 2→1 et 3→1, car étant les plus importants (en effet ils sont les seuls à représenter l'échange de flux vidéo pendant la mission). La mesure active est donc plus précise.

On constate que OLSR présente ici des moins bonnes performances par rapport aux protocoles réactifs. Ce résultat semble provoqué par le fait que le protocole OLSR attend la perte de plusieurs paquets Hello avant de changer de route. Les résultats les plus intéressants sont ceux concernant la connectivité en état instable. Nous constatons qu'ici, AODV présente de meilleurs résultats par rapport aux autres protocoles. Ce résultat indique que AODV est le protocole qui, lorsqu'au moins l'un des trois protocoles ne permet pas la

	AODV	DSR	OLSR
Durée de la simulation	2h 57min 3s		
États connectés / Durée de simulation	61.5 %	61.0%	58.8%
États instables / Durée de simulation	3.78% soit 6 min 41s		
Connectivité en état instable	88.5%	66.7%	15.3%

Trafic 3→1

	AODV	DSR	OLSR
Durée de la simulation	2h 58min 44s		
États connectés / Durée de simulation	62.9 %	61.7%	60.4%
États instables / Durée de simulation	3.79% soit 6 min 46s		
Connectivité en état instable	90.65%	58.2%	24.1%

Trafic 2→1

Tableau 4.2 : Résultats de connectivité sur un test de 3h

connectivité, est lui-même connecté dans presque 90% des cas.

Il est à noter que les zones à états instables correspondent généralement à des transitions entre un état stable connecté et un état stable déconnecté (cf. figure 4.4). Ces états instables sont les plus intéressants, car ils correspondent à des instants où les protocoles de routage ne se comportent pas de la même manière.

Ce résultat est significatif, il indique que dans pratiquement toutes les situations pouvant être obtenues sur notre scénario, AODV est plus prompt à réagir que OLSR et DSR.

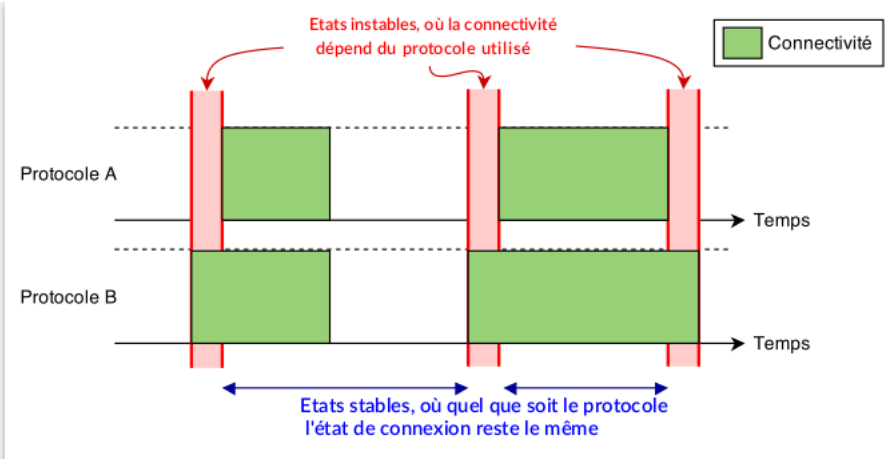

FIGURE 4.4 : Extraction des états stables et états instables

4.2.4 Délai moyen de ré-établissement d'une route après une perte de connectivité

Les résultats obtenus concernant les pertes de connectivité sont exposés dans la figure 4.5. La figure montre que le protocole OLSR est ici largement en retard par rapport aux

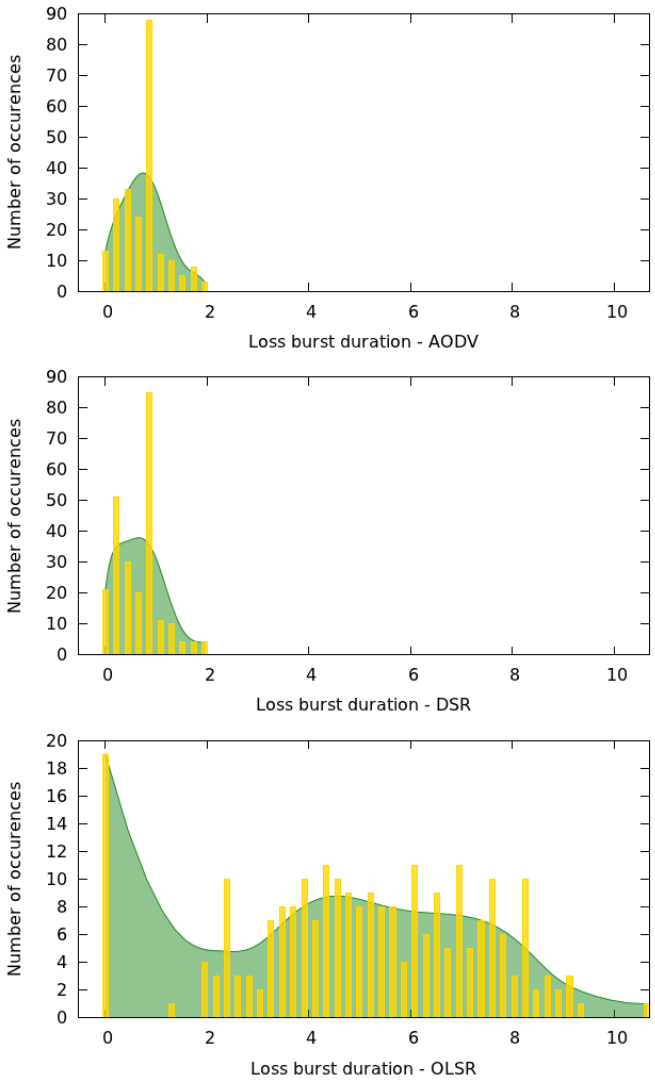

FIGURE 4.5 : Délai de ré-établissement de route après une perte de connexion par protocole (axes des abscissesh en ms)

protocoles réactifs. Ce résultat est justifié par le fait que OLSR ne relance la recherche d'une nouvelle route qu'après la perte de plusieurs paquets. Les résultats pour AODV et DSR sont ici assez similaires, car leur réactivité permet de trouver une route alternative rapidement. Nous pouvons dire que les résultats montrent ici clairement que les protocoles réactifs ont l'avantage, ceci notamment pour leur capacité à réagir promptement à des changements brutaux de topologie. En effet AODV et DSR mettent ici largement moins de temps qu'OLSR pour établir une nouvelle route.

Sur cette figure, la représentation de couleur verte indique la moyenne de la valeur de délai de ré-établissement de route après une déconnexion. La représentation de couleur jaune indique la valeur de l'occurrence pour chaque délai. Par exemple, nous voyons sur la figure que pour AODV, la valeur de rétablissement de route se concentre autour de 1 ms. Cette

valeur est de l'ordre de 4 ms pour le protocole OLSR.

4.2.4.1 Résultats du délai et de l'*overhead*

Le tableau 4.3 résume les résultats que nous avons obtenus sur l'*overhead*, et le tableau 4.4 sur les délais. Au vu des résultats que nous obtenons, il est clair que DSR reste largement en retard (par rapport à ces métriques) vis-à-vis d'AODV et OLSR. Nous pouvons expliquer ce résultat par le fait que DSR nécessite l'utilisation d'acquittements pour déterminer la perte ou non d'un paquet.

Il est important de remarquer que notre implémentation de l'algorithme DSR était compatible avec les acquittements passifs, mais le nombre d'acquittements émis avec DSR ne réduit pas autant. En effet l'acquittement passif ne peut être utilisé que si le nœud suivant sur la route est un nœud relais, un paquet étant acquitté lorsque le nœud précédent détecte que le paquet a été relayé. La part des émissions destinées à des nœuds relais étant de 24% dans notre test, cela ne représente pas un gain suffisant pour rivaliser face à AODV et OLSR.

Nous pouvons également constater ici que les délais sont très faibles. Les protocoles ne semblent pas introduire un délai supplémentaire significatif. Cependant, il est à noter que le protocole DSR est légèrement plus lent que les autres. Ceci est dû au fait que DSR nécessite l'attente d'un acquittement après chaque paquet. Également la taille des entêtes des paquets de contrôle de DSR provoque un léger délai à chaque retransmission. Ensuite, en matière d'*overhead*, DSR reste bien en retard, et AODV garde un léger avantage sur OLSR.

	AODV	DSR	OLSR
Paquets de contrôle	501ko	75999ko	438 ko
% Trafic (octets)	0.034%	4.393%	0.027%
Entêtes ajoutés aux paquets de données	0	26723ko	0
Entête moyen	0	17.8 octets	0

Tableau 4.3 : *Overhead* (au niveau de l'interface du drone 1)

	AODV	DSR	OLSR
Délai moyen	5.32 ms	10.15 ms	5.91 ms
Délai maximal	100.0 ms	100.0 ms	99.8 ms

Tableau 4.4 : Délais moyens et maximums (au niveau de l'interface du drone 1)

Nous constatons également que DSR provoque un *overhead* largement supérieur par rapport à AODV. En matière de délai, AODV présente également des avantages. Compte tenu du caractère temps réel et de l'occupation en bande passante des données utiles, il est judicieux de choisir le protocole offrant moins de délai et d'*overhead*. Par conséquent,

le protocole AODV sort gagnant de cette comparaison, car il propose dans le cas de notre scénario un taux de connectivité plus élevé, un *overhead* limité et un délai acceptable. Nous constatons également que le protocole AODV offre une bonne adaptabilité vis-à-vis de la mobilité des nœuds dans un réseau en termes de délai de reconstruction d'une route.

Dans ce qui suit, nous allons tenir compte de ce résultat et utiliser le protocole AODV comme le squelette de notre protocole de routage sécurisé. En effet, le protocole SUAP se base sur le protocole AODV pour les mécanismes qui sont en lien direct avec l'établissement d'une route dans le réseau. De plus, dans la section 4.5, nous présenterons la mise en œuvre orientée modèle de ce nouveau protocole en utilisant la méthodologie de prototypage rapide que nous avons détaillée dans le chapitre précédent.

4.3 Protocole SUAP

4.3.1 Modèle réseau et d'attaques considérées dans la conception du protocole SUAP

Dans cette partie, les hypothèses que l'on a considérées pour la conception du protocole SUAP seront présentées.

— Comme dans tout type de réseau MANET, le protocole de routage véhicule deux types de messages : les paquets de routage et l'information prise par la charge utile, qui est dans notre cas la vidéo capturée par un drone. Ces deux types de messages ont différentes natures et leurs exigences en matière de sécurité différent. Dans cette thèse, nous ne focalisons que sur la sécurité des paquets de contrôle liés au protocole de routage. Ces paquets sont généralement envoyés entre voisins, et peuvent donc rencontrer un nœud malveillant sur une partie du chemin entre l'émetteur et le destinataire.

— En outre, les nœuds ont besoin d'une infrastructure à clés publiques pour générer et révoquer les clés cryptographiques utilisées. Les services fournis par une PKI passent par des techniques pour la création, la gestion, le stockage et la révocation des certificats. La mise en place de ces services s'appuie sur des éléments tels que l'autorité de certification, l'autorité d'enregistrement et le service de publication. Cette problématique a été étudiée sous forme d'état de l'art au cours de la thèse, mais n'est pas assez mature pour être présentée dans un chapitre dans ce manuscrit. Une section sera consacrée à ce sujet dans la partie perspective à la fin de ce manuscrit. Pour la suite, nous considérons l'existence d'une infrastructure de gestion de clés fiable et sécurisée qui est en charge de la gestion, et de l'invalidation des clés.

— Nous considérons également que les nœuds du réseau sont homogènes ; cela veut dire que les drones et les stations sol utilisés dans notre étude proviennent d'un

même fabricant (Delair-Tech). Cette hypothèse est faite pour ne pas considérer les vulnérabilités causées par un réseau hétérogène. Les travaux cités dans [EvHHW08] et [LZW+09] stipulent que la considération d'un groupe de nœuds hétérogènes peut engendrer des problèmes de sécurité associés à l'égoïsme des nœuds. En effet, si le réseau est constitué des groupes de nœuds hétérogènes, le réseau unique doit offrir des services réseau répondant aux différents cahiers des charges. La mutualisation de ces cahiers des charges peut induire une consommation inégale de la ressource réseau. Cela peut générer un risque d'égoïsme des nœuds du réseau. Nous supposons donc un réseau homogène dans lequel un nœud du réseau en fonctionnement normal ne peut pas décider de ne pas retransmettre un paquet pour économiser sa batterie.

— Nous considérons également que les nœuds du réseau ont suffisamment de ressources pour appliquer les algorithmes cryptographiques tels que le RSA-512 ou le SHA-256. Les ressources de traitement des drones DT-18 sont suffisamment importantes pour justifier cette hypothèse. Cela nous permet donc d'appliquer les algorithmes qui sont jugés lourdes et gourmandes en ressources (mémoire, énergie) dans les autres types de réseau ad hoc [PST+02][MMS09] (par exemple, les réseaux de capteurs).

— Il est aussi important de préciser que tous les nœuds du réseau sont équipés d'une antenne omnidirectionnelle. La portée de la communication est limitée pour chaque nœud ; cela signifie que la distance entre les nœuds peut être représentée par la variable $r < D_{max}$. D_{max} étant la distance maximale de la portée d'un nœud.

— Nous utilisons dans cette étude des algorithmes cryptographiques déjà existants. Nous avons utilisé l'algorithme RSA [ZT11] et SHA-256 [GH03] respectivement pour la signature numérique et la chaîne de hachage. Ces choix sont justifiés par la robustesse de ces algorithmes qui permettent déjà de répondre au besoin de sécurité dans notre cahier des charges. Notre protocole de routage peut également utiliser d'autres algorithmes similaires comme le SHA-512 suivant un besoin spécifique de sécurité. Nous supposons également que ces algorithmes sont suffisamment robustes et que les attaques spécifiques à leur encontre ne sont pas considérées (*brute force attack* [Mih07]).

— Nous supposons que les différents nœuds du réseau sont synchronisés temporellement. Ceci est rendu possible grâce à l'utilisation des équipements GPS à bord des drones. Cette hypothèse est prise en compte pour appliquer notre algorithme de *geographical leashes* que l'on détaillera dans la section 4.3.6. Cette possibilité de synchroniser les drones est l'une des spécificités particulières du réseau UAANET contrairement aux réseaux MANET où il est souvent difficile d'appliquer cette hypothèse [JK09]. Par ailleurs, d'autres outils existent permettant la synchronisation des nœuds. Nous pouvons cité la mise en place d'un serveur NTP [Mil85] qui à partir d'un serveur de synchronisation peut synchroniser un groupe de nœuds. Toutefois, son inconvénient est l'ajout de trafics supplémentaires sur le réseau par l'échange des paquets NTP.

On peut également parler des problèmes de précision de l'ordre de la milliseconde du protocole NTP qui n'est pas acceptable au regard des performances souhaitées. Il faut aussi signaler que l'utilisation du protocole NTP va ajouter de la vulnérabilité puisqu'il faut sécuriser à la fois les paquets et le serveur de synchronisation.

— Dans notre étude, nous n'assurons pas la confidentialité des paquets de routage. Ce service de sécurité n'est pas primordial dans un réseau où tous les nœuds se trouvant au sol ou en vol peuvent joindre le réseau. Cela signifie que les nœuds externes n'appartenant pas à notre réseau peuvent écouter les paquets de contrôle échangés en se mettant à la portée d'un émetteur. Si un attaquant intercepte un paquet de routage, ses actions par rapport à ce paquet sont limitées, car toute action de modification sera détectée par des mécanismes d'authentification de messages. Aussi, en cas de réinsertion du paquet, le numéro de séquence (qui est authentifié) dans le paquet permettra de détecter l'ancienneté du message. Ce choix est aussi justifié par le fait que le message de routage en lui-même ne constitue pas une information secrète, car aucune information sensible n'est décrite dans les formats des paquets de routage.

— Nous considérons un modèle d'attaquant suivant le modèle de Dolev-Yao[Cer01] . Le modèle de Dolev-Yao suppose que l'intrus peut intercepter les messages cryptés, mais ne peut les modifier sans la clé de sécurité correspondante. Comme énoncée précédemment, les attaques sur la méthode de chiffrement ne sont pas considérées. Cette hypothèse considère le chiffrement parfait. Ainsi, pour déchiffrer un message, il est nécessaire de connaitre la clé de déchiffrement correspondante. La connaissance d'un intrus se limite aux clés publiques de tous les nœuds, sa propre clé privée, les identités des agents (adresse IP) et aux éventuelles données publiques du protocole. En conséquence, les attaques présentées dans la section 2.3.4 peuvent être exécutées dans le réseau.

4.3.2 Description du protocole SUAP

Le protocole SUAP [MML16a] est un protocole sécurisé réactif qui a été mis en œuvre pour s'exécuter dans les réseaux ad hoc de drones. Il se base sur le protocole AODV [PBRD03b] et SAODV [Zap02]. Il est caractérisé par des mécanismes de sécurité qui garantissent l'authentification des champs non mutables (c'est-à-dire des champs du protocole de routage qui restent statiques à l'émission jusqu'à la réception du paquet par le destinataire, par exemple, les adresses des nœuds source et destinataire), l'intégrité des champs mutables (par exemple, le compteur de nombre de sauts) et la non-répudiation. Il est également composé des mécanismes de détection de l'attaque *wormhole*.

Dans ce qui suit, nous verrons dans un premier temps les mécanismes et les vulnérabilités du protocole SAODV. L'exécution de l'attaque *wormhole* dans un réseau UAANET sera

étudiée. Nous montrerons également une vulnérabilité particulière du protocole SAODV qui est causée par un nœud attaquant faisant une écoute illicite, qui transfère un paquet, mais qui ne le modifie pas pour ne pas être détectée par le mécanisme d'authentification employé. Cette attaque (similaire à l'attaque *wormhole*, mais qui est exécutée par un seul nœud) a pour objectif de faire croire à deux nœuds distants qu'ils sont voisins. Ensuite, dans la deuxième partie, une analyse détaillée des travaux existants dans le but de contrer ces attaques sera présentée pour mettre en avant leurs inconvénients. Et finalement en troisième partie, la spécification du protocole SUAP sera détaillée.

4.3.3 Analyse des solutions existantes

Au moment de la rédaction de ce manuscrit, aucune étude de sécurité associée aux protocoles de routage UAANET n'a été publiée. Quelques protocoles de routage ont été proposés, comme nous l'avons évoqué dans le chapitre 1, mais aucun d'entre eux n'a considéré la problématique de la sécurité du réseau UAANET.

Néanmoins, des études de sécurité relatives à la communication directe entre une station sol et un drone existent dans la littérature. Dans ce type d'architecture, l'objectif est de sécuriser les modules physiques du système et de restreindre l'accès au seul lien de communication. Puisque ce genre de configuration n'est pas similaire à la configuration de notre architecture, les solutions existantes relatives à ces sujets n'ont pas été considérées. Dans le contexte d'une flotte de drones, l'étude la plus proche de notre problématique est celle d'Arkam et al. dans [ABC+] qui s'intéresse à l'amélioration de la sécurité d'une flotte de drones au travers d'un module physique embarqué sur chaque drone de la flotte et permettant de garantir un certain nombre de services de sécurité. Dans notre travail, il a été décidé dès le départ de ne se focaliser que sur une solution logicielle en tenant compte de l'architecture physique existante du système UAS composé des drones DT18. Nous nous sommes alors tournés vers les réseaux MANET dans lesquels quelques protocoles de routage sécurisés ont été proposés comme nous l'avons montré dans le chapitre 2. Compte tenu du résultat des évaluations effectuées dans la section précédente, nous avons fait le choix de considérer le protocole SAODV. Ce protocole a été considéré pour les raisons suivantes :

— l'authentification et l'intégrité des messages de routage sont apportées à l'exception des failles de sécurité dont on discutera dans la section suivante. SAODV permet de contrer les attaques en vérifiant l'authentification des messages par un procédé de signature et hachage.

— le protocole SAODV permet de contrer l'attaque *blackhole* en s'assurant qu'un nœud n'appartenant pas au réseau ne puisse acheminer de faux paquets suite à une demande de route. Dans le protocole SAODV, deux situations sont considérées pour répondre à une requête de route. Soit le nœud destinataire répond directement à la

requête afin d'éviter qu'un nœud malveillant ne puisse créer une route fictive. Soit un nœud intermédiaire est autorisé à répondre, mais à condition qu'il ait été bien authentifié au préalable.

— le protocole SAODV assure aussi l'intégrité du nombre de sauts à chaque retransmission. Ceci est important pour éviter qu'un nœud malveillant ne le modifie. Ce mécanisme est réalisé avec une fonction de hachage à sens unique.

— par rapport aux protocoles existants (notamment ARAN et SRP qui ont été étudiés dans le chapitre 2), SAODV n'est pas coûteux, car un message de routage n'est pas signé à chaque retransmission. La signature n'est effectuée que par les nœuds source et destination. Les nœuds intermédiaires se contentent de vérifier la signature à chaque saut. La différentiation de l'approche utilisée pour les champs mutables et non mutables est alors appropriée.

4.3.4 Protocole SAODV

L'idée principale du protocole SAODV consiste à faire usage d'une signature pour protéger les données statiques des messages de contrôle à l'aide d'un algorithme de chiffrement asymétrique, puis de recourir à une chaîne de hachage dans l'optique de protéger l'intégrité de la partie variable dont le compteur de nombre de sauts.

SAODV propose le principe de la double signature qui permet d'authentifier les paquets de réponse spécifiquement envoyés par des voisins qui connaissent la destination [4].

4.3.4.1 Signature numérique dans SAODV

Comme nous l'avions montré précédemment, le premier mécanisme de sécurité dans SAODV est la signature numérique qui est une forme de chiffrement asymétrique permettant de garantir l'authenticité des informations non mutables. Le protocole SAODV s'appuie sur les champs du protocole AODV et s'ajoute en tant qu'extension (la figure 4.6 illustre le format d'extension de SAODV). Un nœud émetteur d'un paquet RREQ ou RREP souhaitant joindre un autre nœud, signe chaque message transmis, ce qui permet d'éviter la participation des nœuds externes malveillants. À la réception de ces messages, les nœuds intermédiaires vérifient d'abord l'authenticité du message avant de traiter le paquet.

Par ailleurs, il est important de préciser qu'il existe deux types de formats pour les paquets de découverte de route : les paquets avec une seule signature et les paquets avec une double signature. L'ajout du principe de la double signature est justifié par le constat qu'un nœud malveillant peut répondre à une requête par un faux paquet de réponse. En effet, dans

4. Il est important de préciser que conceptuellement, à notre sens, le principe de double signature n'apporte rien de plus en matière de sécurité. Nous pouvons bien imaginer désactiver le champ Destination Only dans le paquet requête pour ainsi permettre à un nœud voisin quelconque de répondre s'il connait une route vers la destination

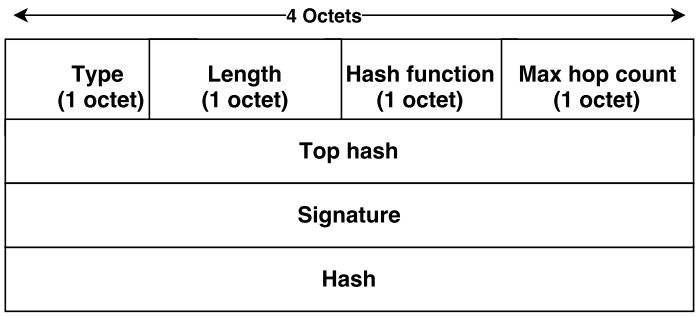

FIGURE 4.6 : Format d'extension d'un message SAODV

le protocole AODV, un nœud intermédiaire peut répondre à une demande de route s'il connait un chemin vers la destination recherchée. Toutefois, cet enchaînement ne peut se faire avec l'utilisation d'une seule signature, car ce nœud intermédiaire ne peut pas générer un paquet RREP signé par la destination. Pour résoudre cette limitation, le principe de la double signature a été ajouté. Elle est utilisée comme suit :

supposons que l'on se trouve dans la configuration de la figure 4.7, quand le nœud **S** envoie un paquet requête, il inclut également une deuxième signature (cf. figure 4.8) pour permettre aux nœuds intermédiaires de répondre à une future requête ayant comme destination le nœud S. Supposons, par exemple, qu'à l'état **t0**, le nœud **S** cherche une destination vers le nœud **D**. Dans ce cas, le paquet RREQ est envoyé et retransmis jusqu'au nœud **D**. Supposons ensuite que le nœud **D** cherche une route vers le nœud **S** ; lorsque le paquet RREQ du nœud **D** arrive au nœud **N2** (que l'on suppose être connecté au nœud **S** à travers **N1**), il peut répondre à la place de **S** en envoyant un RREP vers **D**. Cet algorithme peut parfaitement fonctionner dans un contexte MANET, mais il y a quelques points qu'il faut bien remarquer quant à son application dans notre modèle réseau.

— L'utilisation d'une deuxième signature n'est justifiée que si les nœuds sont hétérogènes et qu'ils peuvent être malveillant ou dévient égoïste. Par exemple, un nœud peut créer de faux paquets pour éviter d'être inclus dans le routage des données. Puisque nous supposons que les nœuds du réseau sont homogènes, nous faisons donc abstraction de ce principe de double signature.

— La deuxième signature est nécessaire si nous ne pouvons pas garantir l'authentification de son voisin à un saut. Par exemple, sur la figure 4.7, le nœud **D** peut ne pas faire confiance à **N3**, car ce nœud peut être un nœud qui vient de joindre le réseau. Dans notre protocole de routage, SUAP est doté d'un algorithme d'authentification de voisins qui est exécuté durant la phase de découverte des voisins durant l'échange des messages Hello. Comme nous le verrons, ce procédé permet de garantir que son voisin est fiable. Ce qui implique qu'un nœud du réseau peut autoriser son voisin de répondre à sa place sans la nécessité d'une double signature.

FIGURE 4.7 : Illustration du mécanisme de chaîne de hachage dans SAODV

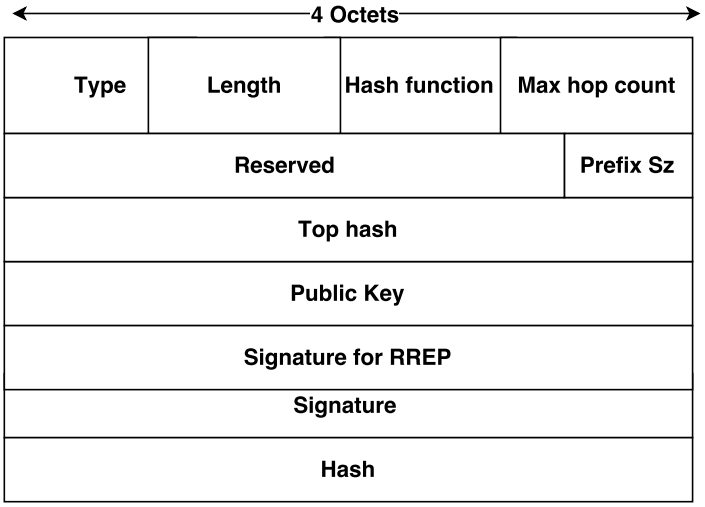

FIGURE 4.8 : Format d'extension d'un message RREQ avec double signature

Le format de l'extension du paquet requête avec double signature (qui est est illustré par la figure 4.8) et expliqué ci-dessous.

— Type : indique le type de paquet (égal à 64).
— Length : est la taille en octets du paquet en n'incluant pas le champ *Type* et *Length*.
— Hash function : est le pointeur de la fonction de hachage utilisé pour calculer la valeur de Hash et de *Top hash*.
— Max Hop Count : est le nombre de sauts maximal qui peut être paramétré à 64 (valeur de TTL) (*network lifetime*).
— Top hash : est une variable qui permet d'authentifier l'intégrité du nombre de sauts.
— Signature : est le pointeur de la méthode de signature utilisée.
— Reserved : paramétré à zéro.
— Prefix Size : est la taille du champ *Prefix* du paquet RREP.
— Signature for RREP : est la signature pour le paquet RREP. Il est utilisé pour permettre à un nœud intermédiaire ayant reçu ce paquet de requête auparavant de répondre à la demande de route.
— Signature : est la signature de tous les champs non mutables.
— Hash : est la valeur de l'empreinte qui correspond à la valeur courante du nombre de sauts.

4.3.4.2 Chaîne de hachage dans SAODV

En ce qui concerne la chaîne de hachage, une fonction **h** est utilisée et qui est connue, à l'initialisation, par les nœuds participant au routage. Cette fonction est choisie par le nœud source. Ce choix est ensuite considéré comme un champ non mutable et protégé donc par une signature. Ici, il est important de noter que ce principe d'obfuscation de la fonction **h** ne considère pas le principe de Kerchoffs [Pet11]. Toutefois, ce principe reste acceptable dans un contexte temps réel, car si on utilise une fonction de hachage basée sur une composition de fonctions (comme étudiée dans [NS11]), l'utilisation d'une fonction de hachage qui est connu à l'avance que par tous les nœuds légitimes reste acceptable. En effet, le délai que met l'attaquant pour trouver la fonction peut être suffisamment important pour que, quand il le trouve, son action n'impacte pas l'intégrité des messages de routage (qui sont exécutés en temps réel).

Pour l'illustrer la construction de la chaîne de hachage, nous prendrons l'exemple suivant dans lequel nous considérerons la figure 4.7.

Sur cette example le nœud source **S** cherche une route vers une destination **D**. Il fait donc les opérations suivantes :

1. Choix de la fonction $Hash\_Function = \mathbf{h}$.
2. Choix d'un nombre aléatoire *seed*
3. Choix du champ Max_Hop_Count qui dépend du diamètre du réseau (qui peut être égal à TTL = 64).
4. Faire Hash = seed
5. Calcul de $Tophash = h^{Max\_Hop\_Count}(seed)$. Dans cette expression,
   - h indique la fonction de hachage
   - $h^i(data)$ est le résultat de l'application de la fonction h, i fois à la donnée $data$

À la réception du message par **N1** :

1. Calcul de $h^{Max\_Hop\_Count-hopcount}(Hash)$ et vérification si c'est égal à la valeur du champ $Tophash$. Comme $Hash$ est égal à $seed$ et que le nombre de sauts actuel est égal à 0, nous avons $h^{Max\_Hop\_Count-0}(seed) = Tophash$. Cette égalité nous indique que le nombre de sauts n'a pas été modifié.
2. Si la vérification est positive, le nœud N1 incrémente ensuite le nombre de sauts et applique la fonction de hachage sur la valeur précédente de $Hash$ (qui est égale à $seed$) avant de renvoyer au nœud N2.
   La nouvelle valeur de Hash est donc $Hash = h(seed)$.

À la réception du message par **N2** :

1. Calcul de $h^{Max\_Hop\_Count-hopcount}(Hash)$ et vérification si c'est égal à la valeur du champ $Tophash$. Comme $Hash$ est égal à $h(seed)$ et que le nombre de sauts actuel est égal à 1, nous avons $h^{Max\_Hop\_Count-1}(h(seed)) = Tophash$. Cette égalité nous indique que le nombre de sauts n'a pas été modifié.

2. Si la vérification est positive, le nœud N2 incrémente ensuite le nombre de sauts et applique la fonction de hachage sur la valeur précédente de $Hash$ (qui est égale à $h(seed)$) avant de renvoyer le paquet.
   La nouvelle valeur de Hash est donc $Hash = h^2(seed)$.

Il est important de préciser que le choix de différencier les champs mutables avec les champs non mutables est justifié par le besoin en performance du réseau. En effet, si tous les champs sont signés, cela peut générer une quantité importante d' *overhead* et ajouté du délai. C'est pour remédier à cela que SAODV ne signe pas tous les messages à chaque retransmission, mais vérifie uniquement l'intégrité du seul champ variable (nombre de sauts), par une chaîne de hachage (qui est moins coûteux).

4.3.5 Analyse des vulnérabilités de SAODV

4.3.5.1 Attaque *wormhole*

L'attaque *wormhole* est une attaque sévère qui vise le processus de routage. Elle est effective contre les protocoles de routage réactifs utilisant le nombre de sauts comme métrique de sélection de routes. Il est à noter que l'attaque *wormhole* peut être lancée en utilisant deux ou plusieurs attaquants pour transporter les paquets d'un endroit à un autre dans le réseau. Cette attaque crée une perturbation au niveau du routage des paquets puisque les nœuds distants vont croire qu'ils sont voisins. En conséquence, les nœuds légitimes sont obligés de suivre l'algorithme du protocole de routage qui est de choisir le chemin le plus court et qui passe par un tunnel *wormhole*.

La figure 4.9 ci-dessous illustre un exemple d'attaque *wormhole*. Il donne l'illusion à Dr1 et à Dr4 qu'ils sont voisins. Sur cette figure, l'attaquant A1 va transférer tous les paquets du nœud Dr1 directement vers Dr4 à travers l'attaquant A2. En conséquence, le nœud Dr4 va croire que Dr1 est a sa portée (c'est-à-dire son voisin). Cette attaque va transmettre donc tous les paquets de contrôle échangés c'est-à-dire (Hello, RREQ, RREP, RERR). Même en n'ayant aucune connaissance des clés cryptographiques ou encore de la fonction de hachage utilisée, les attaquants peuvent dégrader l'intégrité du réseau en transitant les paquets de contrôle et par la suite, capturer les trafics de données échangés.

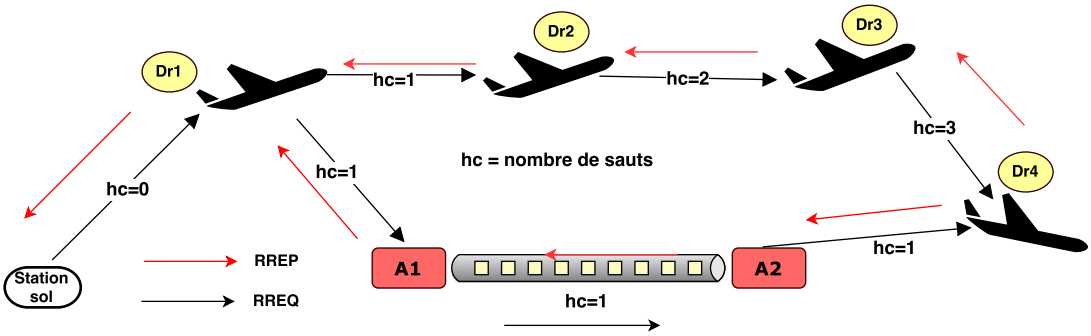

FIGURE 4.9 : Illustration de l'attaque *wormhole*

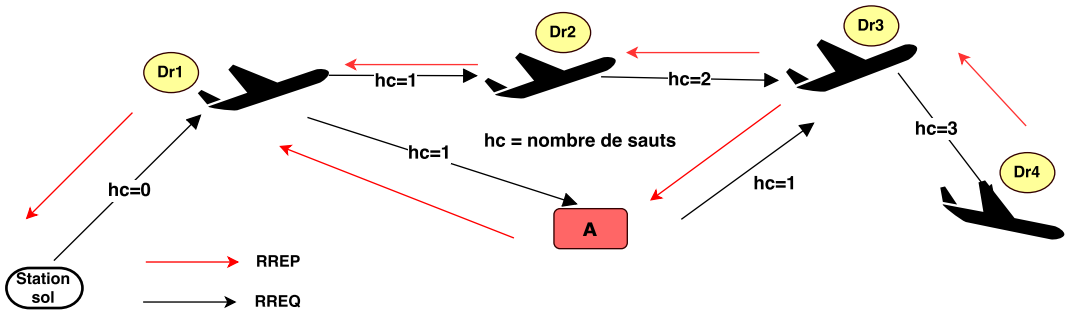

FIGURE 4.10 : Illustration de l'attaque avec un seul attaquant exécutant une version simplifiée de l'attaque *wormhole*

4.3.5.2 Attaque avec un seul attaquant

Comme énoncée dans [ZA02a], le protocole SAODV est également vulnérable contre un seul attaquant qui capture des paquets de routage, mais décide de ne pas modifier le paquet. Cette attaque est possible en raison de la fragilité de l'approche se basant par la fonction de hachage dans SAODV. Comme nous le voyons sur la figure 4.10 ci-dessous, un nœud attaquant A1 qui n'a pas la fonction de hachage utilisée dans le réseau n'incrémente pas le nombre de sauts, afin de réduire le nombre de sauts passant pour lui. Pour réaliser cette attaque, le nœud A va se mettre dans la portée des deux cibles c'est-à-dire Dr1 et Dr3. Nous pouvons donc conclure que la distance entre Dr1 et A et entre A et Dr3 n'excède pas la portée maximale de couverture à un saut. Cette valeur maximale étant égale à D_{max}.

En appliquant l'algorithme de chaîne de hachage qui existe dans SAODV, les nœuds échangent les messages suivants :

Dr1 → Dr2 : (64, Top_hash, Hash_function, Max_hop_count, signature, Hash) avec :

— 64 indique que c'est un paquet de requête.

— $Hash\_function = h$

— $Tophash = h^{Max\_hop\_count}(seed)$

— $Hash = h(seed)$

Et après vérification, le nœud Dr2 envoie le message :

Dr2 → Dr3 : (64, Top_hash, Hash_function, Max_hop_count, signature, Hash) avec :

— $Hash\_function = h$

— $Top\_hash = h^{Max\_hop\_count}(seed)$

— $Hash = h(Hash) = h(h(seed))$

En présence de l'attaquant, le nœud Dr3 recevra donc deux types de messages :

1. Le premier message venant de N2 va être authentifié en comparant à Top_hash, la valeur $h^{Max\_hop\_count-2}(h^2(seed)) = Top\_hash$
2. Le deuxième venant de A1 va être authentifié en comparant à Top_hash la valeur $h^{Max\_hop\_count-1}h(seed) = Tophash$

En conséquence, la route passant par le nœud A n'est pas détecté. Elle est aussi préférée, car comportant moins de sauts pour la longueur de la route.

4.3.5.3 État de l'art des solutions contre l'attaque *wormhole*

Plusieurs approches [PPP15] ont été proposées dans la littérature pour détecter l'attaque *wormhole* dans les réseaux MANET. Le tableau 4.5 synthétise ces différentes approches.

Méthodologie	**Protocole**	**Detection de l'attaque *Wormhole***	**Prevention de l'attaque *Wormhole***
Wormhole detection algorithm based on AODV [YZY13]	AODV	✓	✗
Approche antenne directionnelle [HE04]	AODV	✓	✗
ConSetLoc [NGGC12]	Protocole utilisant le nombre de sauts comme métrique	✓	✓
Packet leashes [PJ03]	Protocole utilisant le nombre de sauts comme métrique	✓	✓

Tableau 4.5 : Synthèse des quelques propositions existantes contre l'attaque *wormhole*

Dans [PJ03], les auteurs ont proposé un mécanisme basé sur le *packet leashes*. Le principe du *packet leashes* s'appuie sur l'ajout d'une information supplémentaire (temporelle ou géographique) au paquet afin de détecter la présence d'un tunnel *wormhole*. L'objectif étant de restreindre la distance maximale de parcours d'un paquet. Pour atteindre cet

objectif, les nœuds doivent être synchronisés. Le principe général se base sur le calcul de la distance parcourue par un paquet en utilisant, soit la position des nœuds pour le *Geographical Leashes*, soit les différences temporelles entre les nœuds pour le *temporal leashes*. Comme nous le verrons, ces mécanismes ne permettent pas de se prémunir contre un tunnel *wormhole*. Ceci est majoritairement dû à l'introduction du délai de traitement qui est non déterministe et non négligeable.

Le *temporal leashes* est un mécanisme basé sur le temps de transmission pour détecter l'attaque *wormhole*. Le principe consiste à faire des recherches de l'attaque *wormhole* pendant le processus de découverte de routes, et ainsi à calculer le temps de transmission entre deux nœuds successifs tout au long du chemin établi. La formule 4.1 indique la méthode de calcul effectuée par les nœuds :

$$te = ts + L/C - \delta \tag{4.1}$$

— te : indique le temps d'expiration

— ts : indique le temps d'envoi du paquet

— C : vitesse de propagation d'un signal sans fil qui est plus ou moins équivalente à la vitesse de la lumière

— L : distance maximale que peut parcourir un paquet pour une transmission entre voisins (un saut)

— δ indique la différence maximale d'horloge entre deux nœuds.

La détection est basée sur le fait que le temps de transmission entre deux nœuds reliés par un tunnel *wormhole* est considérablement plus élevé que celui entre deux nœuds légitimes. L'inconvénient de cette approche est la forte probabilité de générer des faux positives car la différence dépendra surtout des différentes variables temporelles suivantes :

— le délai de transmission qui est le temps pour mettre le paquet à envoyer sur le médium sans fil. Cette valeur dépend notamment du type de connecteur entre le système responsable de la charge utile et l'antenne. Elle dépend aussi de la taille des paquets et de la taille de la file d'attente en cas d'interférence. Le système entre le calculateur de bord et l'antenne en passant par le modem ajoute également un délai qui peut être supérieur au temps de propagation.

— le délai de traitement dans un nœud correspond au délai de traitement du message et dépend de la capacité du traitement du nœud (puissance CPU).

En conséquence, pour que l'algorithme *temporal leashes* fonctionne, la méthode de synchronisation doit tenir compte des variables non déterministes et dépendantes de la caractéristique matérielle de l'équipement utilisé.

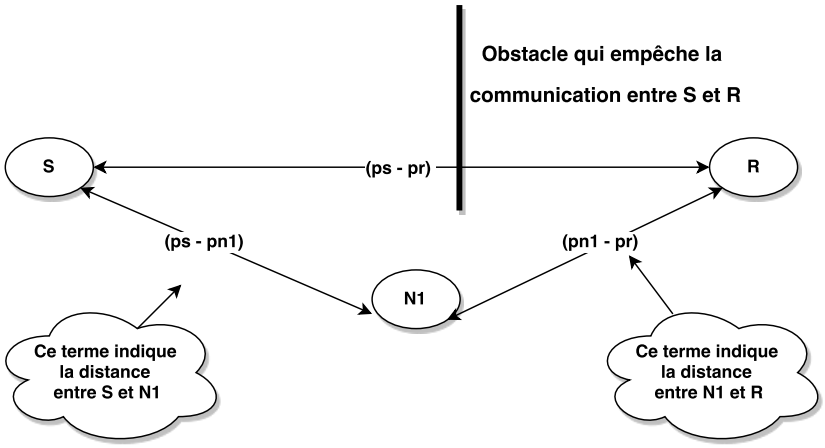

FIGURE 4.11 : Illustration de la limite d'utilisation du mécanisme de *Geographical Leashes*

Le *Geographical Leashes* consiste à ajouter la position du nœud émetteur dans la requête pour permettre à un nœud (qui reçoit un paquet) de mesurer la distance qui le sépare du nœud précédent (cf. formule 4.2). En faisant l'hypothèse que les nœuds sont synchronisés, chaque nœud est capable de déterminer si le paquet a parcouru plus de distance que prévu. La formule de base utilisée est la suivante : *Soit S le nœud source, R le nœud récepteur, ts et tr représentent respectivement le temps d'envoi et de réception du paquet et V la borne maximale de la vitesse des nœuds. La distance entre S et R doit respecter la condition :*

$$dsr \leq (ps - pr) + 2V(tr - ts + \alpha) + \delta \tag{4.2}$$

Cette expression indique la distance entre S et R. La limite de cette approche se présente lorsque le nœud destination R sort de la zone de couverture du nœud S à cause d'un obstacle entre les deux nœuds. Dans ce cas, le paquet doit passer par un autre nœud intermédiaire N1. La figure 4.11 illustre ce cas de figure.

Sur cette figure, le nœud S envoie le paquet en insérant sa position (que l'on note par *ps*). Dans le cas où il n'y a pas d'obstacle, la distance entre S et R peut être vérifiée par la condition représentée par la formule 4.2. Dans le cas contraire, si le paquet est transmis dans un premier temps vers N1 et ensuite retransmis vers R, cette condition ne sera plus respectée même en l'absence d'une attaque *wormhole*. Dans ce qui suit, nous allons démontrer la limite de ce mécanisme en comparant la distance entre les nœuds S et R avec et sans un obstacle.

Avec obstacle nous avons :

$$(ps - pr)^2 = (ps - pn1)^2 + (pn1 - pr)^2 - 2(ps - pn1)(pn1 - pr)\cos(\alpha) \tag{4.3}$$

α indique l'angle que font les segment réliant les nœuds S et R et S et N1.

Puisque :

$$(ps - pn1)^2 + (pn1 - pr)^2 > (ps - pn1)^2 + (pn1 - pr)^2 - 2(ps - pn1)(pn1 - pr)\cos(\alpha)$$

Nous avons donc :

$$(ps - pn1)^2 + (pn1 - pr)^2 > (ps - pr)^2$$

Par la suite,

$$((ps - pn1) + (pn1 - pr))^2 > (ps - pn1)^2 + (pn1 - pr)^2 > (ps - pr)^2$$

Ce qui implique que :

$$(ps - pn1) + (pn1 - pr) > (ps - pr) \tag{4.4}$$

En divisant les termes de cette équation par la vitesse de propagation, nous avons : en supposant que β et σ sont deux variables non nulles, nous avons :

$$\left\{\begin{matrix} tr - ts = \beta & \rightarrow avec \quad obstacle \\ tr - ts = \sigma & \rightarrow sans \quad obstacle \end{matrix}\right\}$$

Ce qui implique $\rightarrow \beta > \sigma$

En considérant cette expression dans 4.5, nous avons :

$$(ps - pn1) + (pn1 - pr) + \beta > (ps - pr) + \sigma \tag{4.5}$$

En ramenant 4.5 à 4.2 :

$$(ps-pn1)+(pn1-pr)+2V((tr-ts)_{obstacle}+\alpha)+\delta > ||ps-pr||+2V((tr-ts)_{sans\_ostacle}+\alpha)+\delta$$

Ainsi, nous avons :

$$dsr_{obstacle} > dsr \tag{4.6}$$

Ce qui prouve que la représentation mathématique 4.2 ne fonctionne pas en cas de présence d'un obstacle.

Par ailleurs, dans [YZY13], une autre proposition a été avancée. Il s'agit d'une modification du protocole AODV pour trouver plusieurs chemins différents entre la source et la destination. Les auteurs considèrent ensuite que le chemin comportant un ou plusieurs nœuds *wormhole* est caractérisé par un nombre de sauts significativement inférieur au nombre de

sauts des autres chemins. Néanmoins, ils ont démontré que dans le cas où la densité des nœuds dans le réseau est faible et qu'il n'est pas possible d'avoir plusieurs routes, cette proposition ne garantit pas la protection contre l'attaque *wormhole*.

Dans [HE04], les auteurs proposent une méthode basée sur l'utilisation d'une antenne directionnelle. Grâce à la capacité de cette antenne à tourner vers la direction d'arrivée d'un paquet, elle peut aider à vérifier l'identité d'un nœud situé à deux sauts ou plus. Dans ce mécanisme, durant l'échange des paquets de découverte des voisins, chaque nœud indique la direction par laquelle les messages ont été reçu. Chaque nœud n'accepte ensuite que les paquets reçus dans une direction opposée à celle déclarée. Les évaluations effectuées ont montré la pertinence de ce type de mécanisme. Elle n'est toutefois efficace qu'en cas d'utilisation d'une antenne directionnelle.

Dans [NGGC12], les auteurs ont proposé une approche intitulée *ConSetLoc* qui se base sur l'évaluation de la relation entre le nombre de sauts et la distance géographique entre les nœuds. Cette approche a été utilisée dans les réseaux de capteurs ayant des positions fixes et inchangées durant l'exécution de l'algorithme de routage. Cela a permis de considérer un intervalle statique de distance relative entre les nœuds. Les évaluations de performances effectuées ont montré l'avantage de cette approche par rapport aux approches existantes dans la littérature. Toutefois, les résultats obtenus dépendent de la position statique des nœuds. Il serait alors intéressant de pouvoir l'adapter dans un réseau plus dynamique comme le réseau UAANET.

4.3.5.4 Synthèse des travaux existants

. Le mécanisme de *packet leashes* peut détecter l'attaque *wormhole* si la connectivité entre deux nœuds n'est interrompue par un obstacle, qui les obligent à recourir à un nœud intermédiaire. L'approche ConSetLoc offre également le même service, mais le protocole n'a pas été testé sur un environnement dynamique où les nœuds ont des positions modifiées en temps réel. Nous avons donc opté pour une solution qui combine ces deux approches en se basant sur une amélioration de l'algorithme de *Geographical Leashes*. Nous cherchons ainsi à établir une correspondance entre la distance relative entre deux nœuds et le nombre de sauts inscrits dans le paquet. Dans notre approche, la distance de voyage d'un message est limitée ; chaque message a un horodatage et une localisation de la source. La destination calcule la distance relative avec sa propre localisation et son horodatage. Par la suite, cette distance relative est associée au nombre de sauts inscrit dans le paquet. Si la distance parcourue par le paquet correspond à la valeur du nombre de sauts que l'on devrait avoir, nous pouvons dire qu'il n'y a pas de tunnel *wormhole* dans le réseau.

Il est important de souligner que cette approche peut s'exécuter dans d'autres types de réseaux MANET, mais son intérêt principal dans le réseau UAANET réside dans le fait que, les nœuds du réseau peuvent être synchronisés.

4.3.6 Proposition d'un mécanisme pour détecter et contrer l'attaque *wormhole*

La première approche permettant de détecter cette attaque se base sur le principe de *Geographical Leashes*. Comme nous l'avons développé précédemment, l'approche *Geographical Leashes* initiale telle qu'elle est présentée dans [PJ03] génère des faux positifs et ne permet pas toujours de détecter l'attaque *wormhole* (notamment en cas de présence d'un obstacle entre deux nœuds habituellement voisins). Nous avons donc formalisé une autre approche pour créer un intervalle de distance ; elle consiste en l'évaluation de la relation entre la distance géographique relative entre deux nœuds voisins et le nombre de sauts dans le paquet. Nous nous basons alors sur une distance qui est non déterministe et bornée par la valeur maximale D_{max} ; D_{max} représentant la portée maximale de l'antenne omnidirectionnelle d'un nœud.

Pour cela, lors de l'envoi d'un paquet de contrôle, chaque nœud inclut sa position. À la réception le nœud calcule la distance relative séparant les deux nœuds, en utilisant la formule de distance euclidienne dans un environnement à trois dimensions (cf. 4.9).

Pour que cet algorithme puisse fonctionner, il est nécessaire de synchroniser les nœuds. Dans un réseau UAANET, les drones ont chacun un GPS leur permettant de connaitre et d'envoyer leur position à la station sol en temps réel. Ceci facilite donc la synchronisation des nœuds. Par ailleurs, les drones utilisés (drone de DT18) utilisent un processeur à 8 cœurs ayant une forte capacité de calcul, ce qui est un avantage supplémentaire pour effectuer les calculs de position.

Dans ce qui suit, nous allons analyser l'attaque *wormhole* et présenter la correspondance entre la distance géographique et le nombre de sauts qu'il est nécessaire de mettre en œuvre afin de détecter cette attaque. Pour cela, les notations illustrées sur le tableau 4.6 sont considérées dans la figure 4.12.

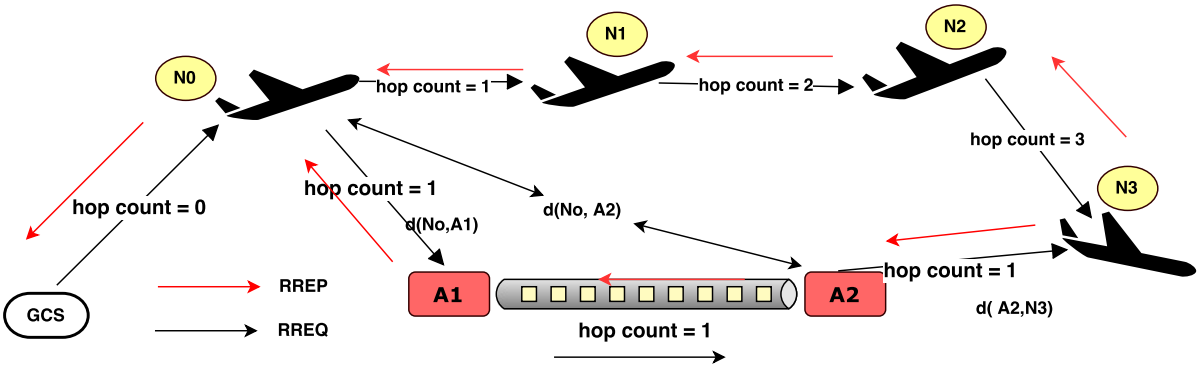

FIGURE 4.12 : Attaque de type *wormhole* avec deux attaquants

Nous pouvons dire que :

Paramètre	Description
hc	Compteur du nombre de sauts
D_{max}	Distance maximale d'un saut
R_{ij}	Distance réelle entre deux nœuds i et j
$d(No, A1)$	Distance entre N0 et A1
$d(A1, A2)$	Distance entre les deux attaquants
$d(A2, D)$	Distance entre la deuxième cible et le deuxième attaquant
$d(No, A2)$	Distance entre N0 et A2
Pi	Somme de la distance parcourue par un paquet à un instant t
T	Somme de la distance totale parcourue sur la route légitime
Dw	Longueur totale du chemin passant par le lien *wormhole*

Tableau 4.6 : Tableau de notation

— l'attaque *wormhole* génère un tunnel ayant comme caractéristique d'acheminer les paquets avec un nombre de sauts inférieur aux autres liens.

— D_{max} représente la portée radio maximale d'un nœud. Cette donnée est disponible au niveau radio puisque il est possible de paramétrer la puissance de signal d'émission et de réception de l'adaptateur sans fil utilisé.

On a : $\forall i, j \in [0, n], R_{ij} \leq D_{max}$ (*n étant le nombre maximal de nœuds dans le réseau*)

Ainsi, la connectivité entre deux nœuds légitimes voisins que l'on note **c** peut être présentée par les conditions suivantes :

$$c(i,j) = \begin{cases} 1 & si \quad R_{i,j} \leq D_{max} \\ 0 & si \quad R_{i,j} > D_{max} \end{cases} \tag{4.7}$$

L'existence d'un tunnel *wormhole* ne respecte pas cette condition, en permettant à deux nœuds distants d'une distance supérieure à D_{max} d'être voisin.

$$c(i,j) = \begin{cases} 1 & si \quad R_{i,j} \leq D_{max} \\ 1 & si \quad R_{i,j} > D_{max} \end{cases}$$

Ensuite, pour que la présence du tunnel ait un sens au point de vue de l'attaquant, nous considérons que le deuxième attaquant A2 ne se trouve pas dans la couverture de N0. En effet, si l'attaquant A2 se trouve dans la couverture de N0, cela implique que le tunnel n'a aucun sens étant donné que son objectif est de créer une connexion entre deux nœuds

séparés par une distance supérieure à Dmax. Cela se traduit par :

$$d(No, A1) \leq D_{max}$$
$$d(A2, N3) \leq D_{max}$$
$$d(No, A2) > D_{max}$$
$$d(A1, N3) > D_{max}$$

Nous avons donc :

$$d(No, A2)^2 = d(No, A1)^2 + d(A1, A2)^2 - 2d(No, A1)d(A1, A2)cos(\gamma))$$

avec γ l'angle que forme le segment tracé par (N0, A1) et (A1,A2).
Ainsi,

$$d(No, A1)^2 + d(A1, A2)^2 > Dmax^2 \rightarrow d(A1, A2)^2 > Dmax^2 - d(No, A1)^2$$

$$d(A1, A2)^2 > (D_{max} - d(No, A1))(D_{max} + d(No, A1))$$

$$d(A1, A2)^2 > (D_{max} - d(No, A1))(D_{max} - d(No, A1))$$

Ainsi,

$$d(A1, A2) > D_{max} - d(No, A1)$$

Sachant que :

$$Dw = d(A1, A2) + d(No, A1) + d(A2, N3)$$

Nous avons donc :

$$Dw > D_{max} \tag{4.8}$$

En nous appuyant sur cette inégalité de distance, nous pouvons faire une comparaison entre nombre de sauts présents dans le paquet et du nombre de sauts que l'on devrait avoir compte tenu de la distance entre les deux nœuds voisins. Nous rappelons que la position des nœuds est inclus dans le paquet, ce qui permettra à chaque drone de calculer la distance parcourue par le paquet. La distance entre deux nœuds i et j est obtenue par la formule ci-dessous :

$$Rij = \sqrt{(xj - xi)^2 + (yj - yi)^2 + (zj - zi)^2} \tag{4.9}$$

(x, y, z) représentent l'ensemble (latitude, longitude, altitude). Par exemple, xj représente

la latitude du nœud j ; xi représente la latitude du nœud i.

Dans le cas d'une attaque de type *wormhole*, nous allons donc constater une anomalie sur la distance et le nombre de sauts hc.

Sachant que :

$$T = \sum_{i=0,j=0}^{n} R_{i,j}$$

— Quand le nœud N0 envoie le paquet, nous avons $T0 = R_{01}$ qui correspond à hc = 1

  — R_{01} indique la distance entre N0 et N1

— Quand le nœud N1 envoie le paquet, nous avons $T1 = T0 + R12$ qui correspond à hc = 2

  — R_{12} indique la distance entre N1 et N2

— Quand le nœud N2 envoie le paquet, nous avons $T2 = T1 + R23$ qui correspond à hc = 3

  — R_{23} indique la distance entre N2 et N3

Pour permettre de détecter l'attaque *wormhole*, le tableau 4.7 est créé.

Valeur de T	**Nombre de sauts hc**
$0 < To \leq Dmax$	0
$D_{max} < T1 \leq 2Dmax$	1
.....	.....
$(n-1)D_{max} < T_{n-1} \leq (n+1)Dmax$	n-1

Tableau 4.7 : Tableau de correspondance entre la distance géographique T et le nombre de sauts

À partir de ce tableau, nous avons donc :

$$\frac{T}{Dmax} - 1 \leq hc < \frac{T}{Dmax} + 1 \tag{4.10}$$

Ainsi, pour détecter donc l'attaque *wormhole*, nous pouvons soit utiliser le tableau de correspondance, soit utiliser cette inégalité et voir si le nombre de sauts appartient à l'intervalle ci-dessus.

Exemple

Pour illustrer un exemple d'utilisation de ce mécanisme, la figure 4.12 est considérée.

1. Dans le cas où le nœud N3 reçoit le message de la part de N2,

   — N3 va calculer la distance R_{23} à partir de la formule 4.9

— Il calcule ensuite la valeur courante de la somme des distances T comme suit : $T2 = R_{01} + R_{12} + R_{23}$

— L'équation 4.10 devient donc :

$$\frac{R_{01} + R_{12} + R_{23}}{Dmax} \leq hc < \frac{R_{01} + R_{12} + R_{23}}{Dmax} + 1$$

— Ensuite, puisque $T2 = R_{01} + R_{12} + R_{23}$, nous avons :

$$\frac{T2}{Dmax} - 1 \leq hc - 1 < \frac{T2}{Dmax} < \frac{T2}{Dmax} + 1$$

— Ainsi, en s'appuyant sur le tableau 4.7, nous devrons donc avoir $hc = 2$ car $2D_{max} < T2 \leq 3D_{max}$

— En comparant, ce résultat à la valeur courante de hc qui à cette étape est égale à 2, nous pouvons dire que le paquet n'a pas été transféré sur un lien *wormhole*.

2. Dans le cas où le message est reçu de la part de A2 :

— N3 calcule la distance parcourue par le paquet et qui est égale à T2=Dw

— L'équation 4.10 devient donc :

$$\frac{Dw}{Dmax} - 1 \leq hc < \frac{Dw}{Dmax} + 1$$

— Sachant que $Dw > D_{max}$, nous avons : $Dw = \gamma D_{max}$ avec $\gamma > 1$. Nous avons donc : $0 < \gamma - 1 \leq hc < \gamma + 1 < 2\gamma \Rightarrow 0 < hc < \gamma + 1$

— En comparant ce résultat à la valeur de hc = 0 dans le paquet (car le paquet n'a pas été modifié par l'attaquant), nous validons que le paquet a été transmis par un ou plusieurs attaquants *wormhole*. De manière générale, tant que $Dw > D_{max}$ (ce qui est toujours vrai pour une attaque *wormhole*), l'attaque sera toujours détectée.

À travers cet exemple, nous avons montré comment notre proposition permet de détecter et de se prémunir contre l'attaque *wormhole*.

4.3.7 Proposition d'un mécanisme pour contrer l'attaque *wormhole* réalisé par un seul attaquant

Pour se protéger contre l'attaque décrite dans 4.3.5.2, le mécanisme présenté précédemment peut être utilisé. Comme expliqué précédemment, cette attaque consiste à transférer un paquet obtenu par écoute illicite vers un autre nœud voisin. Sur la figure 4.10, le nœud A transfère le paquet sans modification vers N3.

Pour cette attaque, nous ajoutons un autre mécanisme qui va s'appuyer sur le mécanisme précédent. C'est-à-dire qu'une fois que l'on a la confirmation que notre voisin se trouve à une distance inférieure à D_{max}, nous pouvons faire confiance à ce nœud en incluant son identité dans une fonction de hachage. Le principe de la détection ici est d'inclure l'identité [5] du prochain nœud dans la fonction de hachage sur chaque nœud. Les opérations suivantes sont effectuées en s'appuyant sur le tableau 4.8.

Champ	**Valeur**
Type	64
Hash function	H
Signature	La signature des champs non mutables
Hashnew	C'est la valeur de taille fixe qui est obtenue après avoir fait l'opération H[Current_node, Next_node, Previous_hash chain]. Les deux premières variables indiquent l'adresse IP du nœud source et du prochain nœud. La troisième variable est la précédente valeur de hash
Hashold	la valeur du hash précédemment

Tableau 4.8 : Illustration des différents champs du paquet requête

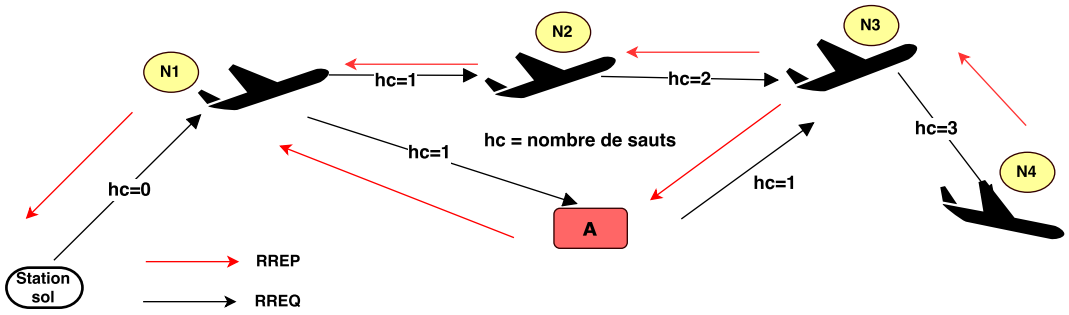

FIGURE 4.13 : Attaque (version simplifiée de l'attaque *wormhole*) avec un seul attaquant

Le nœud N1 fait l'opération suivante :

- Choix de la fonction H à utiliser
- Calcul de $Hashold = H(seed)$ avec seed une valeur aléatoire choisie par le nœud émetteur
- Calcul de $hashnew = H(N1, N2, Hashold)$, N1 étant l'adresse du nœud suivant
- Envoi du message N1 $\Rightarrow N2$: $[64, H, signature, Hashnew, Hashold]$

À la réception, le nœud N2 fait les opérations suivantes :

5. L'identité peut être l'adresse IIP ou la clé publique

— Vérification de l'intégrité en calculant $Hashverifier = H[Previous\_node, Current\_node, Hashold]$ et vérifie si la valeur correspond à $Hashnew$. En se basant sur la propriété à sens unique de la fonction de hachage, une modification du message entrainerait une différence avec Hashverifier.

— $Hashverifier = H[N1, N2, Hashold] = Hashnew$, cela atteste donc que le chemin n'a pas été retransmis par un attaquant.

— Calcul de la nouvelle valeur de $Hashold = Hashnew$

— Calcul de $Hashnew = H[N2, N3, Hashold]$

— Envoi du message N2 $\Rightarrow N3 : [64, H, signature, Hashnew, Hashold]$

Puisque l'attaquant va tout simplement transférer les paquets sans modification, l'attaque sera detecté. En effet :

1. Sur le chemin légitime, nous avons $Hashverifier = H[N2, N3, Hashold]$ qui est égal à Hashnew
2. Sur le chemin *wormhole*, nous avons $Hashverifier = H[N1, N3, Hashold] \neq Hashnew$, le nœud N3, va donc conclure qu'il y a une anomalie dans le réseau.

L'opération est répétée jusqu'au nœud destinataire et également pour le chemin de retour pour l'envoi du paquet RREP. En ce qui concerne la valeur exacte du nombre de sauts, nous pouvons le déduire par le nombre de fois où le hachage a été utilisé pour vérification. Nous pouvons également l'inclure dans le calcul de la chaîne de hachage.

Il est à préciser que ce mécanisme peut être utilisé pour détecter une attaque *wormhole* formée par deux attaquants. La détection est possible tant que deux nœuds qui ne sont pas voisins sont considérés comme voisin par le tunnel *wormhole*. Toutefois, il ne peut à lui tout seul détecter l'attaque *wormhole*, car il est nécessaire de s'assurer de l'absence d'un nœud *wormhole* parmi les nœuds voisins. Ceci ne peut être atteint qu'avec la première approche décrite dans la sous-section 4.3.6.

4.3.8 Limites du protocole SUAP

Le protocole SUAP se base sur le protocole SAODV pour les mécanismes qui sont en lien direct avec l'authentification des messages. Pour les champs non mutables, l'utilisation d'une signature numérique est efficace contre tout type d'attaque de modification ou l'entrée de faux paquets dans le réseau. Pour les champs non mutables, le principe de chaine de hachage permet également d'assurer l'intégrité du nombre de sauts à chaque retransmission. Or, ce dernier principe est connu pour être vulnérable contre l'attaque *wormhole*. Pour cela, nous avons proposé un mécanisme qui se base sur l'utilisation de la fonction de hachage dans SAODV.

Toutefois, l'utilisation du principe de fonction de hachage se heurte à un obstacle qui est la non-considération du principe de Kerchoffs. En effet, dans SAODV, un tableau de fonction de hachage est créé et qui n'est connu que par les nœuds légitimes, leur permettant ainsi d'identifier la fonction qui a été choisie par le nœud source. Cette façon de faire peut être considérée comme une sécurité par obfuscation, car la fonction n'est initialement connue que par les nœuds légitimes.

Dans le cas où la fonction est connue par l'attaquant, quelques cas d'attaques peuvent être exécutés à l'encontre du protocole SUAP.

— l'attaquant peut utiliser la fonction de hachage pour incrémenter le nombre de sauts dans le but d'éviter que le paquet ne soit transmis sur un chemin optimum. Par exemple, pour dégrader la performance du réseau, l'attaquant peut forcer à augmenter le nombre de sauts sur un chemin légitime pour que le paquet soit transmis sur une route non optimale.

— un ou plusieurs attaquants *wormhole* peuvent également modifier la valeur des deux empreintes Hashnew et Hashold pour éviter d'être détectés. Cette forme d'attaque plus évoluée consiste non seulement à créer un tunnel, mais aussi modifier le paquet échangé à l'intérieur de ce tunnel. Ce type d'attaque n'a pas été considérée dans cette thèse, car nous n'avons considéré qu'un tunnel *wormhole* qui ne modifie pas le paquet tel qu'il est défini dans [YZY13]. Dans le cas d'une attaque par modification, si l'attaquant arrive à déduire la valeur de Hashnew à partir de Hashold en prenant en compte l'identité des nœuds légitimes, les paquets passant par un tunnel *wormhole* peuvent ne pas être détectés.

Toutefois, cette attaque de modification de la valeur d'empreinte à des effets limités dans un contexte temps réel et également si une fonction de hachage plus robuste est utilisée. En effet, dans le cas où une fonction de hachage résultant d'une composition de fonctions (comme étudiée dans [NS11]) est utilisé, la fonction peut être rendue publique puisque le délai que met l'attaquant pour calculer la fonction peut être suffisamment important pour ne pas impacter l'intégrité des messages de routage (qui sont exécutés en temps réel durant la mission).

Ce cas spécifique permet donc de conclure que le protocole SUAP peut garantir la détection de l'attaque *wormhole* dans un contexte UAANET. Toutefois, dans le cas où le protocole SUAP est utilisé dans d'autres familles des réseaux MANET (par exemple, les réseaux de capteurs), il peut ne pas suffire pour protéger contre l'attaque *wormhole*. Il serait alors intéressant d'utiliser une autre approche de détection. Par exemple, dans un réseau MANET ayant une faible densité de nœuds [6], il peut être envisageable d'utiliser le

6. En cas de forte densité de nœuds, la solution de relation entre nombre de sauts et distance relative peut dégrader la performance du réseau, car le temps et l'*overhead* associé aux mécanismes de signature des positions à chaque retransmission sont non négligeables

mécanisme de relation entre nombres de sauts et distance relative que l'on a presenté dans la section 4.3.2 pour tous les paquets de routage.

4.3.9 Différents formats des paquets de contrôle dans SUAP

Nous avons vu précédemment deux mécanismes permettant de contrer l'attaque *wormhole* et l'attaque avec un seul attaquant agissant comme un tunnel *wormhole*. Le protocole SUAP met donc en œuvre ces deux mécanismes en utilisant la technique de correspondance entre le nombre de sauts et la distance relative pour les paquets de maintenance de route et la méthode de fonction de hachage imbriquée pour les paquets de découverte de route. La première approche (technique de correspondance entre distance relative et nombre de sauts) permet de s'assurer que le paquet que nous avons reçu ne vient pas d'un tunnel *wormhole* ou d'un nœud malveillant (attaque avec un seul attaquant). Il permet donc de protéger les différents trafics de signalisation contre l'attaque *wormhole*. Toutefois, dans le cas où nous avons une densité élevée des nœuds, cette implémentation peut être lourde, car chaque nœud du réseau doit être synchronisé (pour le calcul de distance relative). Également, puisque les positions sont signées par chaque nœud, la signature sera donc vérifiée à chaque retransmission. Cela peut dégrader la performance du réseau suivant la densité de nœuds. Ainsi, nous avons décidé d'utiliser la première approche que pour la découverte des voisins. Elle ensuite couplé avec la deuxième approche (celle avec l'imbrication des valeurs de hash) qui elle n'exige aucune synchronisation des nœuds entre la source et la destination. Notre protocole SUAP permet donc dans un premier temps de protéger la phase de découverte des voisins pour s'assurer de l'identité de voisin, et dans un deuxième temps, de garantir la mise en place d'une route sécurisée par des mécanismes d'authentification et d'intégrité.

Par ailleurs, il est important de noter que pour notre implémentation, nous avons utilisé l'algorithme préconisé par SAODV qui est le RSA-512 (l'algorithme RSA-1024 est également préconisé). La taille des clés publiques pour chaque paquet est donc de 512 bits. Toutefois, suivant le niveau de sécurité que l'on souhaite atteindre, il est possible d'utiliser les autres variantes à savoir du RSA-2048 ou du RSA-4096.

À l'initialisation, chaque nœud du réseau découvre son voisin en émettant un paquet Hello. Dans ce paquet Hello, chaque nœud inclut sa position pour permettre a son voisin de chercher la distance rélative. Ensuite, en cas de recherche de route, le nœud émetteur envoie une requête directement à ses voisins identifiés durant la phase de découverte des voisins en imbriquant son identité dans la chaîne de hachage (cf. section 4.3.7).

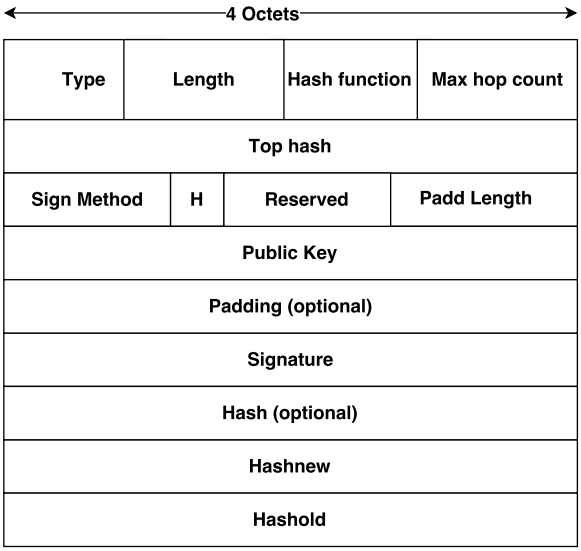

FIGURE 4.14 : Format de l'extension Secure RREQ

4.3.9.1 Format des messages *SRREQ*

Le format de l'extension du paquet requête (qui est est illustré par la figure 4.14) et expliqué ci-dessous. Il est important de préciser que dans ce format de paquet, le champ Hash est optionnel. Ce champ est utilisé dans le protocole SAODV pour vérifier l'intégrité du nombre de sauts. Dans le protocole SUAP, ce champ peut-être maintenu ou ignoré. Dans le cas où il est ignoré, le nombre de fois où la fonction de hachage a été utilisée par les nœuds légitimes peut être utilisé comme indicateur du nombre de sauts.

— Type : 64.

— Length : la taille en octets du paquet en n'incluant pas le champ *Type* et *Length*.

— Hash function : la fonction de hachage utilisé pour calculer la valeur de Hash, Hashnew, Hashold et *Top hash*. Ce champ est codée sur 8 bits.

— Max Hop Count : le nombre de sauts maximal qui est égal à 64 par défaut (*network lifetime*). Ce champ est codée sur 8 bits.

— Public key : la taille de la clé publique est de 512 bits dans notre cas d'étude.

— Top hash : le champ qui permet d'authentifier l'intégrité du nombre de sauts. Ce champ à une taille variable. Dans notre implémentation, elle est codée sur 32 bits.

— Sign Method : la méthode de signature utilisée. Elle est codée sur 8 bits.

— Padding Length : la variable qui spécifie la taille de bits de rembourrage ajouté au paquet. Si elle est mise à zéro, cela indique qu'il n'y aura pas de rembourrage.

— Signature : la signature de tous les champs non mutables.

— Hash : la valeur de l'empreinte qui correspond à la valeur courante du nombre de sauts. Ce champ à une taille variable. Dans notre implémentation, elle est codée sur 32 octets.

— Hashold : la valeur de l'empreinte qui correspond au calcul de l'ancienne valeur des données mutables. Ce champ à une taille variable. Dans notre implémentation, elle est codée sur 32 octets.

— Hashnew : la valeur de l'empreinte qui est calculée à partir de l'ancienne valeur de hash. Ce champ à une taille variable. Dans notre implémentation, elle est codée sur 32 octets.

Il est à noter que le format de l'extension du paquet SRREP pour le paquet réponse est similaire à ce format de paquet.

4.3.9.2 Format des messages *Secure Hello*

Le message Hello est très important dans l'architecture de l'algorithme SUAP. Ce paquet est envoyé en broadcast vers les voisins à un saut pour maintenir la connexion locale à jour. C'est-à-dire que par l'intermédiaire de ce paquet, un nœud connait qui est son voisin à un saut. Dans le RFC d'AODV, ce paquet n'est envoyé qu'après l'envoi de la première requête. Dans la version de notre protocole, il est envoyé en première instance dès que le protocole s'exécute. Un nœud du réseau envoie donc périodiquement un paquet Hello en y incluant sa position (longitude, latitude, altitude). Le nœud voisin recevant ces informations calcule directement la distance relative qui le sépare de l'émetteur pour savoir si le paquet n'a pas emprunté un tunnel *wormhole*. En appliquant l'algorithme 4.10, nous pouvons connaitre l'existence d'un lien *wormhole* ou pas dans le réseau. Si un tel lien existe, il convient alors de supprimer le paquet et de créer une alerte dans le réseau. Dans le cas contraire, nous faisons confiance à ce nœud.

Le format de l'extension de ce paquet est montré par la figure 4.15. Dans cette version de protocole SUAP, nous avons codé les positions sur 4 octets. Il est également important de préciser que la taille de la clé publique utilisée est de 512 bits.

4.3.9.3 Format des messages *SRERR*

Le paquet SRERR est généré dans un de 2 cas suivant :

— Quand un nœud reçoit un paquet de données destiné à un nœud dont il ne connait pas dans sa table de routage

— En cas de perte de lien sur une route active.

Supposons qu'il y a un nœud X qui s'est déconnecté du réseau. Le premier nœud qui remarque cette déconnexion informe les autres nœuds en propageant l'identité (qui est

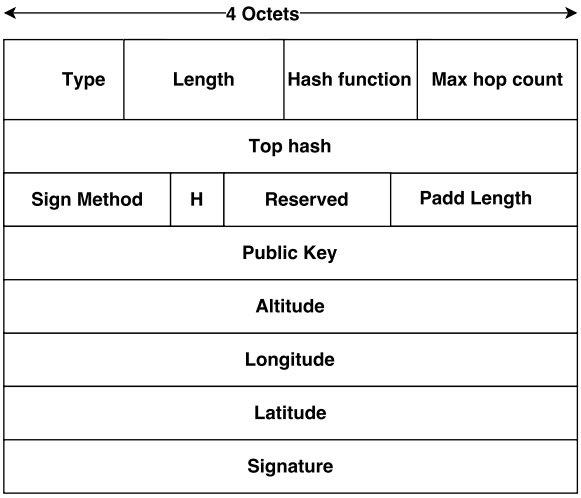

FIGURE 4.15 : Format de l'extension du paquet *Secure Hello*

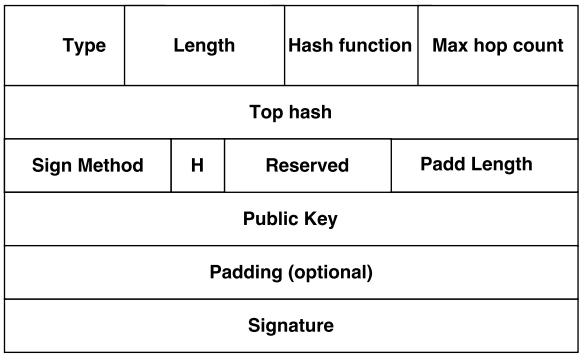

FIGURE 4.16 : Format de l'extension *Secure Error*

généralement son adresse IP) du nœud X vers tous les nœuds qui dépendent de ce nœud. L'entrée correspondante à ce nœud X est par la suite effacée de la table de routage locale de chaque nœud intermédiaire. Ce paquet d'erreur peut être envoyé à un ou plusieurs nœuds en fonction du nombre de nœuds à avertir de la rupture de liaison détectée.

Dans le protocole SUAP, nous garantissons l'authentification de ce paquet, car un attaquant peut générer de faux paquets d'erreurs pour isoler un nœud légitime du réseau. Pour éviter cela, nous procédons à la signature numérique de tous les champs comme dans les paquets de découverte de route. Le format de paquet erreur est montré par la figure 4.16.

4.3.10 Analyse de sécurité du protocole SUAP

Dans cette section, nous avons présenté le protocole SUAP, un protocole de routage réactif sécurisé qui garantit l'authentification et l'intégrité des messages. Ces services sont respectivement assurés par une signature numérique et une fonction de hachage. Par ailleurs,

SUAP permet également de contrer l'attaque *wormhole* qui consiste à créer un tunnel entre deux attaquants. L'approche qui a été avancée se base sur le principe de geographical leashes dans lequel la distance de voyage d'un paquet est limitée. Nous cherchons également la correspondance entre la distance relative des deux nœuds voisins et le nombre des sauts. Le protocole SUAP permet également de contrer l'attaque avec un seul nœud malveillant ne modifiant pas le paquet. Notre proposition complète donc les failles de sécurité identifiées dans la version initiale de SAODV et permet de détecter les attaques ayant pour objectif de modifier la topologie du réseau, de divulguer l'information de routage et de dégrader la performance du réseau. La figure 4.17 synthétise les mécanismes du protocole SUAP.

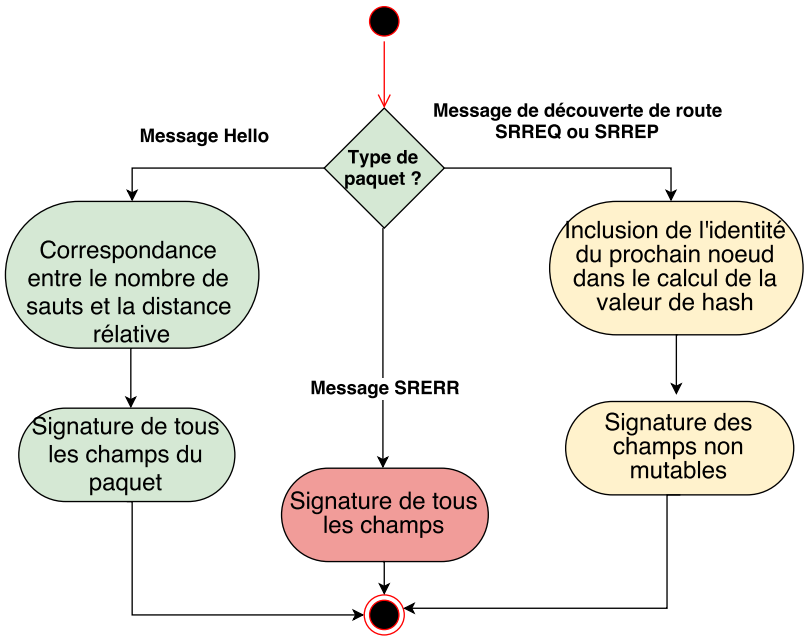

FIGURE 4.17 : Diagramme des mécanismes du protocole SUAP

4.4 Vérification formelle des propriétés de sécurité du protocole SUAP

Dans cette partie du manuscrit, nous allons décrire comment nous avons utilisé AVISPA qui est un outil de spécification et de vérification formelle pour révéler les vulnérabilités qui seraient difficiles à corriger dans l'implémentation. Cet outil est composé d'un outil graphique qui rend possible l'animation des spécifications du protocole SUAP et ainsi faire varier les scénarios d'exécution pour trouver les failles de sécurité qu'il peut comporter.

4.4.1 Nécessité des procédures de vérification

Le bon déroulement de la communication entre les drones de la flotte repose sur la formalisation des méthodes de routage et de sécurité que constitue le protocole SUAP. Celui-ci décrit l'algorithme selon lequel les messages peuvent être envoyés et traités par les nœuds du réseau ainsi que le format des messages échangés. Plusieurs outils de vérification ont été conçus dans cet objectif. L'enjeu est de rendre automatique la preuve des propriétés, et de montrer les traces d'attaques lorsqu'une propriété possède une faille de sécurité. Les différents avantages que l'on peut attendre de l'utilisation des outils de vérification formelle automatique sont cités ci-dessous. Elle est également représentée par la figure 4.18.

1. Retrouver des vulnérabilités connues : l'utilisation d'outil de vérification formelle permet d'éviter la réapparition d'ancienne vulnérabilité. Toutefois, la pertinence de les retrouver peut être questionnée puisque nous savons déjà qu'elles existent. La réponse à cette question est d'une part pour augmenter la confiance que l'on met dans l'outil de vérification (AVISPA dans le cas présent). D'autre part, il y a encore aujourd'hui des protocoles souffrant de vulnérabilités pourtant bien connues au moment de leur conception. Par exemple, l'attaque ping of death [HH99] qui est connue contre le protocole ICMP est encore visible dans certains téléphones IP. Seule l'utilisation systématique d'outils de vérification peut empêcher ce type de réapparition.

2. Trouver des variantes d'attaques : les attaques retrouvées automatiquement n'apparaissent pas toujours sous leurs formes les plus connues. Cela peut amener à réfléchir à évaluer les conséquences de l'attaque.

3. Vérifier l'utilité des contre-mesures : dans notre cas, le protocole SUAP inclut plusieurs contre-mesures (utilisation de signature numérique et de fonction de hachage à sens unique) pour sécuriser la communication. Ces mécanismes de sécurité rendent le protocole SUAP complexe et peuvent eux-mêmes ajouter des vulnérabilités. Il est donc judicieux de pouvoir vérifier leur utilité.

4. Vérifier des cas non prévus : les concepteurs font très souvent des hypothèses sur l'environnement d'exécution du protocole et sur les capacités de l'attaquant. Or, les protocoles peuvent être utilisés dans des contextes d'utilisation imprévus et s'exposent donc à des classes d'attaques plus larges que prévu. Ainsi, la vérification formelle automatique peut permettre de trouver des attaques peu réalistes, mais possibles en théorie. Cela va permettre ensuite d'accepter la faille et de le consigner dans l'hypothèse de départ sur les conditions d'utilisation du protocole.

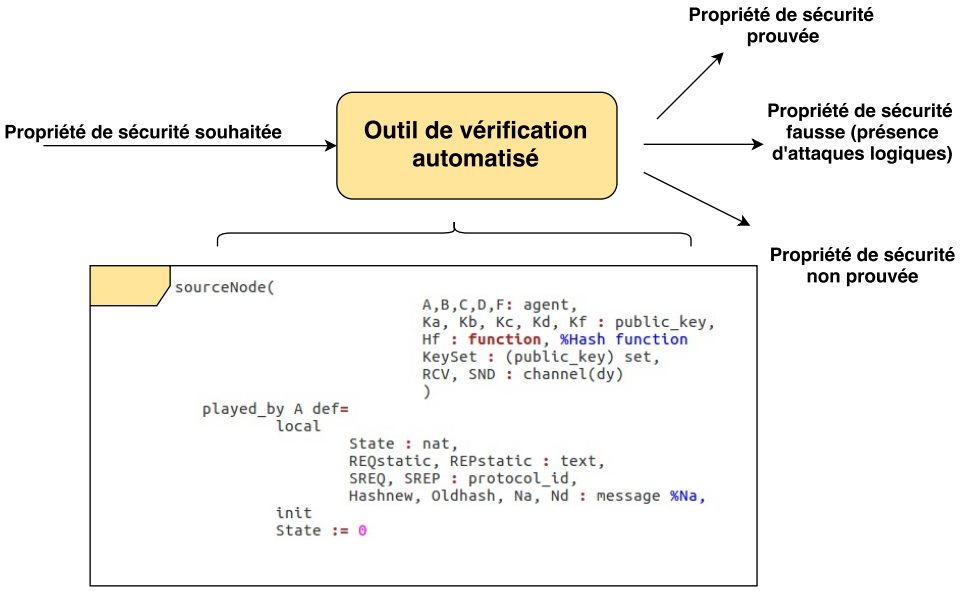

FIGURE 4.18 : Procédure de vérification automatisée

4.4.2 Choix de l'outil AVISPA

Une comparaison d'outils de vérification formelle existant est montré dans [PBP+10]. Nous avons décidé de synthétiser ces outils dans l'annexe B.1.1.Nous avons décidé d'utiliser l'outil AVISPA pour les raisons suivantes :

— Langage d'entrée : l'outil AVISPA possède un langage de spécification HLPSL (High Level Protocol Specification Language) qui lui est propre. Ce langage est simple et proche du formalisme Alice et Bob que nous utilisons pour décrire les protocoles.

— Compréhension de la sortie : l'outil AVISPA possède une présentation des résultats sous forme de graphe et de trace d'attaque compréhensible. Cela nous permet de vérifier aisément la propriété du protocole SUAP.

— Propriétés : AVISPA permet de vérifier les propriétés qui nous intéressent à savoir l'authentification des messages.

— AVISPA est également compatible avec l'outil graphique SPAN qui permet d'avoir une représentation graphique des traces d'attaques. Le rôle de SPAN (Security Animator for AVISPA) est d'aider à la conception des spécifications formelles. SPAN permet d'animer les spécifications HLPSL, c'est-à-dire de produire interactivement des MSC (Message Sequence Charts) qui peuvent être vus comme une trace « Alice & Bob » d'une spécification HLPSL.

— Expressivité : AVISPA n'est pas un outil expressif, il est donc facile à utiliser et est ergonomique. L'expressivité consiste à représenter dans la vérification les différents

types de variables d'exécution par exemple, le canal de communication, l'indéterminisme (c'est-à-dire la notion de choix ou de hasard). Notre objectif n'étant pas de créer un outil de vérification ou d'apporter des modules d'améliorations des outils existants, il était plus judicieux de choisir celui qui facilite le travail de vérification et qui serait le plus rapide à être utilisé.

— Disponibilité : AVISPA est gratuit et est disponible publiquement.

4.4.3 Utilisation de l'outil AVISPA

La vérification formelle du protocole SUAP à l'aide de l'outil AVISPA est divisée en deux parties.

1. D'une part, nous procédons à la spécification du protocole en utilisant un langage de spécification haut niveau. C'est-à-dire que l'on exprime en langage mathématique, les échanges de messages effectués entre les différents intervenants (ou agents) du protocole. Pour cela, le langage HLPSL est utilisé. Ce langage offre la possibilité de créer une abstraction du protocole et permet de spécifier en détail les différents aspects sécurité du protocole (par exemple, les rôles des nœuds, les opérateurs cryptographiques, les modèles d'intrus, etc.) Par la suite, cette spécification est traduite automatiquement dans un format intermédiaire IF grâce à un traducteur (HLPSL2IF).
2. La deuxième étape consiste à vérifier la spécification du protocole SUAP avec les différents *back-ends*. AVISPA utilise 4 modules différents (les *back-ends*) qui prennent en entrée le format IF, et qui offrent la possibilité de faire 4 analyses différentes du même protocole. Ces *back-ends* (expliqué en annexe B.1.2) intégrés à AVISPA permettent de vérifier les propriétés de sécurité, selon les besoins.

En outre, il est important de remarquer qu'il est nécessaire de considérer l'intrus dans le processus de vérification, c'est-à-dire qu'il faut spécifier son comportement. Les hypothèses communément utilisées se basent sur le modèle de Dolev-yao [Cer01]. Ce modèle suppose que le chiffrement est parfait et que l'intrus est le réseau. En effet, le chiffrement parfait permet de restreindre les capacités de déchiffrement de l'intrus. Nous considérons qu'il est incapable de trouver l'information sans la clé de déchiffrement. La seconde hypothèse concerne les capacités de l'intrus. En spécifiant que l'intrus est le réseau revient à dire que les agents lui envoient leurs messages. Il peut par la suite, les transmettre ou non vers le destinataire. L'intrus peut également envoyer à n'importe quel nœud un message formé à partir de la combinaison de ses connaissances et le chiffrer avec les clés qu'il aura obtenues.

Dans ce qui suit, nous allons voir la spécification que l'on a réalisée pour l'étude formelle de sécurité du protocole SUAP.

4.4.4 Cas d'application du protocole SUAP

Pour vérifier le protocole SUAP, nous avons décidé de spécifier le comportement possible du protocole. Sa spécification a pour objectif de vérifier si le protocole garantit l'authentification des messages comme prévu. La spécification concernant les mécanismes contre l'attaque *wormhole* n'a pas pu être réalisée, car l'outil AVISPA ne permet la comparaison des entiers qui est nécessaire pour pouvoir prendre en compte la rélation entre le nombre de sauts et la distance rélative. Cette propriété sera vérifiée pendant l'étude de validation du protocole qui sera détaillé dans le chapitre suivant.

Par ailleurs, pour spécifier le protocole SUAP, il convient de décrire quelque règles de caractérisation.

Le comportement des nœuds de communication est divisé en rôles. Tous les nœuds du réseau UAANET exécutant les mêmes actions partagent le même rôle. Dans notre cas, nous avons 3 rôles. Le nœud source, les nœuds voisins qui transfèrent le paquet et le nœud destination. Chaque agent (le terme que l'on utilise dans AVISPA désignant un nœud) est associé à un rôle. Dans un rôle, nous décrivons également l'état initial, le changement d'état et les informations à changer à chaque changement d'état. La spécification du nœud source est montrée par la figure 4.20.

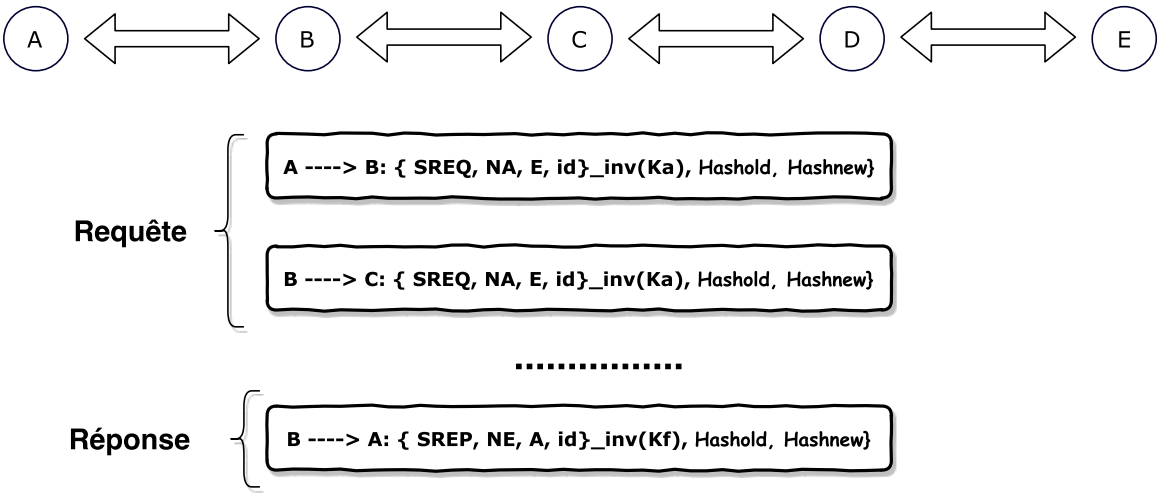

FIGURE 4.19 : Exemple d'échange dans le protocole SUAP avec 5 nœuds

À travers ces représentations, voici les propriétés du protocole que l'on souhaite analyser :

— l'authentification du champ NA, Tophash et de l'identificateur du nœud E pendant la phase de découverte. En particulier le nœud E doit pouvoir vérifier l'authenticité du couple (NA, E) ;

— l'intégrité du champ hashnew. Chaque nœud recevant le paquet calcul la variable *Hashverifier* expliquée précédemment et le compare à la valeur courante de *hashnew* ;

```
%*****************************************Source*****************************************************%
role sourceNode(
                                A,B,C,D,E: agent,
                                Ka, Kb, Kc, Kd, Ke : public_key,
                                Hf : function, %Hash function
                                KeySet : (public_key) set,
                                RCV, SND : channel(dy)
                                )
        played_by A def=
                local
                        State : nat,
                        REQstatic, REPstatic : text,
                        SREQ, SREP : protocol_id,
                        Hashnew, Hashold, Na, Nd : message
                init
                State := 0


                transition
                        step1.
                                State = 0  /\ RCV(start) =|>
                                State':= 8 /\ Na' := new()
                                                /\ Hashnew' := new()
                                                /\ Hashnew' := Hf(A,B,Hashold)
                                                /\ SND(SREQ.REQstatic.Na'.Hashold.Hashnew'.{Na'.SREQ.REQstatic}_inv(Ka))
                                                /\ witness(A,B,Na,Na')

                        step2.
                                State = 8 /\ RCV(SREP.REPstatic.Nd'.Hashold.Hashnew'.{Nd'.SREP.REPstatic'}_inv(Kb).H) /\ in(Kb, KeySet)/\
Hashnew'=Hf(A,B,Hashold)=|>
                                State':=9 /\ wrequest(A,B,Na,Na')

end role

%*****************************************R1:intermediate_node1*****************************************************%GGG
```

FIGURE 4.20 : Spécification HLPLS du nœud source

— l'authentification du champ ND et l'identificateur du nœud A durant la phase de découverte (paquet de réponses).

Comme nous pouvons le voir, nous vérifions à la fois l'authentification de bout en bout des champs statiques et l'intégrité des champs mutables. Notre spécification HLPLS permet de décrire ceci en ordonnant au nœud A d'initialiser la session et d'envoyer les messages contenant les identificateurs des paquets, et les valeurs d'empreinte. Ce message est ensuite transmis aux nœuds B, C et D jusqu'à ce que le nœud connaissant le destinataire répond avec un message de réponse. Pour les nœuds intermédiaires, quand l'un d'eux reçoit un paquet, il vérifie le certificat et la signature du nœud précédent. Quand le paquet arrive aux destinations, le couple des variables (NA, E) est vérifié. Si la vérification est réussie, le nœud destinataire génère un paquet de réponse en y incluant les identificateurs de paquet, l'adresse IP du nœud source, le certificat, et tout le message signé. Toute comme le paquet requête, ce paquet est également muni du couple des variables (NE, A) qui sera vérifié par les nœuds intermédiaires.

4.4.4.1 Analyse de spécification du protocole SUAP

Le résultat illustré dans le tableau 4.9 indique que notre protocole satisfait bien les services de sécurité attendus à savoir l'authentification de bout en bout et l'intégrité des messages (authentification saut par saut). Toutefois, comme l'illustre la figure B.2, nous pouvons voir qu'un intrus se trouvant dans la portée d'un nœud valide la possibilité d'obtenir le

message et de le retransmettre sans modification. Cette attaque est celle dont a parlé dans 4.3.5.2. Cela prouve d'une part que notre spécification est correcte et d'autre part indique que cette attaque peut effectivement rompre l'intégrité du réseau. Notre proposition présentée dans 4.3.6 peut contrer cette attaque comme le verrons dans le chapitre suivant. Nous présenterons également les résultats d'évaluation obtenus en simulation et par expérimentation réelle par rapport à cette attaque.

Sortie OFMC	**Sortie ATse**	**Sortie SATMC**
%% Number of warnings : 9 % OFMC SUMMARY **SAFE** DETAILS BOUNDED_NUMBER PROTOCOL /home/span/old.if GOAL as_specified BACKEND OFMC COMMENTS STATISTICS parseTime : 0.00s searchTime : 0.04s visitedNodes : 5 nodes depth : 4 plies	%% Number of warnings : 9 %ATSE SUMMARY **SAFE** DETAILS BOUNDED_NUMBER PROTOCOL /home/span/old.if GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 3 states Reachable : 1 states	%% Number of warnings : 9 SUMMARY **SAFE** DETAILS STRONGLY_TYPED_MODEL BOUNDED_NUMBER PROTOCOL /home/span/old.if GOAL %check the HLPSL Specification BACKEND SATMC STATISTICS fixedpointReached 5 steps stepsNumber 5 steps ATTACK TRACE %% no attacks have been found

Tableau 4.9 : Résultats obtenus par l'utilisation des différents *back-ends* AVISPA sur le protocole SUAP

1. $A \rightarrow I : \{SREQ, E, Na'\}\, inv(Ka), Hash, certA$
2. $A \rightarrow B : \{SREQ, E, Na'\}\, inv(Ka), Hash, certA$
3. $I \rightarrow B : \{SREQ, E, Na'\}\, inv(Ka), Hash, certA$
4. $B \rightarrow I : \{SREQ, E, Na'\}\, inv(Ka), Hash, certA, certB$

Dans ce cas de figure, l'intrus **I** agit comme un nœud légitime en relayant la requête vers le nœud B. Cette requête est par la suite transférée par le nœud B vers le nœud suivant dans la table de routage. Le nœud B n'a aucun moyen de savoir que le nœud I n'est pas un nœud valide, car ce dernier ne modifie pas le paquet.

Bien qu'il est impossible de savoir exactement la raison de cette vulnérabilité avec l'outil AVISPA, nous pouvons déduire qu'il est causé par une attaque de type écoute illicite effectuée par l'intrus.

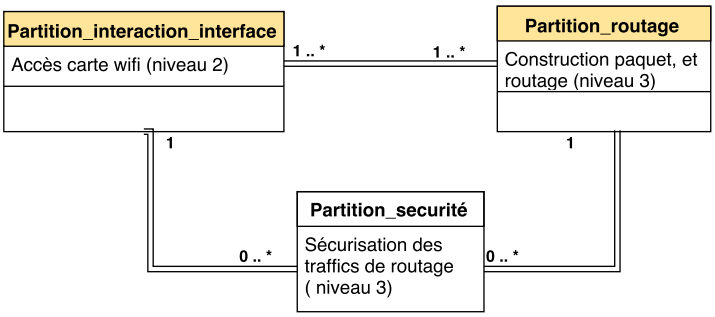

FIGURE 4.21 : Diagramme de classe haut niveau représentant le protocole SUAP

4.5 Mise en œuvre du protocole SUAP

Le contexte de l'élaboration du protocole SUAP a été présenté au chapitre 3. Nous avons analysé la méthodologie de la mise en œuvre du protocole SUAP. Dans la section précédente, nous avons présenté les fonctionnalités attendues du protocole. Dans cette section, nous montrerons les aspects techniques, les choix de mise en œuvre et les techniques utilisées pour concevoir le protocole SUAP. Dans un premier temps, nous détaillerons l'architecture du logiciel du protocole SUAP et présenter une vue globale de sa décomposition modulaire.

4.5.1 Architecture de l'algorithme SUAP

Le protocole SUAP doit assurer plusieurs fonctions développées dans la section 4.3.2. Il doit avant tout traiter et router les paquets de contrôle échangés entre les nœuds. Ensuite, il doit pouvoir assurer la sécurité des communications en garantissant l'authentification et l'intégrité des messages. L'assurance de ses services de sécurité ne devrait pas porter atteinte au fonctionnement du réseau en gaspillant de ressources réseau. Ensuite, les exigences non fonctionnelles telles que la sûreté et la sécurité fonctionnelle du système doivent être respectées par l'utilisation de la méthodologie développée dans le chapitre 3.

La figure 4.21 présente l'architecture du protocole SUAP. Elle formalise la décomposition des fonctionnalités en trois classes. Cela nous permet de nous concentrer pour chaque classe et notamment sur le sous-ensemble de fonctions à modéliser. L'abstraction des sous-ensembles liés aux autres classes a permis de simplifier la modélisation.

La classe de partition de routage est liée aux fonctionnalités de routage pour le protocole au niveau réseau. Ici, nous nous intéressons exclusivement au protocole IPv4. Cette classe gère également le filtrage des paquets à la réception pour choisir le traitement correspondant aux paquets de découverte et de maintenance de route.

La classe de partition d'interfaçage permet de lier les classes gérant les paquets de routage

aux entrées et sorties matérielles. Cette association consiste à recevoir les trames reçues par les cartes réseau, à décomposer les données de ces trames et à les transférer à la classe correspondante. Il est à noter que cette liaison est bidirectionnelle, car la partition d'interfaçage gère également l'émission des paquets IP sur le réseau sans fil.

Les mécanismes concernant la sécurisation des paquets de routage sont mis en œuvre par la classe de partition de sécurisation.

Le chapitre précédent a présenté la chaîne d'outils utilisée pour la méthodologie orientée modèle. Les deux outils principaux que l'on va évoquer sont les outils Simulink et Stateflow, qui servent à modéliser et mettre en œuvre les différents mécanismes de SUAP, à l'aide de diagrammes de blocs et de diagrammes à états de transitions. Nous avons donc associé à chaque classe de partition montrée par la figure 4.21, un modèle Simulink encapsulant des blocs et des diagrammes Stateflow. La conception de ce modèle est détaillée à la section suivante.

4.5.2 Modélisation du protocole SUAP

4.5.2.1 Partition de routage

La première classe de partition que nous allons présenter est la classe de routage qui est dédiée à la gestion des paquets de routage.

Pour pouvoir le conceptualiser, nous nous sommes basés sur une architecture générique de protocole de routage qui est représentée par la figure 4.22. Cette architecture générique est composée de plusieurs composants : le bloc *Demux*, le bloc *Maintenance block*, le bloc *Route discovery block*, et le bloc d'envoi de paquets. Pour comprendre leur fonctionnement, nous allons parcourir le rôle de chaque bloc fonctionnel.

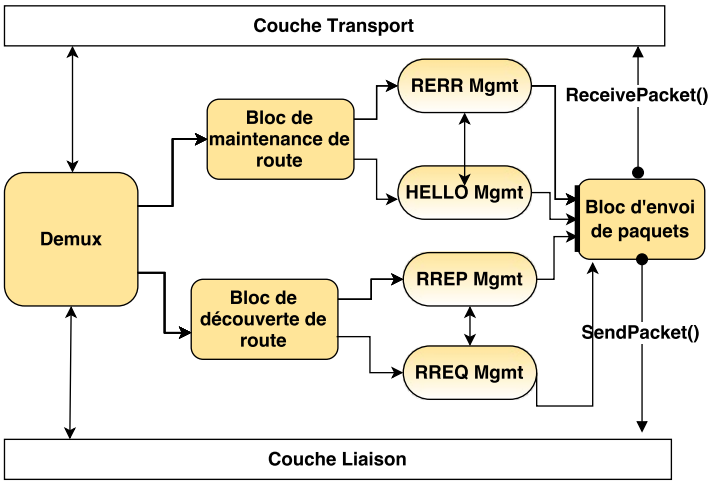

FIGURE 4.22 : Conception de la classe de partition de routage

— Le bloc Demux : réceptionne les paquets transmis par la couche liaison (pour les paquets externes qui arrivent dans le système) ou par la couche de transport (pour les paquets qui ont fini d'être encapsulés pour être envoyés sur le réseau). Il assure un étiquetage interne des paquets en se basant sur le champ *type* du paquet. Ces paquets sont ensuite envoyés vers les blocs adaptés. Par exemple, suite à la réception d'un paquet de la couche transport, le bloc Demux passe la main a un sous bloc *PacketForwarder* qui consulte la table de routage pour vérifier l'existence d'une route fonctionnelle. Dans le cas où aucune route n'existe, le système passe la main au bloc *Route discovery bloc* qui va générer un paquet requête et l'envoie vers le bloc Packet_handler pour être envoyé vers la partition d'interfaçage. La mise en œuvre de ce bloc est réalisée par la machine à états de la figure 4.25. Celle-ci est connectée à 3 interfaces réseau de réception des paquets et à une seule sortie connectée au traitement suivant pour les paquets. Le nombre d'interfaces ici est arbitraire et rien n'interdit de l'augmenter pour des besoins futurs.

— Les blocs *Route discovery* et *Route maintenance* doivent déterminer si le paquet est une requête, réponse, erreur ou un message Hello. Ces 4 formats de message sont ensuite assignés dans 4 blocs indépendants, mais qui héritent soit de *Route discovery* ou *Route maintenance*. À la sortie de ces 4 blocs, le paquet est vérifié s'il possède suffisamment d'information pour être envoyé au bloc suivant qui va l'acheminer vers la couche transport ou la couche liaison.

— Enfin, le bloc d'envoi de paquets est utilisé pour mettre à jour les champs du paquet local en fonction de son état. C'est-à-dire que dans le cas d'un paquet destiné à être traité localement, nous vérifions l'adresse IP de destination, et il est transmis à la couche transport. Si c'est un paquet à envoyer sur le réseau, il est transmis sur la couche liaison.

L'application de cette architecture générique est montrée par la figure 4.23.

Ce modèle de routage réceptionne et filtre les paquets sur le port par défaut 654. Ensuite, il vérifie le type de paquet qui peut être un paquet de découverte de route ou un paquet de maintenance de route. En fonction du résultat, le paquet est acheminé vers le bloc correspondant. Une illustration du diagramme Stateflow de la gestion de requête est montrée par la figure 4.24.

Ce modèle contient différents blocs logiques garantissant chacune des fonctions spécifiques au routage des paquets. La position des blocs sur le diagramme n'a pas d'importance particulière pour la transformation du diagramme en code source. Elle est positionnée de gauche à droite pour faciliter la conception et la lecture du modèle. Ce modèle peut être expliqué comme suit :

— tout d'abord, les paquets sont transmis par le système au premier bloc (à gauche de la figure4.23) que l'on appelle *Packet_Mgmt*. C'est un bloc d'ordonnancement

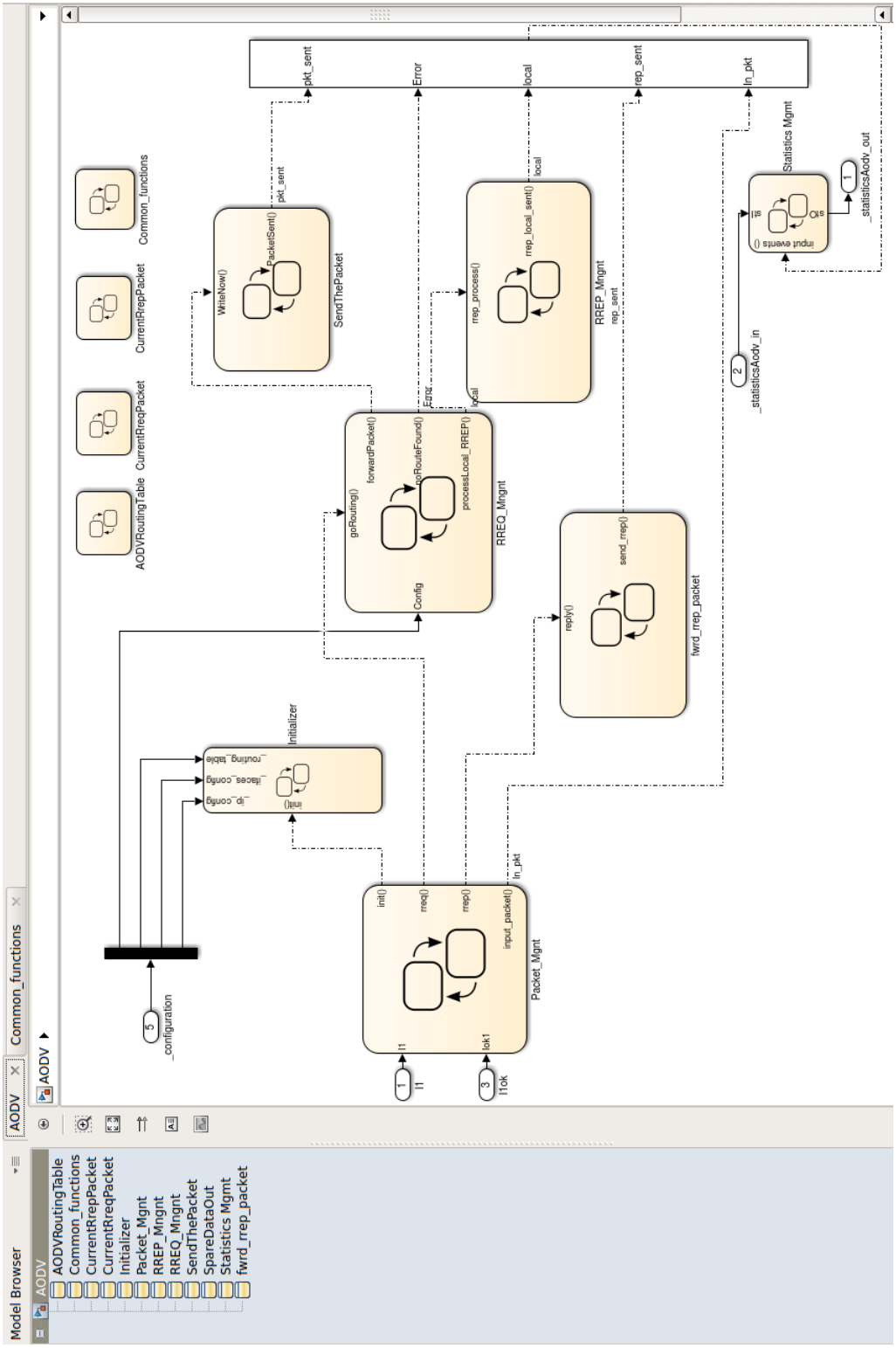

FIGURE 4.23 : Diagramme (version simplifiée) de blocs Simulink pour la fonctionnalité du routage

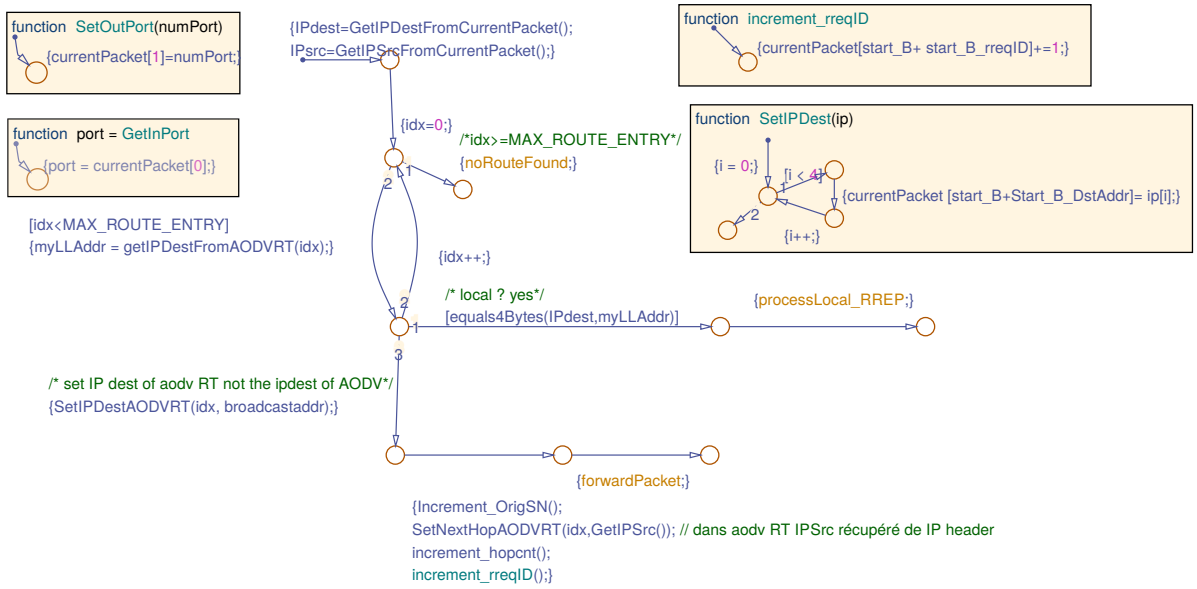

FIGURE 4.24 : Diagramme à états-transitions de routage des paquets (version simplifiée)

qui séquence les paquets reçus pour le transmettre aux blocs suivants. Il assure un étiquetage interne des paquets pour les acheminer vers le bloc correspondant (bloc de découverte ou maintenance de route). Sa mise en œuvre est réalisée par la machine à états illustrée par la figure 4.25. Dans ce bloc, nous récupérons le paquet dans la variable globale *currentPacket* pour pouvoir manipuler ses champs. Le champ type de ce paquet est testé pour savoir si c'est un paquet de type RREQ ou de type RRREP. Quatre sorties sont possibles à partir de ce bloc, la première est l'événement *Init* qui initialise la configuration de la table de routage et des interfaces réseau. Cette configuration est formalisée par un fichier externe qui va être passé au programme. La deuxième sortie est *Input_packet* qui sert à compter le nombre de paquets entrants. Le programme choisit ensuite entre deux blocs (bloc de découverte ou de maintenance de route), selon le résultat du champ type identifié. Les deux autres sorties concernent la gestion de la requête et de la réponse.

— ensuite, s'il y a un message de requête, le processus du programme passe au bloc *RREQ_Mgmt* à travers l'événement *goRouting*. Dans ce bloc, l'adresse IP source (IPsrc) et destination (*IPdest*) du paquet entrant sont récupérées. Ensuite, nous testons si le nombre de requêtes n'a pas dépassé le nombre maximum *MAX_ROUTE_ENTRY*

qui est fixé à 16 dans notre cas. S'il a dépassé cette valeur, le programme sort en envoyant une variable *NoRouteFound* à la sortie.

Ensuite, l'adresse IP du nœud local *MyLLAddr* est récupérée. Nous évaluons si cette variable est équivalente à une variable locale *IPdest*. En fonction du résultat, le programme passe la main soit à l'événement *processLocalRREP* pour envoyer un message RREP à la source, soit l'événement *Forwarding* pour diffuser le message aux nœuds voisins.

Trois résultats sont possibles après le déroulement de ce bloc :

1. si aucune route ne correspond à la demande, alors le paquet est rejeté. Le signal *NoRouteFound* visible sur le diagramme est alors déclenché et sert à incrémenter une variable interne du logiciel ;
2. dans le cas où l'adresse de destination du paquet correspond à l'adresse du nœud recevant le paquet, la machine à états de routage redirige le paquet vers le sous-système *ProcessLocal*. Ce sous-système permet de répondre à une requête.
3. si l'adresse de destination ne correspond pas à l'adresse locale du nœud courant, le paquet est alors routé vers le port de sortie via le sous-système *Forwarding* qui consiste à transmettre le paquet au nœud suivant dans la table de routage.

Il est important de préciser que la conception de la structure de la table de routage a été ajoutée dans l'étape de code de collage. En effet, ici, il s'agit d'une table dynamique dont les entrées changent en fonction de la connectivité. Il est donc difficilement représenté en modèle sur un diagramme Stateflow dont les entrées sont formalisées statiquement.

Nous avons montré le diagramme à états et transition de routage des paquets requêtes par la figure 4.24. Par rapport à cette figure, nous pouvons constater que l'algorithme teste itérativement chaque entrée entrainant le comportement associé : *forwaPracket* ou *NoRouteFound* ou *ProcessLocal_RREP*.

— le bloc *SendThePacket* quant à lui permet de récupérer le paquet à travers la variable globale *currentPacket* et de l'envoyer dans le réseau.

— le bloc *RREP_Mgmt* permet de créer un paquet RREP et l'envoyer vers le nœud source. En particulier, dans ce bloc, l'événement *fwrd_rrep_packet* permet à un nœud intermédiaire de mettre à jour les champs « TTL », « HopCount » et « IP destination » du paquet RREP reçu et de l'envoyer au nœud suivant (vers le nœud source).

— enfin le dernier bloc *Statistics_Mgmt* reçoit les variables des différents blocs qui permettent de faire des statistiques pour savoir si le paquet est reçu, envoyé ou s'il y a eu des erreurs.

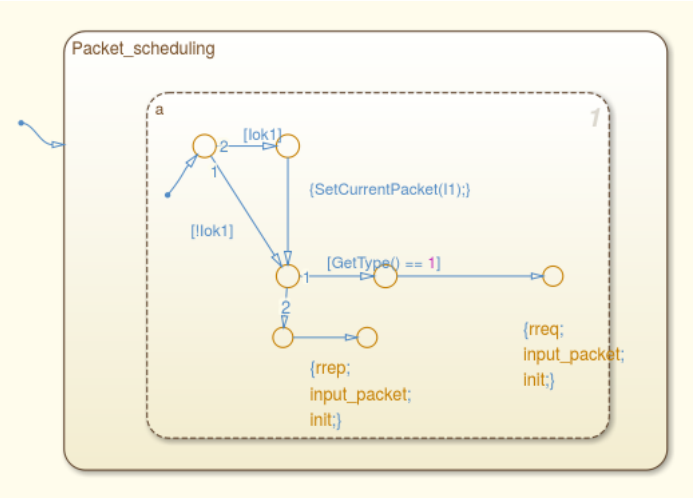

FIGURE 4.25 : Diagramme à états-transitions d'ordonnancement des paquets (version simplifiée)

À la fin du parcours de ce modèle, le paquet est envoyé au bloc de traitement des paquets de routage pour être envoyé à la couche applicative ou à la couche, pour signaler la fin des traitements.

4.5.2.2 Partition de sécurisation

Pour concevoir la classe de partition des fonctionnalités sécurisées, nous avons utilisé une deuxième architecture générique montrée par la figure 4.26.

Cette architecture générique est composée de plusieurs blocs fonctionnels assurant l'identification du paquet jusqu'à l'encapsulation des entêtes du protocole de routage sécurisé. Il est aussi important de remarquer qu'il fonctionne avec le modèle de routage que l'on a présenté précédemment. Le bloc modèle de routage représenté par le bloc *Routing verifier* vient s'ajouter à cette partition.

Ce modèle peut être expliqué comme suit : à la réception d'un paquet, le bloc *Packet identifier* supprime les paquets de routage n'ayant pas d'extension sécurisé. C'est une première étape de filtrage pour éliminer les paquets non autorisés. Cette suppression est matérialisée par le bloc *Packet rejector*.

Ensuite, nous procédons à la désencapsulation du paquet sécurisé qui est effectué par le bloc *content extractor*. Dans ce bloc, nous allons procéder à différentes vérifications de sécurité du paquet. Premièrement, le bloc *wormhole tester* est appelé pour appliquer les calculs de distance et de fonction de hachage présenté à la section 4.3.6. Le bloc *wormhole tester* identifie le type de paquet (découverte ou maintenance de route) et applique ainsi le traitement approprié. En cas d'échec de test de *wormhole tester*, le paquet est rejeté par

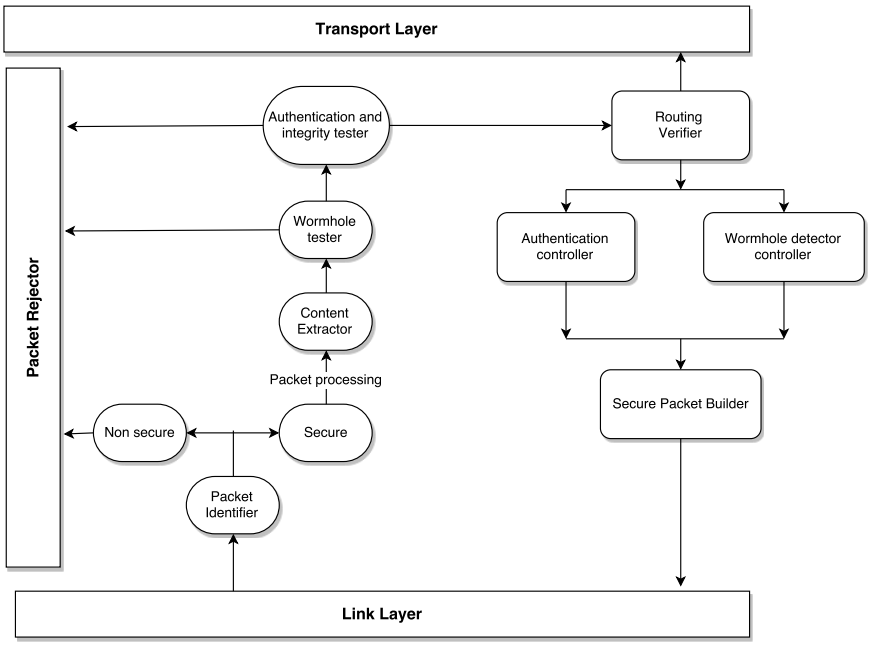

FIGURE 4.26 : Modèle d'architecture pour la conception de la partition de sécurisation

le module *Packet rejector*. Deuxièmement, le paquet sera transmis au bloc de vérification d'authentification et d'intégrité. L'objectif de ce bloc est de vérifier la signature des champs non mutables. Le nombre de sauts sera également vérifié par l'intermédiaire d'une chaîne de hachage. Une fois toutes ces vérifications effectuées, le paquet est transmis à la partition de routage que l'on a vu précédemment. Une illustration (version simplifiée) de diagrammes états transitions du bloc *content extractor* est montré par la figure 4.27.

En outre, dans le cas où un paquet doit être retransmis ou si un paquet de réponse est envoyé, suivant son type, il sera ajouté des entêtes de sécurité correspondantes. L'authentification des messages mutables et non mutables est réalisée et les paramètres de détection et de prévention sont ajoutés au paquet. Le paquet sera ensuite envoyé à la classe de partition d'interfaçage pour envoyer le paquet sur le réseau. Cette architecture générique a ensuite été implémentée par Simulink et Stateflow.

4.5.2.3 Partition d'interfaçage du matériel avec la partition de sécurisation et de routage

La dernière classe de partition est la classe de partition d'interfaçage. Elle permet de lier les entrées et sorties du matériel aux entrées et sorties des applications.

Cette classe de partition à deux objectifs complémentaires. Le premier est de recevoir les trames IP, extraire les paquets contenus et les transmettre aux instances de partitions gérant le routage et la sécurisation des paquets. Le deuxième objectif est de créer un paquet de routage en fabriquant les trames, et en rassemblant les différents entêtes de

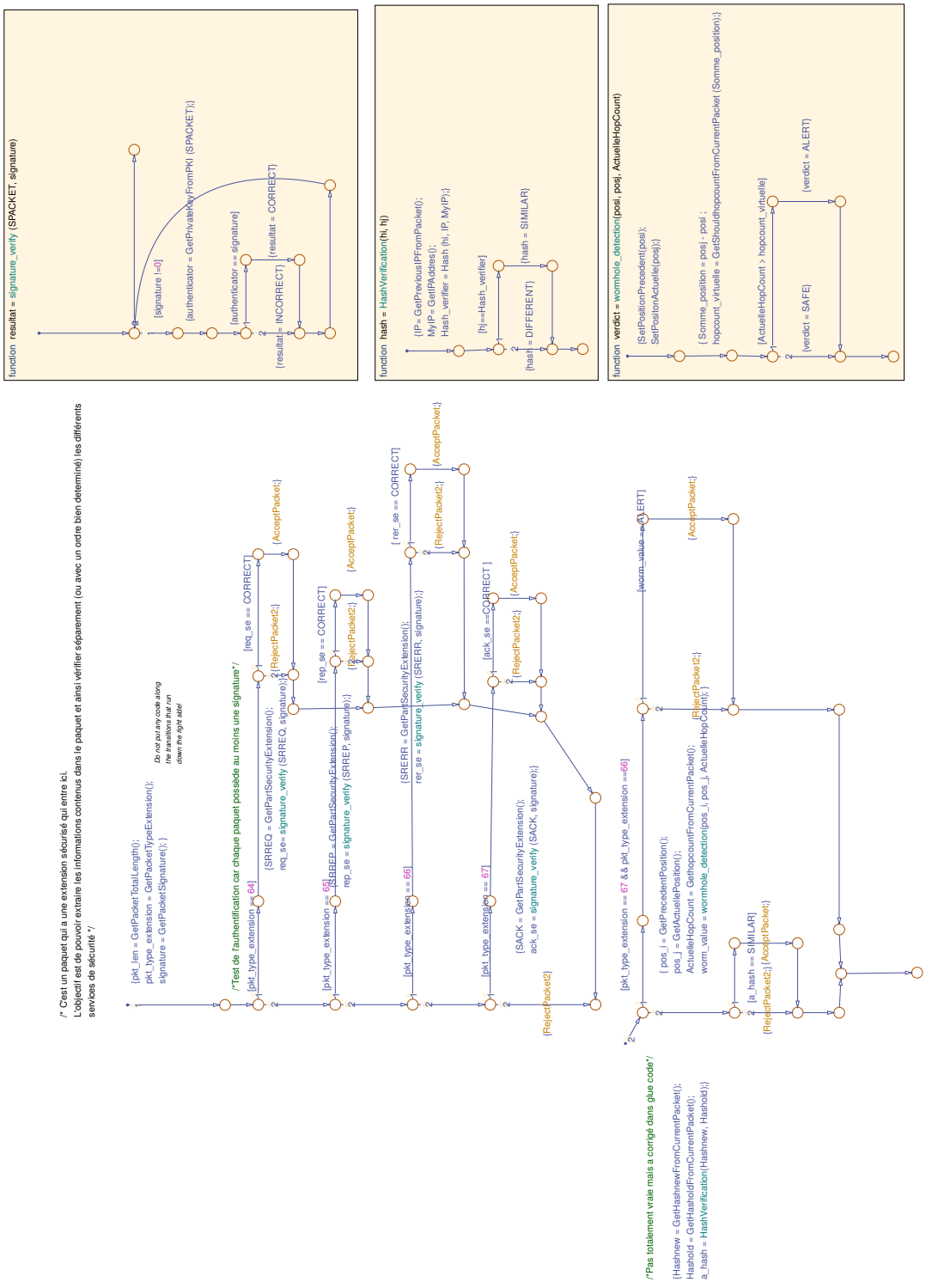

FIGURE 4.27 : Extrait des diagrammes à états-transitions pour le bloc content_extractor

routage et de sécurité.

Il est important de préciser que la partition d'interfaçage a été développée directement en langage C, contrairement aux classes de routage et de sécurité modélisées avec Simulink et Stateflow. Ce choix est justifié par le fait qu'il est particulièrement difficile d'accéder au matériel avec des modèles de haut niveau. Néanmoins, ce choix est compatible avec notre méthodologie de développement, car chaque partition est développée de manière indépendante. Développer entièrement une partition par écriture manuelle de code est en accord avec notre modèle.

4.5.3 Apport de l'approche orientée modèle dans la conception du protocole SUAP

En utilisant les outils de vérification formelle fournis par Mathworks (cf. figure 3.2), l'approche MDD proposée est à la fois une méthodologie et un environnement qui permet de faire du prototypage rapide, depuis une modélisation fonctionnelle jusqu'à son implémentation matérielle. À partir d'un modèle de haut niveau, nous avons pu générer du code C par l'intermédiaire de l'outil *Embedded Coder* expliqué dans 3.5. Nous rappelons qu'avant d'obtenir le code C, le modèle passe par des outils de vérification formelle (interne à Matlab) qui vont permettre de confirmer que le système modèle et le code généré ont le comportement attendu. Les méthodes de vérification formelle sont réalisées via des procédures mathématiques qui vont chercher toutes les exécutions possibles pour identifier les erreurs de conception. L'avantage d'utiliser les outils de Mathworks est qu'il est possible d'appliquer cette vérification sur le modèle et sur le code. D'une part, la vérification formelle des modèles est réalisée pour corriger les erreurs de conception du modèle. Cette identification d'erreur est obtenue par une simulation du modèle pour valider chaque partitionnement. D'autre part, la vérification formelle du code est réalisée par une analyse statique du code et des méthodes formelles qui vont permettre de détecter différents types d'erreurs de type « débordement de variable », « division par zéro », accès non autorisé à une variable de classe et d'autres erreurs d'exécution du code source. Nous pouvons particulièrement distinguer les deux outils suivants :

— tout d'abord, le modèle SUAP passe par l'outil Simulink Design Verifier qui va appliquer des méthodes formelles pour confirmer le bon comportement du modèle. Il passe en revue chaque bloc fonctionnel au sein du modèle les erreurs telles que le dépassement d'entier, la logique morte, la division par zéro et les violations d'accès à un tableau. Pour chaque erreur trouvée, l'outil indique le bloc ou le diagramme Stateflow conduisant à l'erreur par un jeu de test de simulation. Ce dernier permet ensuite le débogage.
— ensuite, le Simulink code inspector est utilisé qui va comparer le code généré avec le modèle source. Il examine systématiquement chaque bloc, diagramme d'états et

paramètres du modèle pour déterminer leur équivalence dans l'opération, les opérateurs et les données dans le code source généré. Il génère ensuite des rapports de traçabilité pour permettre d'améliorer le modèle et de soumettre aux autorités de certification la preuve de la conformité entre le modèle et le code source.

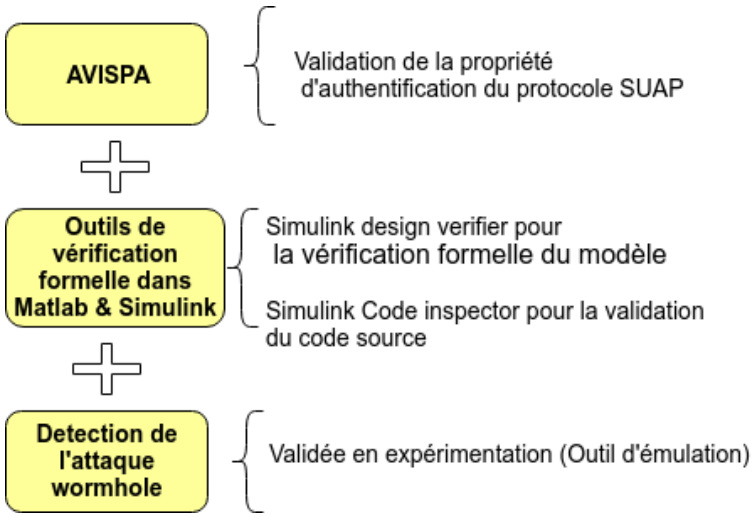

FIGURE 4.28 : Synthèse des différentes démarches de validation pour la conception du protocole SUAP

Ces outils de vérifications formelles ont aidé à la vérification de la conception du protocole SUAP. Elle vient complète la vérification des propriétés de sécurité faite par AVISPA décrite dans 4.4. Le mécanisme de détection du protocole SUAP sera validé en expérimentation présentée dans le chapitre suivant (cf. figure 4.28).

4.5.4 Implémentation du protocole SUAP

L'implémentation système du protocole SUAP résulte de l'association des codes générés par *Embedded Coder* (qui est le générateur inclus dans Matlab) et les codes de collage que nous avons ajoutés manuellement.

SUAP est implémenté en C (la figure 4.29 illustre une représentation de cette implémentation) et localisé dans l'espace *kernel* (exécution du module suapk.ko) du système Linux. Il est également caractérisé par un *daemon suapd* dans l'espace utilisateur qui communique avec le module noyau à travers la bibliothèque libnl [Gra] (Netlink Protocol Library). Cette libraire permet une communication du noyau avec l'espace utilisateur à travers un socket Netlink. Cette communication se traduit par l'échange de données qui va permettre, par exemple, de construire des messages de routage, l'envoi ou la réception des données.

Par ailleurs, libnl offre également des méthodes permettant de gérer le framework *netfilter*. Ce dernier est chargé de gérer les différents paquets réseau transitant sur une machine.

Il met à disposition des appels système permettant au protocole de routage de manipuler des paquets transitant sur les interfaces réseau.

Ensuite, il a été nécessaire de compiler par un procédé de *cross-compilation*, le module noyau obtenu dans le système Linux de l'architecture ARM utilisé par les drones DT18. Voici la liste des opérations que nous avons effectuées pour exécuter le protocole dans un drone DT18.

1. Cross-compilation du protocole SUAP. Cette étape consiste a générer un exécutable pour une architecture autre que celle utilisée pour la création de ce dernier. Dans notre cas, nous utilisons un cross-compilateur pour architecture ARM alors que le PC utilisé possède un jeu d'instructions x86. Cette méthode de *cross-compilation* nous a permis de gagner du temps à la compilation (les PC sont plus puissants que les plateformes ARM) et d'avoir accès à des outils plus ergonomiques pour le développement, comme des programmes de gestion de projets par exemple, qui ne sont pas disponibles sur la cible, puisque dépourvue dans notre cas d'interface graphique.
2. Génération de noyau avec openembedded et bitbake [Lau05]. Le protocole SUAP s'exécute dans un environnement Linux, construit grâce à OpenEmbedded. Il faut donc, lors de la génération de l'ensemble des fichiers qui seront mis dans le système ARM, prendre en compte les exécutables formant le protocole SUAP.
3. Étape de configuration du noyau Linux embarqué. Ceci est nécessaire pour compléter les dépendances manquantes des fichiers sources du noyau.
4. Test unitaire du protocole SUAP. C'est-à-dire exécuter le module dans les cartes ARM utilisées en configurant le modem utilisé en mode ad hoc.

En outre, il est important de souligner qu'il était nécessaire de créer une application de génération de clés pour s'appairer au protocole SUAP. Comme la création d'un système de gestion de clés n'est pas traitée dans cette thèse, nous avons décidé dans la version actuelle de notre protocole de routage de charger automatiquement les clés utilisées au démarrage du drone.

4.6 Conclusions

Dans ce chapitre, nous avons présenté les mécanismes de sécurité du protocole SUAP. Nous avons commencé par décrire comment le protocole SUAP est fondé sur la base du protocole AODV. À l'aide de notre outil hybride de test fait pour un réseau UAANET, nous avons montré que AODV est le protocole qui offre de meilleures performances par rapport à DSR et OLSR en matière d'*overhead*, de délai de bout en bout et de temps de récupération de route. Par la suite, des mécanismes de sécurités ont été ajoutés pour

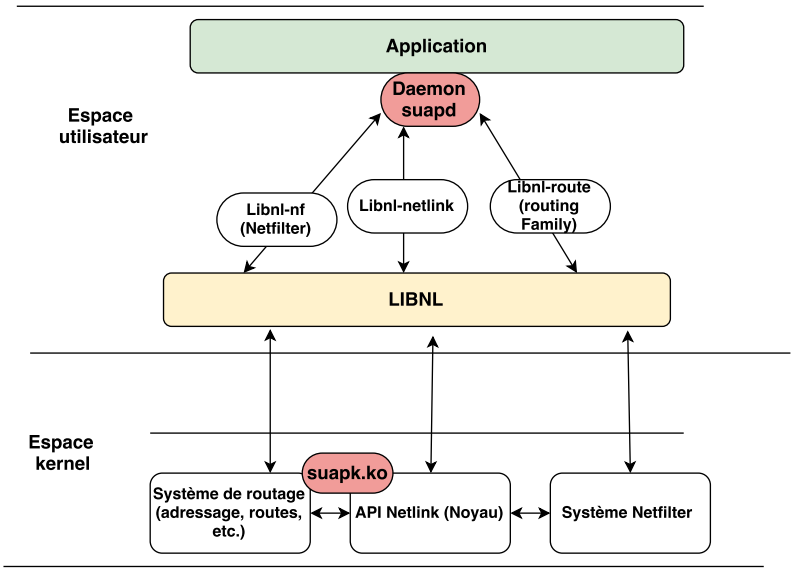

FIGURE 4.29 : Architecture du système Linux avec SUAP

créer le protocole SUAP. Son idée fondatrice est de garantir l'authentification des champs mutables et non mutables séparément. Ce mécanisme a été inspiré du protocole SAODV que l'on a amélioré par la suite, avec des approches de sécurité pour contrer des variantes de l'attaque *wormhole*. Le principe de cette approche de détection et de prévention contre l'attaque *wormhole* est l'association entre le nombre de sauts et la distance relative entre deux nœuds. Nous avons également ajouté un autre mécanisme contre l'attaque *wormhole* qui consiste à prendre en compte l'identité du nœud suivant dans le calcul de la valeur d'une empreinte. Le premier mécanisme est implémenté pour sécuriser les paquets de maintenance de route et le second pour les paquets de découverte de route. Par ailleurs, nous avons également présenté les mises en œuvre réalisée par la méthode orientée modèle. Le protocole SUAP a été vérifié formellement avec l'outil AVISPA qui a pu démontrer le respect de l'authentification des messages. Cette phase de vérification formelle était indispensable pour entamer sa validation, son implantation au sein d'une application réelle et l'évaluation de ses performances.

En outre, dans ce chapitre, nous avons également présenté la mise en œuvre des fonctionnalités du protocole SUAP à travers 3 classes de partitions, dont une partition de routage, de sécurité et d'interface avec la couche bas niveau et entre les fonctionnalités de la couche réseau. L'originalité introduite dans cette conception vient de l'emploi de modèles de haut niveau dans la mise en œuvre de protocole existant. La puissance d'expressivité des modèles apporte la rapidité de conception, la facilité de lecture du système, la réutilisation et l'adaptation des modèles.

Dans le prochain chapitre, nous monterons les résultats d'expérimentation du protocole SUAP réalisée à la fois en émulation et en environnement réel.

Chapitre 5

Évaluation des performances du protocole SUAP par simulation et par expérimentation réelle

Dans ce chapitre, nous allons parler de la dernière phase de validation (fonction de routage et de sécurité) de notre protocole de routage en détaillant l'évaluation des performances de la partition routage et sécurité du protocole SUAP. Ces expérimentations ont été réalisées à la fois en local par l'outil d'émulation présenté dans le chapitre précédent et aussi en environnement réel par l'utilisation des drones DT-18 détaillés en introduction. Nous commencerons par présenter les différentes topologies réseau envisagées, puis nous continuerons par les différentes métriques permettant de quantifier les performances réseau. Enfin, nous présenterons les résultats obtenus qui nous permettront de conclure sur la conformité entre la spécification du cahier des charges de ce projet SUAP et la fonctionnalité de notre architecture de communication sécurisée.

Contents

5.1 Introduction 156
5.2 Cadre d'expérimentation 157
5.2.1 Topologies réseau d'études 157
5.3 Validation de la partition de routage 159
5.3.1 Topologie de test de routage 159
5.3.2 Métriques considérées 160
5.3.3 Résultats obtenus et interprétation 162
5.4 Validation des fonctions de sécurité du protocole SUAP 170
5.4.1 Validation de la partition de sécurisation (authentification des messages) en environnement émulé 171
5.4.2 Validation de la partition de sécurisation (authentification des messages) en environnement réel 177
5.4.3 Validation du mécanisme de détection contre l'attaque *wormhole* 184
5.5 Conclusions 188

5.1 Introduction

Ce chapitre présente la validation des fonctions de routage et de sécurité réalisées par une étude d'évaluation des performances du protocole de routage SUAP. Cette étude de performance est d'une part réalisée à partir de l'outil hybride d'émulation et de simulation présenté dans la section 4.2.1. Nous rappelons que cet outil a été créé pour se rapprocher de la condition réelle de vol d'une flotte de drones et pour que l'on puisse tester en local notre protocole avant de l'évaluer en environnement réel. Ce rapprochement est représenté par l'exécution du protocole dans un système d'exploitation proche de celui utilisé par les drones. Des modèles de mobilités réels des drones sont également introduits dans l'outil. Le choix de combiner le concept de simulation et d'émulation est aussi motivé par la réduction de la tache de développement pour l'implémentation du protocole. En effet, nous n'avons besoin que de développer une seule version du code (écrit en C) qui fonctionne à la fois dans une architecture x86 et une architecture ARM. D'autre part, nous présenterons également dans ce chapitre, les différents scénarios de test réel qui ont été réalisés pour valider la fonction de routage et de sécurité du protocole SUAP en environnement réel. L'intérêt d'effectuer des tests réels est de valider en environnement réel le fonctionnement attendu de l'algorithme développé à partir d'une approche orientée modèle.

Dans ce chapitre, nous commencerons par présenter le cadre d'expérimentation, puis nous continuerons par la validation de chaque partition en présentant à la fois la topologie réseau d'évaluation et les différentes métriques permettant de quantifier les performances réseau de notre protocole. Enfin, les résultats obtenus seront interprétés.

Il est important de préciser que l'objectif des évaluations de performances que l'on présente ici consiste à vérifier les différentes fonctions de notre protocole de routage. C'est-à-dire qu'elle s'inscrit dans le registre, de la méthodologie de prototypage rapide expliqué dans le chapitre 3 pour permettre de valider les fonctionnalités non validé avec les outils de vérification formelle inhérente à l'outil Matlab & simulink et AVISPA. Dans ce cas, elle permet de valider la dernière étape de la méthodologie (résumé de la figure 3.2) concernant l'exécution du binaire finale sur le système final.

5.2 Cadre d'expérimentation

5.2.1 Topologies réseau d'études

Il est nécessaire de mettre en place des topologies d'expérimentations adéquates avec les différents systèmes pour permettre de valider chaque partition du protocole SUAP. Ici, la partition signifie les différents modules (ou fonction) du protocole de routage. Nous supposons que nous avons trois modules à vérifier à savoir le routage, les mécanismes d'authentification et d'intégrité et la solution contre l'attaque *wormhole*. Nous avons donc 3 validations à faire pour le protocole SUAP. Pour cela, des topologies de test réel et d'émulation ont été mises en place. Le système de drones utilisé pour la réalisation d'une plate-forme de test est composé des éléments suivants :

— Trois aéronefs de type DT18 ;

— Trois stations de pilotage associées à chaque drone . Il est important de préciser que seule une des stations (station 1 en considérant la figure 5.1) reçoit les données de la charge utile provenant du drone le plus proche. Les autres stations (par exemple, le nœud «station 2» et le nœud «station 3» sur la figure 5.1) ne sont sur le réseau que pour répondre à la réglementation française de déploiement de drones civils qui exige un lien de sécurité associé à chaque drone à chaque instant (pour anticiper un éventuel dysfonctionnement du système UAS) ;

— 3 logiciels de supervision pour la gestion de la mission (suivi du plan de vol, interactions avec les composantes de l'environnement).

L'interaction entre ces sous-systèmes est montrée par la figure 5.1. Sur cette figure, une liaison unique est dédiée pour la communication entre l'autopilote de chaque drone et une station de contrôle. Sur cette topologie, l'information de charge utile est envoyée depuis le drone Dr3 jusqu'à la station 1 par l'intermédiaire des nœuds voisins (Dr2 et Dr1). Il est à noter que c'est seulement l'information acquise par la charge utile que l'on relaie sur le réseau lors des tests réels effectués. Les trafics de commande sont envoyés en communication directe vers chaque drone.

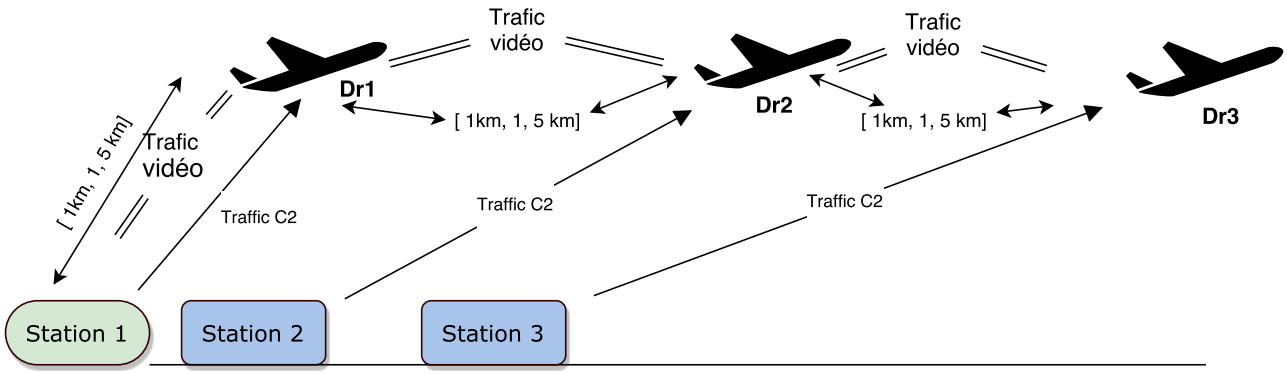

FIGURE 5.1 : Topologie commune de test réel

5.2.1.1 Drones DT18

L'entreprise Delair-Tech possède plusieurs gammes de drones qui peuvent répondre à plusieurs missions. Le produit le plus mature, car le premier à avoir été développé est le DT18. Il a été opéré avec succès dans des milieux très variés (montagne, bord de mer, plaine) et pour toutes sortes de missions grâce au large éventail de charges utiles qu'il peut emporter. La description technique de ce drone est présentée dans le tableau ci-dessous.

Caractéristique	**Valeur**
Modèle	DT18
MTOW	< 2 kg
Payload	250 g
Portée	100 km
Vitesse en vol	50km/h
Résistance au vent	jusqu'à 45km/h
Transmission temps réel	Jusqu'à 15km.
Autopilot	Delair technology

Tableau 5.1 : Spécification du drone DT18

5.2.1.2 Stations de contrôle

Deux types de stations ont été utilisées pour réaliser les tests en vol. Ces stations sont celles qui sont utilisées actuellement et rendues disponibles par l'entreprise Delair-Tech. Elles sont souvent déployées pour l'utilisation classique des drones DT18 et permettent de suivre l'évolution du drone pendant sa mission. Elles permettent notamment la modification du plan de vol en envoyant les données C2 (commande et contrôle) traduisant le modèle de

mouvement souhaité (cercle, waypoint, rectangle, etc.) à travers un logiciel de contrôle sol.

Par ailleurs, il est important de préciser que nous avons directement utilisé les stations existantes sans modification, car elles permettaient déjà de recevoir des informations de navigation du drone. En revanche, il était impossible de piloter trois drones simultanés avec une seule station. Il a donc été nécessaire de déployer trois stations et de faire en sorte qu'il n'y ait pas d'interférence pour l'envoi des commandes. Nous avons donc modifié le canal radio de chaque liaison de commande pour avoir un lien unique pour chaque drone.

5.2.1.3 Liaison de communication

Nous avons énoncée précédemment que chaque drone est piloté par une station sol spécifique. Pour éviter une interférence pour l'envoi de la commande des trois drones, trois types de lien sont déployés pour chaque drone. Chaque drone est également associé à un lien de sécurité qui est utilisé dès que la carte embarquée au bord du drone est en panne ou lorsque la connexion à travers le lien principal est mauvaise.

Par ailleurs, les trafics capturés par la charge utile sont véhiculés avec un lien modifié 2.4 GHz. C'est surtout ce dernier qui nous intéresse pour la communication ad hoc.

5.3 Validation de la partition de routage

5.3.1 Topologie de test de routage

Nous nous sommes d'abord focalisés sur la validation de la partition routage. Cette partition se base sur le protocole AODV que nous avons étudié dans le chapitre précédent. Nous avons choisi de développer une nouvelle version du protocole en nous appuyant sur la méthode de prototypage rapide présentée précédemment. Ce choix se justifie par notre objectif de certification finale du produit qui reste plus accessible en menant une approche de conception orientée modèle plutôt que de certifier un code logiciel existant (approche faisable, mais souvent très onéreuse, car beaucoup plus longue). Le détail de cette modélisation est détaillé dans la section 4.5.2.1. Ensuite, nous nous sommes intéressés à l'étape de validation permettant de tester les fonctionnalités de cette partition. Pour cela, nous avons choisi une topologie composée de trois nœuds comme le montre la figure 5.2.

L'objectif de cette topologie est de faire transiter de la vidéo à partir du drone Dr2 vers la station sol à travers le drone Dr1. Les trafics échangés sont montrés dans le tableau 5.2. Cette validation ne nécessitait pas l'utilisation d'un troisième drone, car avec trois nœuds, nous pouvions déjà évaluer la performance de la partie routage du protocole SUAP.

Nous pouvons diviser les trafics applicatifs échangés entre les nœuds en 4 classes :

1. Tick : flux de signalement pour communiquer au drone qu'il est toujours en com-

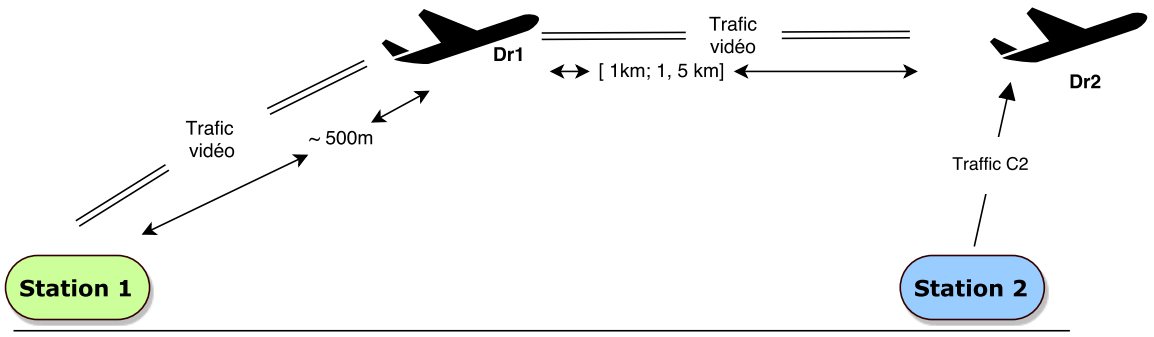

FIGURE 5.2 : Topologie de test pour la partition de routage du protocole SUAP

Type	Source —— Destination	Taille du paquet	Débit d'échange
Tick	Station1 —— Dr1 Station2 —— Dr2	64 octets	1 paquet/sec
Georef	Dr1 —— Station1 Dr2 —— Station2	80 octets	3 paquets/sec
Commande	Station 1 —— Dr1 Station 2 —— Dr2	80 octets	1 paquet/sec
Vidéo	Dr2 ——— Station1	1400 octets	25 datagrames UDP/sec width=720,height=576

Tableau 5.2 : Différents flux échangés lors du test en vol

munication avec la station au sol. Environ 100 octets envoyés à une fréquence de 1 Hz.

2. Georef : flux permettant de déterminer la position du drone
3. Le trafic de commande du drone proprement dit pour l'envoi des commandes à l'aide d'une application de *monitoring*. La figure 5.3 illustre le mouvement du drone Dr1.
4. Video : trafic émis en temps réel depuis le Drr (à partir d'une caméra embarquée à bord du drone) vers la station 1 à travers le drone Dr1.

5.3.2 Métriques considérées

Pour évaluer la performance du protocole de routage, nous avons considéré les métriques suivantes :

— *Overhead* qui mesure la quantité des trafics de signalisation échangés entre les nœuds

— Stabilité de la route qui mesure le délai au cours duquel la connectivité n'est pas interrompue.

— Délai de rétablissement de route pour mesurer la réactivité du protocole de routage à trouver une route en cas de perte de lien

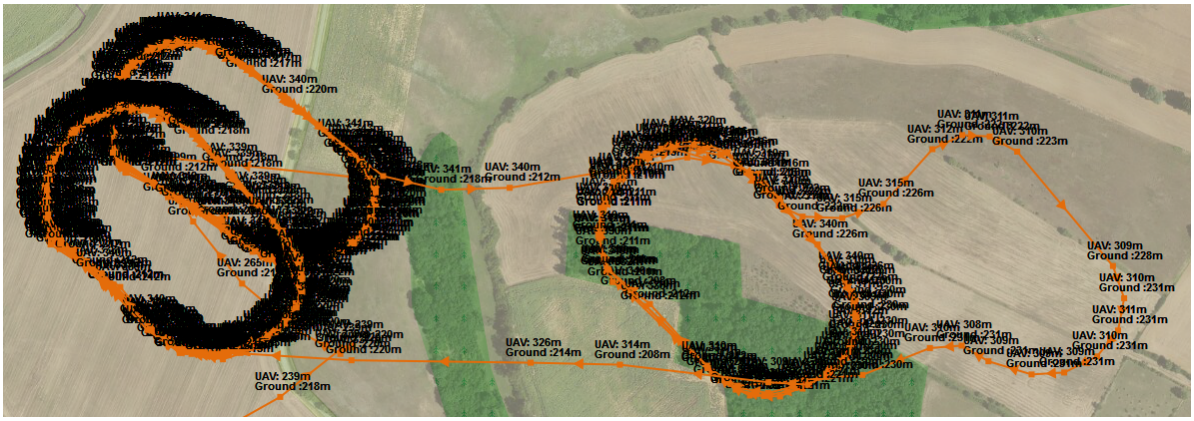

FIGURE 5.3 : Mouvement du drone Dr2

— Délai de bout en bout

— Taux de pertes de paquets de données pour mesurer la quantité des paquets de données perdus

Par ailleurs, les valeurs que l'on a obtenues durant ce test seront à chaque fois comparées avec les résultats d'évaluation par émulation du protocole AODV que l'on a présentés dans le chapitre 4. Ces résultats sont résumés dans le tableau 5.3. Cette comparaison nous permet ainsi de situer les valeurs obtenues pour les tests réels et de conclure sur la pertinence de ces résultats dans le cadre d'une comparaison avec une implémentation de référence du protocole AODV (implémentation présentée dans le chapitre 4).

Métrique	**Protocole AODV**
overhead (Paquets de contrôle)	501 ko
overhead (% Trafic (octets))	0.034 %
Délai moyen de bout en bout	5.32 ms
taux de pertes	3.05 %
Connectivité	88.5 %
Durée de vie d'une route	18.53 secondes
Délai de ré-établissement de route après une perte de connexion	1.94 ms

Tableau 5.3 : Synthèse des résultats obtenus pour l'évaluation du protocole AODV en émulation

Overhead	valeur
Paquets contrôle	552 ko
Trafic % (octets)	0.05 %

Tableau 5.4 : *Overhead* du protocole pour un test de 28 min

5.3.3 Résultats obtenus et interprétation

5.3.3.1 Taille d'overhead

Le tableau 5.4 illustre le résultat de l'*overhead* que l'on a recensé. Au regard des résultats obtenus et en les comparant à la performance du protocole de référence AODV que l'on a synthétisée dans le tableau 5.3, nous pouvons conclure que l'*overhead.* généré par le protocole de routage reste dans le même ordre de grandeur que les résultats obtenus en émulation. Nous pouvons conclure que les trafics générés par le protocole de routage ne perturbent pas le transfert de données utiles. Ici, le résultat est expliqué à la fois par la non-utilisation d'acquittement à chaque paquet envoyé et aussi par la non-nécessité de constamment mettre à jour une route déjà établie. En effet, une fois qu'une route entre la station 1 et le drone Dr2 est établie, le protocole ne cherche plus à savoir la topologie complète du réseau. Chaque nœud se contente de vérifier périodiquement la symétrie du lien avec son voisin direct (voisin à un saut) par des paquets Hello. Nous pouvons conclure le protocole ne génère pas un nombre important de paquets de contrôle (faible occupation de la bande passante allouée).

5.3.3.2 Stabilité d'une route

Il faut savoir que l'*overhead* généré par un protocole de routage dépend de la mobilité des nœuds dans le réseau. Plus les nœuds sont mobiles et plus il y aura des messages de redécouverte de route échangés entre les nœuds ce qui signifie l'instabilité de la route.

Pour évaluer le degré de stabilité d'une route, il suffit d'inférer le nombre de messages de recherche d'une route envoyé par les nœuds. Le nombre total de ruptures d'une route au cours de la mission est proportionnel au nombre de paquets de découvertes d'une route durant la mission.

Métrique	Valeur
Durée de vie moyenne d'une route	14.34 s
Durée de vie maximum d'une route	34.85 s

Tableau 5.5 : Stabilité de la route entre Dr2 et station 1

Au regard de ces résultats (figure 5.4 et tableau 5.5), nous pouvons observer une coupure de route en moyenne tous les 14.32 secondes. En comparant ce résultat au tableau 5.3

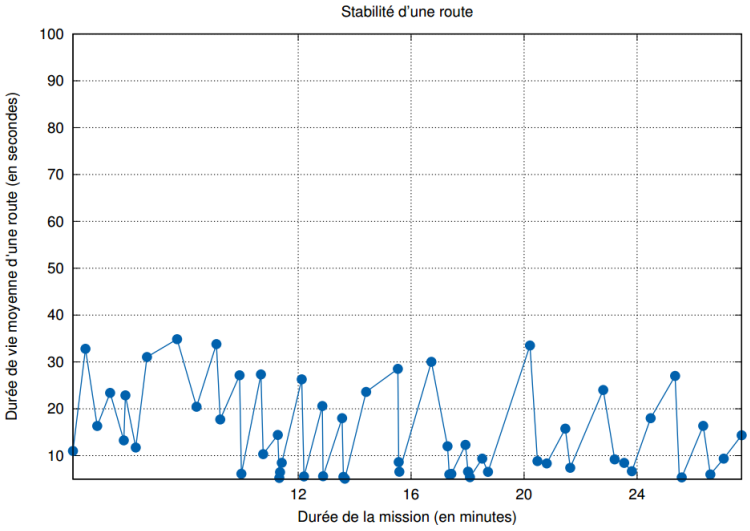

FIGURE 5.4 : Stabilité de route entre le nœud station 1 et le nœud Dr2

qui maintient une route en moyenne tous les 18 secondes (valeur que l'on a obtenue en émulation et simulation en rejouant le même type de mobilité que durant ce test réel), nous pouvons observer une performance moins bonne pour notre implémentation, mais qui reste toutefois dans le même ordre de grandeur. Le résultat que l'on a obtenu sur cette métrique en émulation dépend des modèles de mobilité réels que l'on a inclus. Ce qui n'est pas pris en compte dans l'outil d'émulation était la position des antennes de chaque nœud à chaque changement de modèle de mouvement pendant la mission. En effet, l'outil que l'on a utilisé ne pouvait pas simuler les pertes d'atténuation de signal réceptionné par une antenne. Par conséquent les pertes liées à ce phénomène n'ont pas été quantifiées.

Le résultat obtenu pour ce métrique est causé principalement par le degré de mobilité des nœuds qui crée des déconnexions. Il y a aussi la différence de plan de rayonnement et de polarisation de l'antenne d'émission et de réception qui peut atténuer le signal. Néanmoins malgré la présence de ces pertes de connexion, la durée de rétablissement d'une route (figure 5.7) qui est de l'ordre de 2 ms laisse penser que ces coupures de routes ne dégradent pas significativement la communication.

Afin d'expliquer un peu plus la raison de ces pertes de routes, nous avons corrélé les traces d'analyse des paquets réseau avec le log de mouvement des drones. Les traces montrent des paquets de maintenance et de redécouverte de route à chaque fois qu'un drone tourne (en faisant un virage) en changeant, par exemple, son vol suite à la réception d'un message de position. Une des raisons qui explique ces pertes de route est due au changement de plan (par inclinaison) de l'antenne d'émission du drone Dr1 et de l'antenne de réception du drone Dr2 à un instant donné. Ce changement est causé plus précisément par des modes de mobilité différents entre les deux drones (par exemple, lorsque le drone Dr1 passe en mode waypoint et que le drone Dr2 passe en mode circulaire). En conséquence,

la combinaison de la perte de propagation et de l'atténuation du signal peut induire la perte momentanément d'un lien.

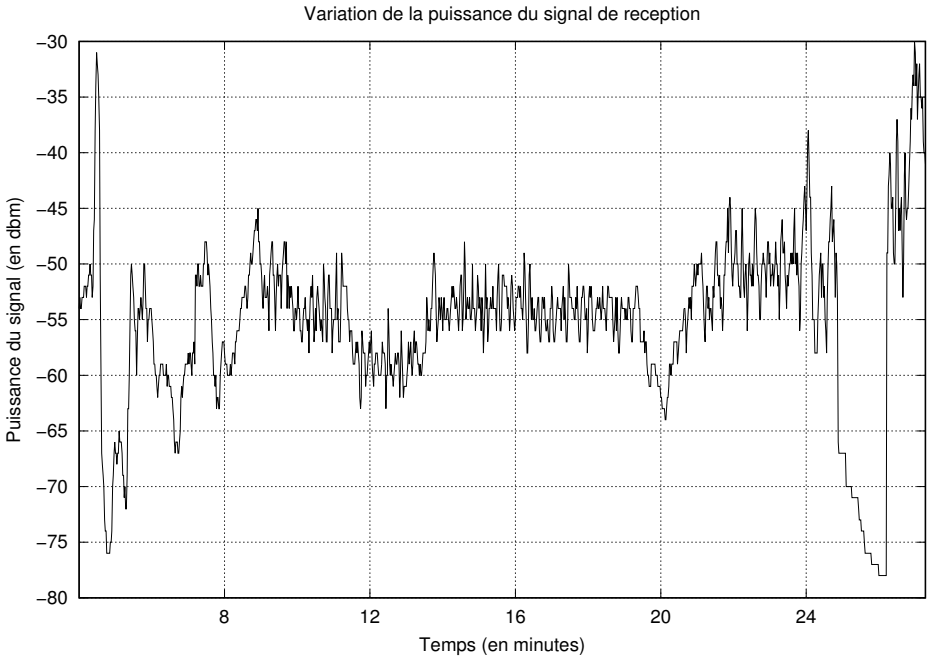

FIGURE 5.5 : Variation de la puissance de réception du signal du drone Dr2

Par ailleurs, les coupures de route peuvent également être expliquées par l'introduction de nombreux paramètres en situation réelle. Par exemple, lorsque le drone est en virage, l'aile intérieure masque l'émission du modem à bord puisque celui-ci est situé sous le drone. Globalement, les nombreux mouvements du drone rendent difficile le maintien de la communication pour une longue durée.

À travers la figure 5.5, nous constatons que la puissance de réception du signal du drone Dr2 présente des oscillations constantes et des pics d'oscillations assez importantes de manière intermittente. L'oscillation constante s'explique par le mouvement instable du drone pendant son exécution d'un plan de vol spécifique, c'est-à-dire que, par exemple, lorsque le pilote envoie une commande pour une trajectoire rectiligne, le drone n'effectue pas totalement des trajectoires rectilignes, mais avec une certaine inclinaison causée par les conditions climatiques (la force du vent, par exemple). Étant donné que les commandes envoyées depuis la station sol sont des successions de mouvements rectilignes, circulaires, rectangulaires, ces mouvements impactent la performance du réseau. En nous inspirant de la figure 5.6 qui montre la corrélation entre la puissance du signal et la durée de vie

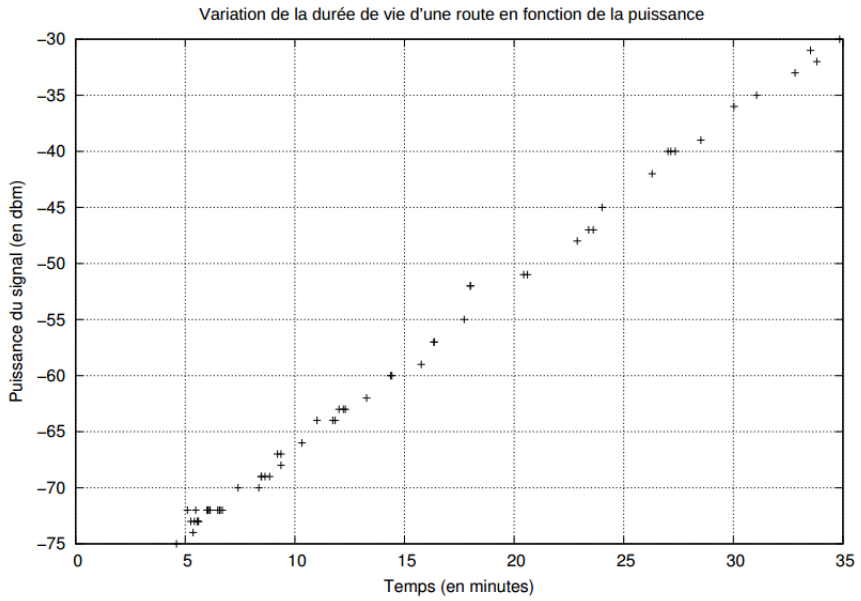

FIGURE 5.6 : Stabilité moyenne d'une route entre Dr2 et la station sol en fonction de la puissance du signal du drone Dr2

d'une route, nous pouvons apercevoir que le délai tend à s'accroitre (meilleure qualité de service) au fur et à mesure où la qualité du signal augmente. La bande passante allouée aux données étant inchangées au cours du temps ; cette fluctuation provient donc d'autres facteurs physiques et topologiques.

5.3.3.3 Durée de ré-établissement d'une route

En ce qui concerne la métrique de délai de ré-établissement d'une route après une perte de connexion, ce délai est égal à l'intervalle de temps entre le départ d'un paquet requête (identifié avec un numéro de séquence) et l'arrivée d'un paquet réponse sur un nœud donné (pour notre cas, le nœud Dr2).

Délai	valeur
Délai moyen	2.08 ms
Délai maximum	4.30 ms

Tableau 5.6 : Délai de rétablissement de route après une déconnexion

Nous pouvons constater que les délais sont assez faibles (tableau 5.6 et figure 5.7). En comparant ces valeurs au résultat d'émulation d'AODV qui est en moyenne 1.94 ms, nous pouvons dire que le protocole semble bien s'adapter au changement de topologie causé par le mouvement des drones. Cela peut aussi s'expliquer sur le fait que le protocole n'attend pas la perte de plusieurs paquets avant de recherche une nouvelle route. Il en résulte

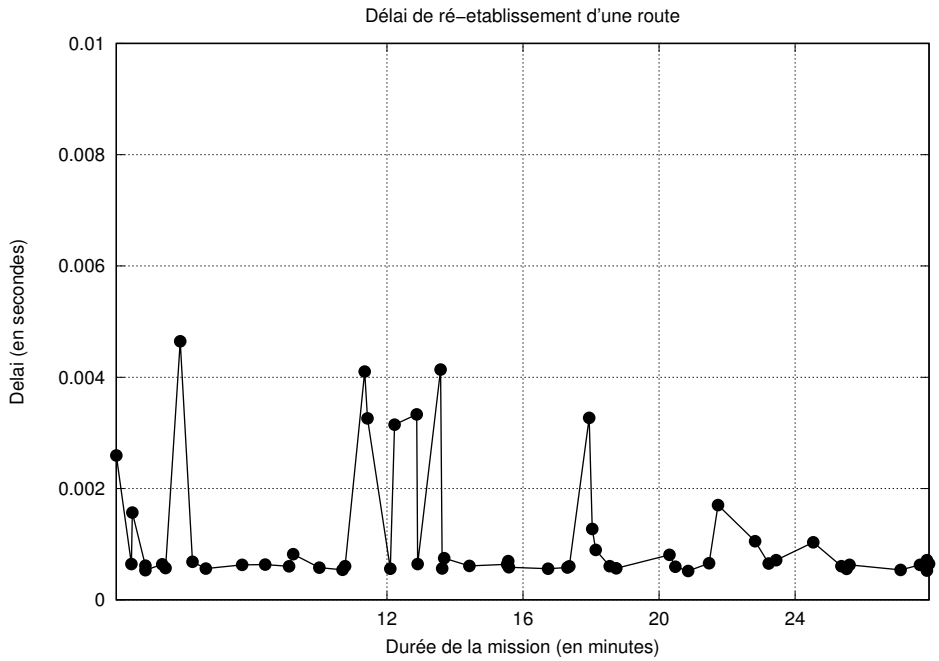

FIGURE 5.7 : Délai de ré-établissement d'une route en cas de perte de lien

que les temps de réparation de route sont relativement faibles pour un environnement si dynamique. Nous pouvons aussi mentionner le fait qu'en cas de coupure de lien entre deux nœuds voisin (par exemple, entre Dr1 et Dr2), le rétablissement de route par des paquets de maintenance ne se fait qu'entre ces deux nœuds. Globalement, le protocole tend à trouver assez vite une route dans le réseau. Ici, la vitesse de déplacement des nœuds a peu d'influence sur cette valeur qui tend vers une constante.

5.3.3.4 Délai de bout en bout

Le délai de bout en bout est la durée de transfert d'un paquet entre l'émetteur et un récepteur final.

Délai	**Valeur**
Délai moyen	5.49 ms
Délai maximal	93 ms

Tableau 5.7 : Délai moyen entre station 1 et Dr2

La répartition de ces délais est illustrée par la figure 5.8.

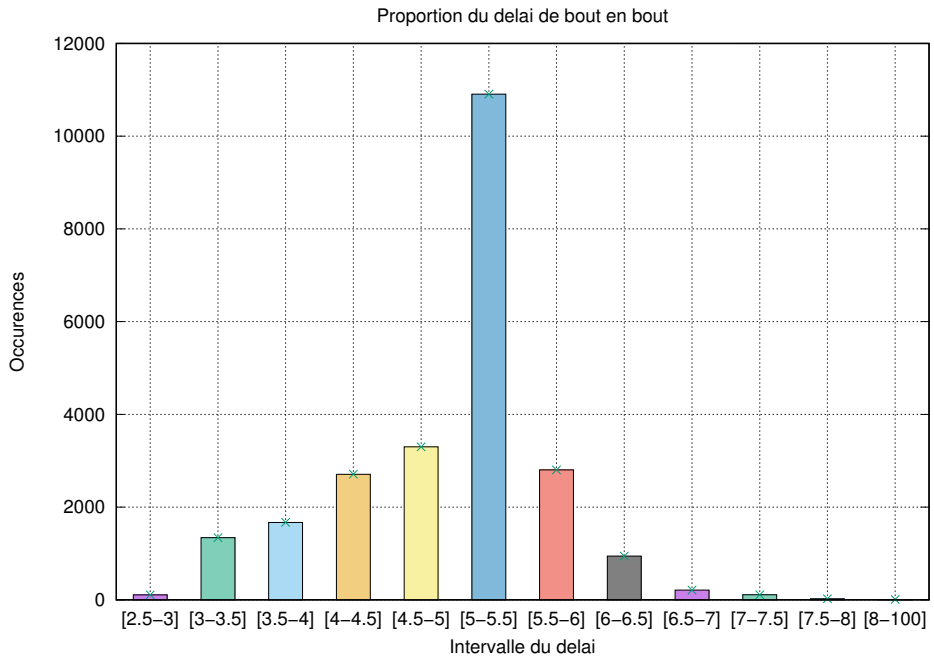

FIGURE 5.8 : Répartition des valeurs délai de bout en bout au cours de la mission

Nous pouvons constater que les délais sont concentrés dans l'intervalle [5, 5.5] millisecondes. En comparant ce résultat au tableau 5.3, nous pouvons dire que notre implémentation se comporte de façon équivalente à la version de référence d'AODV que nous avons testé en simulation (résultats du chapitre 4). Avec ces faibles délais que les deux versions mettent en évidence, les protocoles ne semblent pas introduire un délai supplémentaire significatif. Cependant, ce délai est légèrement plus grand que les valeurs de délai obtenues lors d'étude en simulation du protocole AODV. Nous émettons l'hypothèse que ceci est dû au fait que le protocole de routage (ayant été modélisé) fait face à l'introduction de paramètres supplémentaires de l'environnement temps réel comme la vitesse des drones et le changement en continu de la topologie du réseau.

Comme les critères de qualité de service sur notre flux de données exigent surtout un délai court d'acheminement, la fixation de la durée de mise en attente est donc suffisamment courte. Ce délai faible permet de pouvoir retranscrire au niveau du récepteur (GCS) le flux vidéo acquis par le drone.

5.3.3.5 Délai moyen pour l'acheminement des paquets de données utiles

Ce délai représente le temps qu'il faut à un paquet vidéo de partir du drone Dr2 (nœud source) et d'arriver aux stations sol(nœud destination).

À travers la figure 5.9, on peut noter que le délai tend vers une constante, mais qui présente des pics correspondant au moment où la route a été coupée. Ces pics sont dus généralement à la perte de route et donc à la durée de la mise en attente des paquets. D'après ce que nous avons vu un peu plus haut, ces coupures de routes sont causées principalement par le degré mobilité des nœuds et des pertes de polarisations des antennes. Ces éléments vont engendrer des coupures de liaisons dans le réseau. Par conséquent, les paquets de données sont emmagasinés dans le buffer pour un temps qui va dépendre de la durée de rétablissement de la route.

Comme les critères de qualité de service sur notre trafic de données exigent surtout un délai court d'acheminement et sans grande importance pour le taux de pertes, la fixation de la durée de mise en attente est donc suffisamment courte . Cette valeur faible de délai permet de pouvoir retranscrire au niveau du récepteur (GCS) le flux vidéo acquis par le drone.

Délai	**Valeur**
Délai moyen	9.15 ms
Délai maximal	1.05 s

Tableau 5.8 : Délai moyen pour l'acheminement des paquets de données utiles

5.3.3.6 Perte de paquet de données

Cette métrique a pour objectif de mesurer combien de paquets de données (trafic vidéo) sont perdus au cours de la mission. C'est-à-dire la quantité des paquets perdus entre le moment où un lien est perdu et le moment où il est rétabli. L'évolution du taux de pertes au cours du temps est montrée par la figure 5.10. Nous constatons que pendant toute la durée de la mission, le taux de pertes n'excède pas 4 %. En émulation, cette valeur est égale en moyenne à 3.05 % contre 3.51 % ici en test réel.

Ces deux derniers résultats (taux de pertes et délai d'acheminement des paquets vidéo) permettent de valider les performances de notre implémentation du protocole AODV. En effet, au regard des critères de qualité de service des missions que nous avons identifiées dans notre cahier des charges initial, AODV répond à ces critères de performances et permet de rendre le service pour lequel il a été choisi et mis en œuvre.

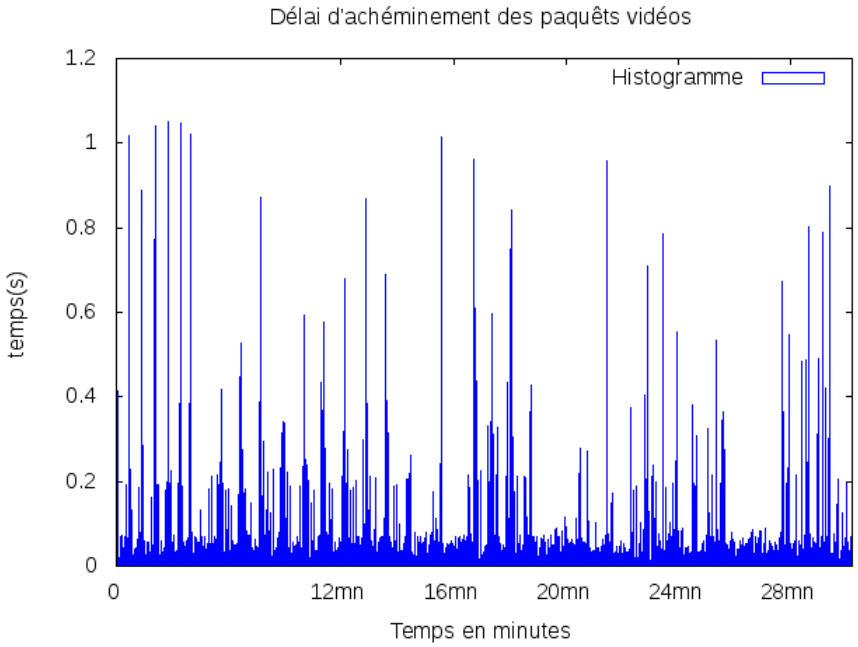

FIGURE 5.9 : Délai moyen pour l'acheminement des paquets de données utiles

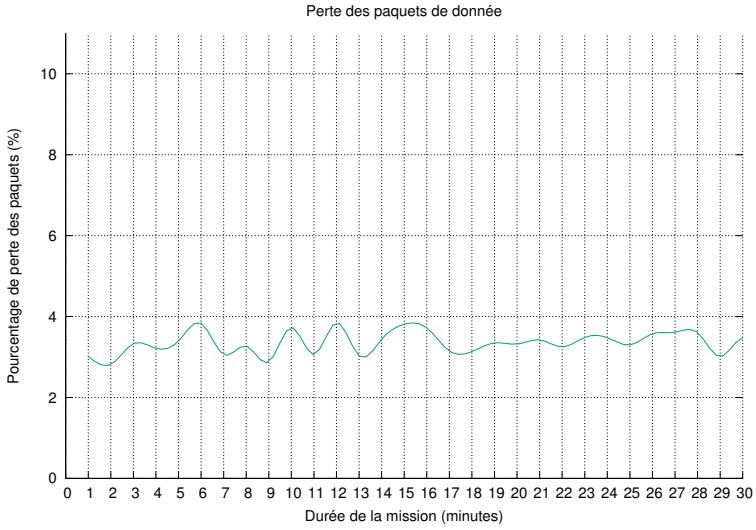

FIGURE 5.10 : taux de pertes mesurés

5.3.3.7 Discussion sur la validation de routage

Dans cette section, nous avons détaillé la validation de la partition de routage du protocole SUAP. Les résultats obtenus sont acceptables compte tenu des exigences du trafic vidéo. Le tableau 5.9 illustre la différence entre les résultats obtenus en simulation avec AODV et les résultats obtenus en test réel. Nous nous sommes intéressés à la quantité de trafic de signalisation généré au cours de la mission. Leur quantité n'occupe que faiblement la bande passante disponible, car le protocole développé ici est réactif et ne génère des paquets de découverte de route qu'en cas de besoin. En ce qui concerne le délai de bout en bout et le délai de rétablissement d'une route, elles sont généralement faibles à cause de la faible densité des nœuds. Ce qui fait que la vitesse de déplacement nœuds a peu d'influence. Ces derniers résultats sont en mettre en regard des critères de performance obtenus dans le chapitre 4 en comparant les différents protocoles de routage existants dans le domaine ad hoc (et en particulier une version existante du protocole AODV). Notre implémentation du protocole AODV obtenu par l'utilisation de la méthode de prototypage rapide orientée modèle offre donc des résultats satisfaisants.

Par ailleurs, à la suite de ce test, nous avons pu mener à terme le cycle de développement de la méthode MDD, c'est-à-dire le processus qui s'échelonne de la spécification jusqu'à la partie déploiement. À la suite de cette étude, nous constatons donc que notre implémentation offre plutôt une bonne adaptabilité vis-à-vis de la spécificité du réseau ad hoc de drones (vitesses des nœuds, degré de mobilité, modèle de propagation, nombre de nœuds, topologie dynamique). Dans la prochaine section, nous aborderons la validation de la partition de sécurisation (authentification et intégrité) du protocole SUAP.

Paramètre	**Résultat d'émulation d'AODV**	**Test réel avec la version modélisé du protocole AODV**
Taux de pertes	3.05 %	3.50 %
Délai moyen de bout en bout	5.32 ms	5.49 ms
Délai moyen de rétablissement de route	1.94 ms	2.08 ms
Stabilité d'une route	18.53 s	14.34 s
Overhead	501 ko (0.034 %)	552 ko (0.05 %)

Tableau 5.9 : Synthèse des résultats obtenus en émulation et en test réel

5.4 Validation des fonctions de sécurité du protocole SUAP

Dans cette section, nous montrerons la validation du mécanisme d'authentification et d'intégrité des messages du protocole SUAP. Cette validation a été menée à la fois en

émulation et en environnement réel. Dans un premier temps, nous nous focaliserons sur l'étude par émulation.

5.4.1 Validation de la partition de sécurisation (authentification des messages) en environnement émulé

5.4.1.1 Paramètres d'émulation

Notre objectif est de pouvoir évaluer les propriétés de sécurité du protocole SUAP qui n'ont pas été vérifiées par l'outil AVISPA et l'outil de vérification interne de Simulink et Stateflow de l'outil Mathworks. Pour ce faire, la topologie illustrée par la figure 5.11 est considérée pour simuler un attaquant *blackhole* et la figure 5.12 pour deux attaquants *blackhole*. Nous évaluerons les performances de sécurité du protocole SUAP pour ces deux configurations.

Ces deux modes d'attaques de l'attaque *blackhole* nous permettent de valider les mécanismes d'authentification et d'intégrité des messages du protocole SUAP. C'est-à-dire que chaque paquet généré par n'importe quel nœud que ce soit les attaquants ou les nœuds légitimes doit être authentifié avant d'être analysé et possiblement transmis à ses voisins. Pour permettre une bonne compréhension de ce chapitre, nous rappelons ci-dessous le descriptif de l'attaque *blackhole* en nous basant sur les deux configurations de test que nous avons considéré ci-dessous.

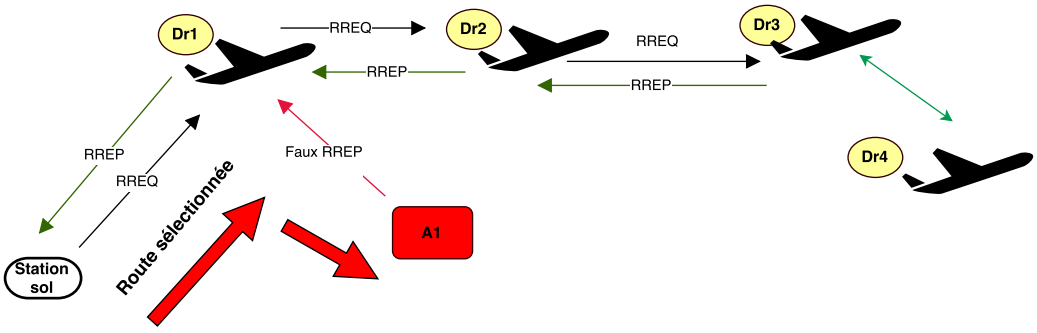

FIGURE 5.11 : Illustration de l'attaque *blackhole*

Dans la première configuration (illustrée par la figure 5.11), nous utilisons 4 nœuds légitimes. Dans un premier temps, la station sol cherche une route vers le drone Dr4 pour envoyer de la donnée. Ce paquet est diffusé dans tout le réseau jusqu'à ce qui atteigne le nœud Dr3 qui connait une route vers le nœud Dr4. Le nœud Dr3 répond alors avec un paquet réponse indiquant qu'il connait une route vers la destination demandée. En parallèle, le nœud attaquant A1 crée un faux paquet RREP et l'envoie au nœud Dr1 avec une meilleure métrique (par exemple, le nombre de sauts permettant d'atteindre la

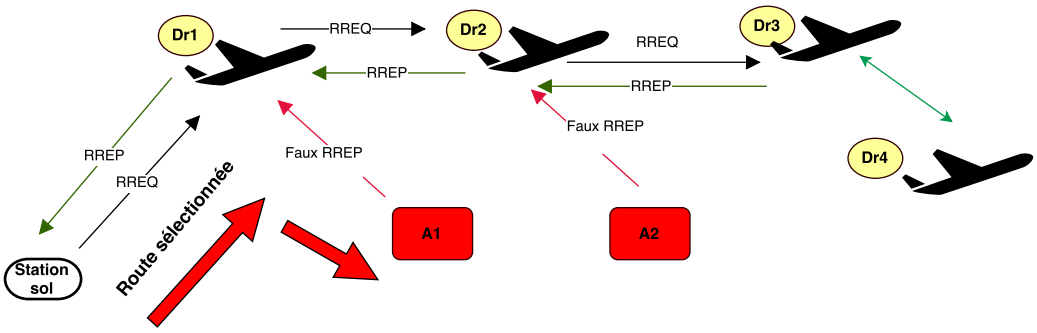

FIGURE 5.12 : Deux attaquants *blackhole*

destination) que le paquet RREP générée par le nœud Dr2.

Si les paquets ne sont pas authentifiés, la station sol va acheminer ces données vers l'attaquant. Pour implémenter cette attaque, nous avons appliqué une version d'attaque *blackhole* sur le nœud A1 qui va rejeter tous les paquets de données qu'il reçoit sur son interface. Nous tenons à préciser que nous aurions pu implémenter une autre variante plus sophistiquée de l'attaque *blackhole* qui est l'attaque *greyhole* qui consiste à sélectionner les paquets à jeter, mais l'attaque *blackhole* dans sa version originale permettait déjà d'évaluer notre mécanisme de sécurité. Sur cette configuration, le nœud Dr4 n'aura donc jamais les données applicatives qu'il est censé recevoir.

Par ailleurs, dans une deuxième configuration, nous avons ajouté un deuxième attaquant illustré par la figure 5.12. Ces deux nœuds attaquants sont mobiles dans le réseau. L'intérêt d'activer leur mobilité est qu'il est intéressant d'observer comment les attaquants interagissent directement avec les nœuds légitimes pour mesurer les performances de notre protocole par rapport au nombre d'attaquants. Compte tenu de la faible densité du réseau, le fait de simuler des attaquants immobiles n'aurait pas suffi pour valider efficacement les mécanismes d'authentification.

5.4.1.2 Métriques considérées

Pour montrer les performances réseau du protocole SUAP en cas d'attaque, nous considérons les métriques suivantes :

— Taux de livraison pour mesurer le nombre de paquets réceptionnés par une destination comparée au paquet envoyé. Cette métrique donne une idée sur l'aptitude de protocole de routage à garder une route stable pour l'échange des paquets de données en présence d'attaquants ;

— Délai de bout en bout entre les nœud source et destination ;

— *Overhead* des mécanismes de sécurité pour d'évaluer le trafic supplémentaire induit

par l'utilisation des différents mécanismes de sécurité. Ce coût d'utilisation concerne les mécanismes d'authentification et d'intégrité. Il permet d'évaluer si ces mécanismes ne portent pas atteinte à la performance du réseau ;

— Connectivité entre les différents nœuds du réseau. Cette mesure nous permet de quantifier la stabilité d'une route en présence d'attaquants.

Il est important de noter que chaque résultat présenté ici résulte de plusieurs répétitions du même scénario de simulation pour nous permettre de déduire la valeur moyenne et sa reproductibilité. Nous comparons également le protocole SUAP avec sa version sans sécurité (c'est-à-dire, la partition de routage du protocole SUAP) que l'on a validée précédemment. Nous avons décidé d'utiliser le terme AODV pour désigner la partition de routage du protocole SUAP.

5.4.1.3 Taux de livraison des paquets

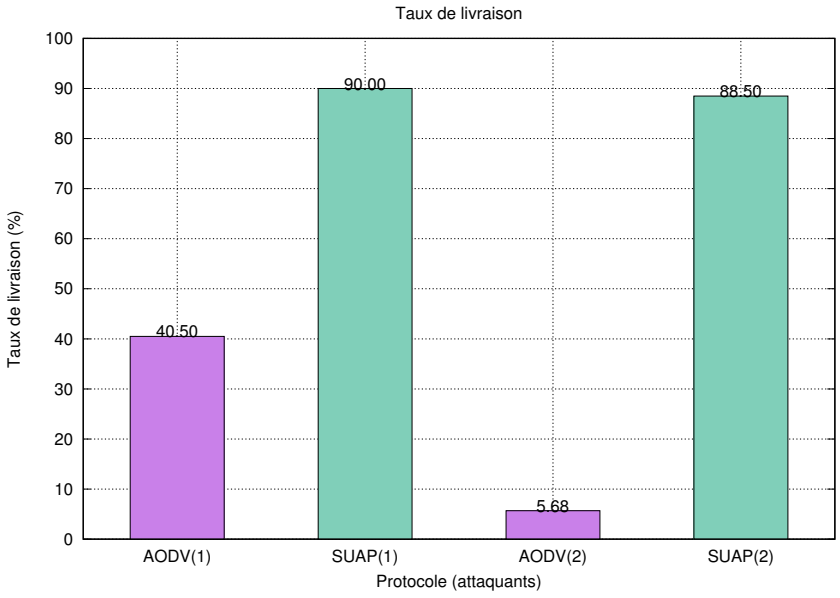

FIGURE 5.13 : Taux de livraison pour les deux protocoles avec un et deux attaquants *blackhole*

La figure 5.13 présente les taux de livraison des paquets pour le protocole SUAP et AODV et avec la présence d'un et de deux attaquants *blackhole*. Le résultat montre une différence remarquable à la fois lorsqu'il y a un attaquant et deux attaquants. Ces résultats sont justifiés par plusieurs raisons :

1. Premièrement, en cas d'utilisation du protocole AODV, la majorité des paquets de données n'arrivent pas au nœud destination. Ils sont rejetés par les attaquants *blackhole*.

2. Ensuite, bien que les attaquants jettent les paquets de données, le taux de livraison n'est pas complètement nuls car l'utilisation de modèles de mobilité réelle inclus dans le simulateur fait que les nœuds du réseau y compris les attaquants peuvent sortir de la couverture des nœuds légitimes. Lorsqu'ils n'interviennent plus dans le réseau pendant un temps donné, des paquets de données arrivent à être acheminés par la destination. Dès qu'ils reviennent dans le réseau, les paquets commencent à nouveau à être supprimés.

3. Il est aussi important de noter qu'avec deux attaquants, nous remarquons une faible valeur du taux de livraison de donnée, car ici les nœuds *blackhole* sont mobiles et interagissent avec les drones en mouvement. À cause de la faible densité du réseau, le nombre d'attaquants à un impact important sur le taux de livraison.

4. En ce qui concerne le protocole SUAP, le nombre d'attaquants n'a pas un impact important sur le taux de livraison des paquets. Chaque paquet envoyé est authentifié par le mécanisme d'authentification et d'intégrité. Les faux paquets sont donc ignorés et les paquets de données ne sont acheminés que par la route légitime. Cette valeur n'atteint pas toutefois le 100 % à cause de la mobilité des nœuds qui peut entrainer dans certains cas, des ruptures de liens temporaires.

5.4.1.4 Délai de bout en bout

Le délai de bout en bout entre la station sol et le drone Dr4 est illustré par la figure 5.14. Ce délai tend vers une constante pour chaque protocole. Ici, il est important de remarquer que ce qui est important n'est pas le délai de bout en bout pour les paquets de routage AODV, mais pour le routage sécurisé. Nous apercevons que le délai que met SUAP est légèrement supérieur par rapport au protocole de routage normal. Cette légère augmentation est justifiée par les délais de signature et de vérification de signature. Ces valeurs sont acceptables en tenant compte de l'application temps réel considéré. Nous pouvons donc dire l'authentification des paquets est assurer tout en ayant des valeurs acceptables pour ce métrique. En effet, un délai de moins de 10 ms pour les flux vidéo qui doivent être acheminés est acceptable au regard de la mission visée et du cahier des charges que nous nous sommes fixés.

Il est important de préciser que ces valeurs dépendent des algorithmes cryptographiques utilisés et notamment de la taille des clés. Dans notre implémentation, nous avons utilisés du RSA-512 pour la signature, et du SHA-256 pour le hachage. Ces algorithmes ont été choisis pour la performance du réseau comme préconisé dans [Zap08].

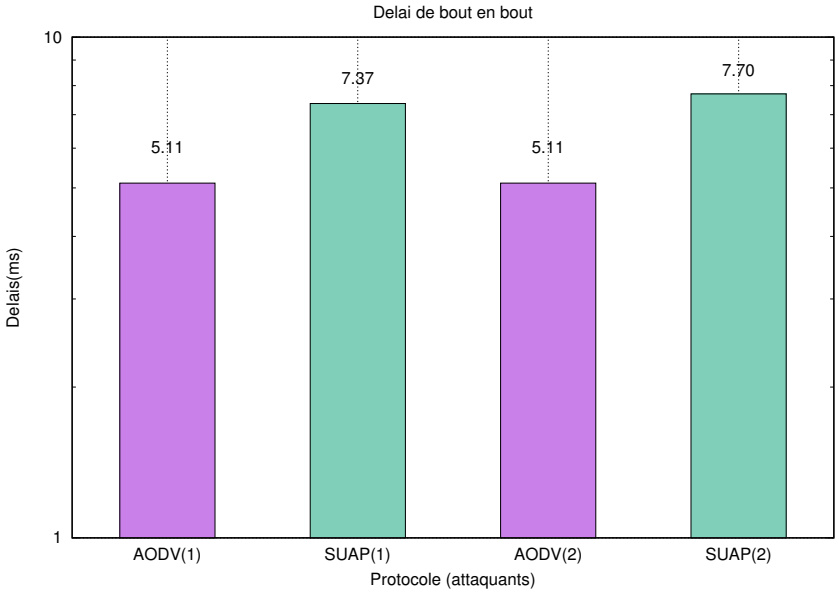

FIGURE 5.14 : Délai de bout en bout pour chaque protocole en variant le nombre d'attaquants

5.4.1.5 Overhead

Le tableau 5.10 présente le coût des paquets de sécurité par rapport au trafic de signalisation normal. C'est un paramètre important quant à la bande passante qui pourra être allouée aux applications. Moins le coût de sécurité est important, plus il laisse de place aux applications sur le médium de communication.

	AODV	SUAP
Paquets de signalisation	501 ko	2903 ko
% Trafic (octets)	0.034 %	0.46 %
% Trafic sécurité par rapport au trafic de routage (octets)		45 %

Tableau 5.10 : *Overhead* sur le test d'une heure

Au vu des résultats que nous obtenons, nous pouvons dire que le protocole SUAP consomme plus de ressources de bande passante que le protocole AODV à cause des entêtes de sécurité utilisée. Toutefois, ces trafics supplémentaires ne représentent que 0.46 % du trafic total échangé. Ce taux de 0,46 % est acceptable au regard des fonctions de sécurité qu'il apporte à l'application finale. D'autre part, dans une optique de la présence de nœuds malveillants, il n'y a pas d'alternative à cet overhead, car ne pas utiliser de sécurité se traduirait immédiatement par un taux de livraison de paquets extrêmement faible (cf. la

section 5.4.1.3 qui nuirait par contre à la réalisation de la mission de vidéo surveillance.

5.4.1.6 Connectivité

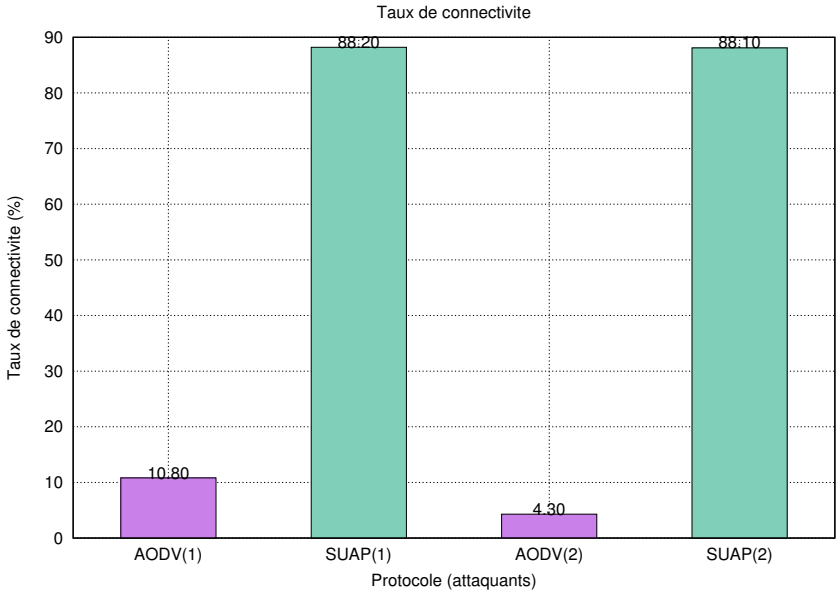

FIGURE 5.15 : Résultats de connectivité sur un test d'une heure

Les résultats que nous avons obtenus concernent la connectivité sont exposés dans la figure 5.15. Il indique le pourcentage de connectivité de la route en présence d'attaquants avec les deux protocoles. Ces résultats montrent que le protocole AODV souffre considérablement de l'effet de l'attaque *blackhole*. En effet, le nœud malveillant A1 fait partie de la route pendant la majorité de la communication. Par conséquent, la connectivité entre les nœuds du réseau est de l'ordre de 10 %. Il tend vers une valeur plus faible en présence d'un deuxième attaquant *blackhole*. En utilisant le protocole SUAP, nous apercevons que la route légitime est gardée pendant 88 % du temps. Cela nous indique les mécanismes d'authentification et d'intégrité utilisées permettent de garder l'intégrité du réseau sans passer par un nœud malveillant. Nous pouvons également constater que cette valeur de 88 % est statique et qu'elle n'est pas proportionnelle au nombre d'attaquants dans le réseau. Contrairement à cela, la connectivité du protocole AODV est inversement proportionnelle au nombre d'attaquants dans le réseau. Nous pouvons aussi constater que cette valeur de connectivité n'atteint pas les 100 %, car la mobilité des nœuds impacte l'état de connectivité entre deux nœuds voisins.

Par ailleurs, en comparant les résultats que l'on obtient pour cette métrique au résultat

d'émulation d'AODV et du test réel de la partition routage, nous observons un résultat qui reste dans le même ordre de grandeur (à savoir 88 %). Cela signifie que les briques de sécurité n'induisent pas une diminution significative en matière de connectivité dans le réseau UAANET en présence d'attaquants.

5.4.2 Validation de la partition de sécurisation (authentification des messages) en environnement réel

La topologie utilisée ici est représentée par la figure 5.16. Elle est constituée des 4 nœuds, dont trois drones DT18 et une station sol. Ensuite, nous avons ajouté une deuxième station sol qui va jouer le rôle d'un attaquant *blackhole*. Pour réaliser ce test, il était nécessaire de créer un attaquant proche du nœud émetteur Dr3 pour vérifier que la route légitime passant par les nœuds Dr1 et Dr2 est choisie pendant toute la durée du test. Nous avons donc diminué la puissance de l'antenne du modem utilisée pour avoir une portée inférieure ou égale à 500 m. La station 4 (attaquant *blackhole*) quant à elle a une puissance d'émission maximum pouvant aller jusqu'à 1,5 km. Nous faisons donc en sorte que l'attaquant puisse contacter tous les nœuds et surtout le drone le plus lointain. Cette topologie nous permet ainsi de vérifier si le protocole SUAP permet de se prémunir de l'attaque *blackhole* en condition réelle.

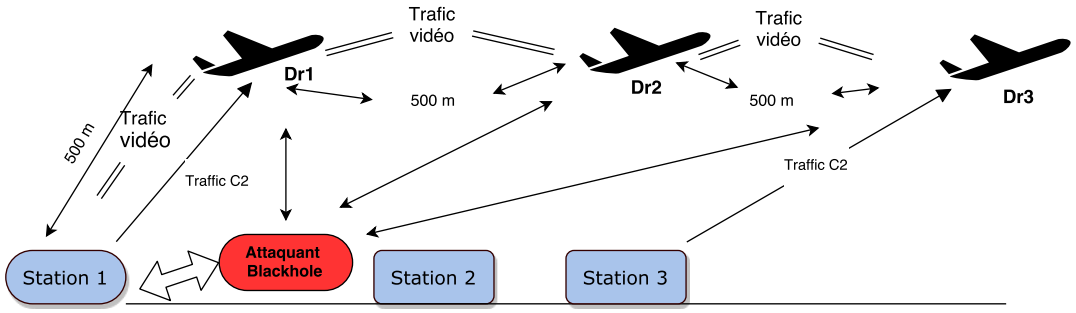

FIGURE 5.16 : Topologie considérée pour valider la partition de sécurisation en environnement réel

Il est important de préciser que l'on aurait pu également utiliser d'autres équipements pour jouer le rôle d'un attaquant. L'utilisation d'une station sol nous permet de s'affranchir de l'implémentation du code de l'attaque *blackhole* dans une autre architecture machine. Les précédentes configurations restent inchangées. Les trafics échangés entre les nœuds sont résumés dans le tableau 5.11.

La figure 5.17 illustre le contenu du champ TTL (Time to Live) capturé sur chacun des nœuds. Elle indique qu'entre 7 et 23 minutes du test, les trois drones sont alignés en vol et communiquent en mode ad hoc. Le comportement du protocole SUAP est donc étudié

Type	Source —— Destination	Taille du paquet	Débit d'échange
Tick	Station1 —— Dr1 Station2 —— Dr2 Station3 —— Dr3	64 octets	1 paquet/sec
Georef	Dr1 —— Station1 Dr2 —— Station2 Dr3 —— Station3	80 octets	3 paquets/sec
Commande	Station 1 —— Dr1 Station 2 —— Dr2 Station 3 —— Dr3	80 octets	1 paquet/sec
Vidéo	Dr3 ——— Station1	1400 octets	25 datagrames UDP/sec width=720,height=576

Tableau 5.11 : Différents flux échangés lors du test en vol

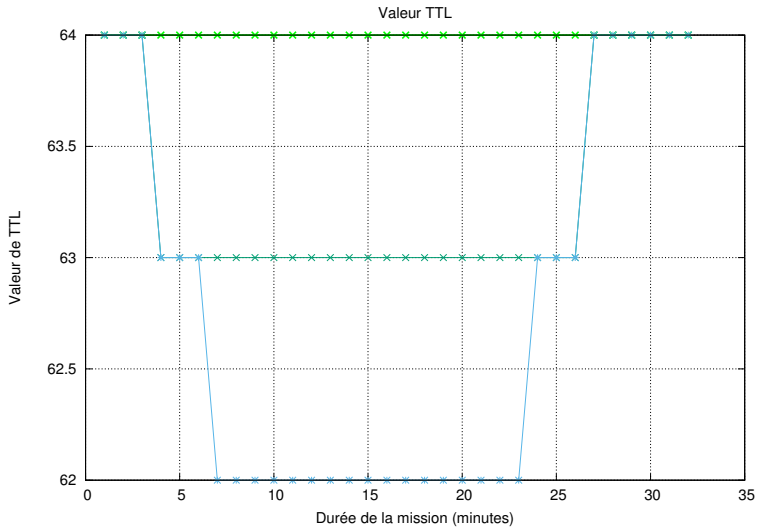

FIGURE 5.17 : Valeur TTL du drone Dr1 (ttl=62), Dr2 (ttl=63) et Dr3 (ttl=64)

entre cette période.

5.4.2.1 Taux de pertes des paquets

Le taux de pertes est présenté par le tableau 5.12. Comme nous pouvons le voir, les pertes sont surtout concentrées sur la connexion entre les trois drones. En effet, dû aux mobilités des drones et aux différents schémas de déplacement des trois drones durant le test, des liens sont perdus puis retrouvés au cours de la mission. Au total, le taux de pertes observé est de 5.57 % pour les paquets de données qui sont envoyés depuis le Dr3 vers le nœud station 1. Ce taux de pertes est acceptable en le comparant aux résultats obtenus en

émulation.

Source	Destination	taux de pertes
Station 1	Dr1	0.53
Dr1	Dr2	2.32
Dr2	Dr3	2.70
Totale (Station 1 — Dr3)		5.57

Tableau 5.12 : Taux de pertes (%) mesurés entre chaque nœud

5.4.2.2 Overhead

Les résultats de l'*overhead* sont montrés dans le tableau 5.13. Ces résultats sont également proches de celle que l'on a trouvée en émulation. La propriété de routage réactif fait que peu de paquets de trafic de signalisation sont échangés sur le réseau. Le protocole n'agit qu'en cas de demande d'établissement de route. À cause de la faible densité des nœuds, la vitesse de déplacement des nœuds n'a pas un impact direct sur cette valeur.

Source	Destination	Taille des paquets de contrôle (en octets)	Pourcentage par rapport au trafic total
Station 1	Dr1	89 760	0.075 %
Dr1	Dr2	153 653	0.093 %
Dr2	Dr3	153 321	0.0927 %
Total			0.270 %

Tableau 5.13 : *Overhead* sur le test de 16 minutes

5.4.2.3 Durée de vie moyenne d'une route

Délai	Valeur
Délai moyen	15.15 s
Délai maximum	20.44 s

Tableau 5.14 : Stabilité d'une route en présence d'un attaquant *blackhole*

Le tableau 5.14 et la figure 5.18 illustrent les résultats que l'on a obtenus pour la durée de vie moyenne d'une route entre la station 1 et Dr3. Ce résultat est à comparer avec la figure 5.4 et le tableau 5.5 qui illustrent la stabilité d'une route avec le protocole AODV (dans le cas où il n'y a pas d'attaquant dans le réseau). Ce que l'on peut conclure est que la durée de vie moyenne d'une route reste dans le même ordre de grandeur avec et sans sécurité. Cela nous indique que notre protocole de routage se comporte comme le protocole AODV en termes de performances et que l'ajout des mécanismes de sécurité ne nuit pas la performance réseau tout en garantissant l'absence d'attaque dans cette configuration.

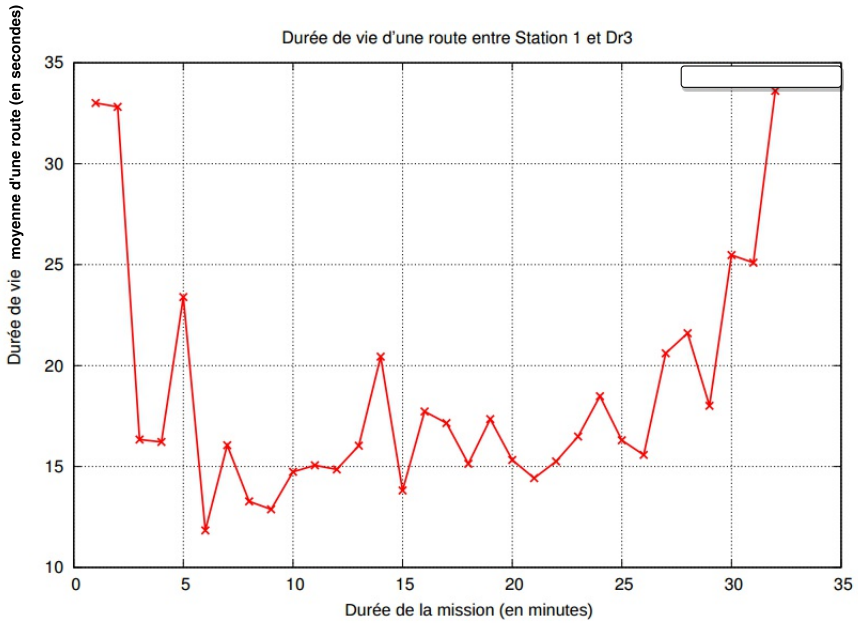

FIGURE 5.18 : Stabilité d'une route avec la présence d'un attaquant *blackhole*

Pour résoudre la perte de connectivité entre un ou plusieurs nœuds conduisant à la perte de la route, le délai de rétablissement d'une route est illustré dans le tableau 5.15. Nous avons un délai moyen de 2.23 milliseconde qui permet de rattraper les pertes de connectivité fréquentes sur le réseau. En faisant le lien avec la valeur de ré-établissement de route que l'on a obtenue durant le test réel du protocole AODV, nous apercevons le même ordre de grandeur (2.23 ms contre 2.08 ms).

Délai	valeur
Délai moyen	2.23 ms
Délai maximum	5.79 ms

Tableau 5.15 : Délai de rétablissement de route après une déconnexion

5.4.2.4 Pourcentage de routes établies entre station1 et Dr3

Pour cette métrique, l'objectif est d'évaluer le nombre d'établissements de route passant par la route légitime (composée des nœuds Dr3, Dr2, Dr1, station 1) et passant par l'attaquant *blackhole*. Le but est de vérifier si les paquets provenant du nœud attaquant conduisent à l'établissement d'une route. Les résultats obtenus sont résumés par la figure 5.19. La ligne rouge représente les paquets du drone Dr2 et la ligne verte représente les paquets envoyés par l'attaquant. Pour calculer cette métrique, l'adresse IP de l'attaquant (dans la trace mesurée) a été filtrée sur le nœud Dr3, et nous vérifions s'il a été accepté ou pas comme instigateur d'une route.

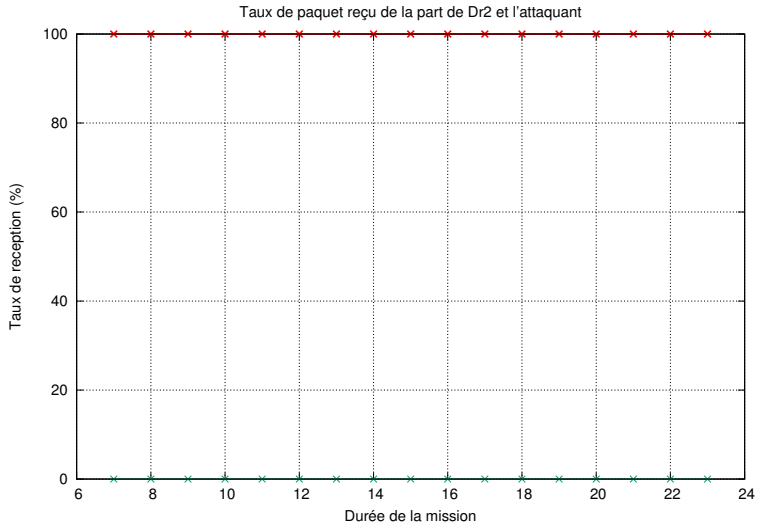

FIGURE 5.19 : Pourcentage des paquets de contrôle conduisant à un établissement de route venant de Dr2 et de l'attaquant mesuré sur l'interface du nœud Dr3

On remarque sur cette figure que 100 % des paquets conduisant à l'établissement d'une route proviennent du nœud Dr2 qui est son voisin légitime. Le nœud Dr3 ne traite donc aucun paquet non authentifié du nœud attaquant. Cela nous indique que les algorithmes d'authentification des messages utilisés sont fonctionnels.

5.4.2.5 Délai de bout en bout

Délai de bout en bout pour l'échange des trafics de signalisation	**Valeur**	**Délai de bout en bout pour l'échange des trafics de charge utile**	**Valeur**
Délai moyen	7.43 ms	Délai moyen	9.2 ms
Délai maximum	100 ms	Délai maximum	104 ms

Tableau 5.16 : Délai de bout en bout pour les paquets de contrôle et les trafics temps réel

Le délai de bout en bout pour chaque trafic est présenté dans le tableau 5.16. Cette valeur est généralement faible (inférieures à 10 ms en moyenne) et n'impacte pas négativement l'échange des trafics temps réel. Elle est toutefois supérieure au résultat que l'on a obtenu pour le protocole de routage AODV. Ceci est dû au temps non négligeable que met la bibliothèque de cryptographie LIBGCRYPT [1] que l'on utilise pour la signature et le hachage et leur vérification. Pour illustrer la proportion de paquets qui dépasse les 10 ms,

1. Pour plus d'informations : https ://www.gnupg.org/software/libgcrypt/index.html

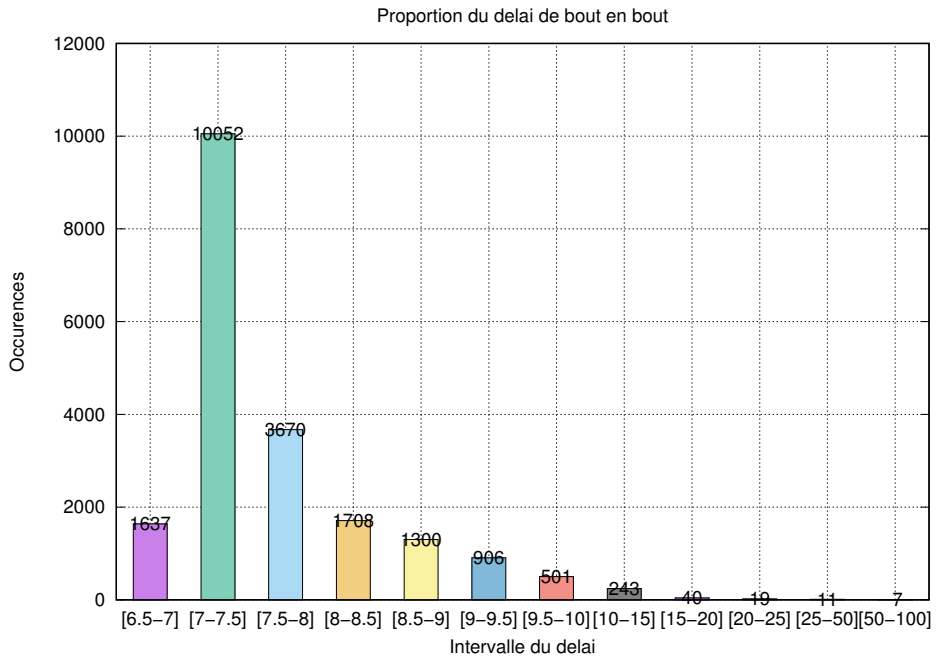

FIGURE 5.20 : Répartition des valeurs délai de bout en bout au cours de la mission

la figure 5.20 montre l'évolution du délai de bout en bout au cours de la mission. Nous apercevons que les valeurs que l'on a obtenues se concentrent dans l'intervalle [7, 7.5]. Le délai supérieur à 100 ms ne représente que 7 occurrences durant la mission.

5.4.2.6 Évalutation du protocole AODV en environnement réel en présence d'un attaquant *blackhole*

Pour mieux situer les résultats que nous avons obtenus pour les mécanismes d'authentification des messages, le protocole AODV a été testé en environnement réel en présence d'un attaquant *blackhole*. Ce test a été effectué avec la même configuration que le test du protocole SUAP présenté dans le tableau 5.11 et la figure 5.16. Les résultats obtenus sont synthétisés dans le tableau 5.17. Dans ce tableau, nous avons illustré les résultats obtenus avec les résultats du protocole SUAP présenté précédemment. Il est à noter que seules les métriques permettant de mettre en avant la différence entre AODV et SUAP ont été retenues.

Il est important de préciser que la mobilité des drones au cours de ce test est différente de celle utilisée pour l'évaluation du protocole SUAP. Cela est dû au fait qu'il était difficile de reproduire la même mobilité en environnement réel sur deux tests différents. Les

Paramètre	**Test réel du protocole AODV en présence d'un attaquant *blackhole***	**Test réel du protocole SUAP en présence d'un attaquant *blackhole***
Taux de perte	98.20 %	5.57 %
Pourcentage de routes établies entre Station 1 et Dr3	1.76 %	100 %
Overhead	210 ko	397ko

Tableau 5.17 : Synthèse des résultats obtenus pour l'évaluation en environnement réel du protocole AODV en présence d'un attaquant *blackhole*

résultats obtenus montrent toutefois la faiblesse du protocole AODV en présence de l'attaque *blackhole* en environnement réel. Ces résultats peuvent être expliqués de la manière suivante :

— pour le taux de perte, la majorité des paquets de données sont perdus, car à chaque fois qu'une demande de route est envoyée, l'attaquant *blackhole* intervient en publiant de faux paquets amenant à des routes erronées.

— pour le pourcentage de routes établies entre la station 1 et le nœud Dr3, elle n'est que de 1,7 % durant toute la mission, car les routes établies incluent le plus souvent, l'attaquant *blackhole*.

— pour l'*overhead*, la valeur obtenue pour le protocole AODV est inférieure au résultat que l'on a obtenu pour le protocole SUAP (cf. les résultats présentés dans le tableau 5.13). Ces trafics supplémentaires représentent le coût des mécanismes de sécurité utilisés. Toutefois, ce résultat n'impacte pas négativement l'échange des données comme nous l'avons présenté précédemment (cf. les résultats de délai de bout en bout présentés dans le Tableau 5.16).

Discussions

Dans cette section, nous avons présenté l'étude de validation de la partition d'authentification des messages du protocole SUAP. Les résultats que l'on a obtenus à la fois en émulation et en environnement réel nous montrent une bonne adaptation du protocole SUAP face à l'attaque *blackhole*. L'attaque *blackhole* a été mise en avant, car elle permet de vérifier que chaque nœud du réseau arrive bien à vérifier le niveau de sécurité d'un message avant de le traiter. Nous avons remarqué que l'attaque *blackhole* n'a pas une influence directe sur la performance du protocole SUAP contrairement au protocole de routage AODV. Nous pouvons donc conclure que notre implémentation développée par une approche orientée modèle produit le fonctionnement haut niveau spécifié dans le cahier des charges. Ces premiers résultats sont positifs, mais ne permettent pas de valider la totalité des fonctionnalités de sécurité du protocole SUAP. Ainsi, dans la partie sui-

vante, nous étudierons la validation du mécanisme de détection de l'attaque *wormhole* que nous avons proposé dans le chapitre 4 et qui est un des composants importants de cette architecture de sécurité.

5.4.3 Validation du mécanisme de détection contre l'attaque *wormhole*

Dans cette partie, nous analyserons la précision de détection de l'attaque *wormhole* par le protocole SUAP. Cette étude de validation a été faite exclusivement en émulation. Pour valider cette partition, nous avons considéré une topologie à 5 nœuds légitimes et deux attaquants comme le montre la figure 5.21. Il est important de préciser que nous avons fait le choix de l'émulation pour cette phase de validation, car nous ne disposions pas d'assez de nœuds mobiles et de systèmes embarqués compatibles avec les moyens de communication du DT18 pour envisager une validation en environnement réel de ce mécanisme de sécurité. Cette piste future de validation en environnement réel sera abordée dans le chapitre conclusions et perspectives.

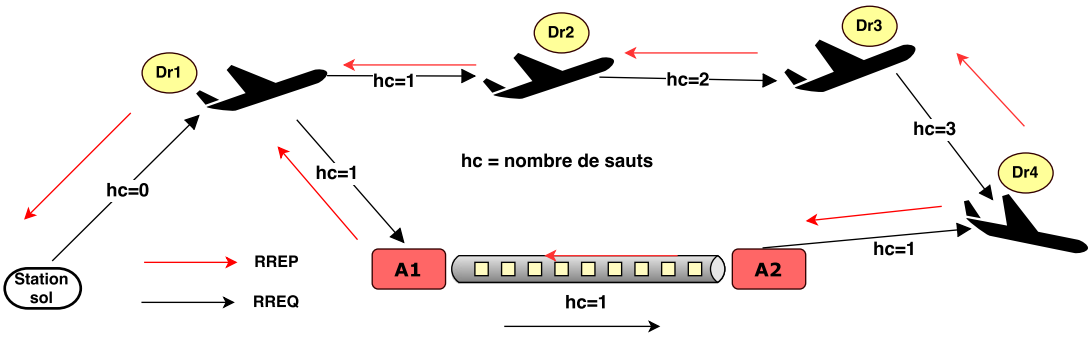

FIGURE 5.21 : Topologie de validation de l'attaque *wormhole*

Les paramètres de ce scénario sont présentés sur le tableau 5.18

Paramètres	**Valeur**
Nombre de nœuds légitimes	5 (4 drones et une station sol)
Mobilité	Rejoue de mobilité réelle
Protocole de routage	SUAP et AODV
Protocole MAC	802.11
Portée radio	100m
Durée de simulation	600 s
Capacité du canal	54 Mbit/s

Tableau 5.18 : Paramètres de l'évaluation pour la validation du mécanisme de détection de l'attaque *wormhole*

Nous avons comparé le protocole SUAP avec le protocole AODV pour étudier l'influence de l'attaque. Pour cela, nous avons vérifié que la route passant par le tunnel entre A1

Type	Source→Destination	Taille du paquet	Fréquence
Tick	1→2,1→3,1→4	64 octets	1.0 paquet/s
Georef	2→1,3→1, 4→1	64 octets	1.8 paquet/s
Command	1→2,1→3, 1→4	64 octets	0.034 paquet/s
Vidéo	2→1,3→1,4→1	4 Mbit/s	

Tableau 5.19 : Trafics générés : 1=Dr1, 2=Dr2, 3=Dr3, 4=Dr4

et A2 n'est pas utilisée pour transmettre des données. Pour permettre de conclure sur la pertinence de notre mécanisme, nous avons considéré deux métriques :

1. d'une part, la quantité de données (qui a été envoyée depuis le drone Dr4) qui est réceptionnée par la station sol (en passant sur les deux routes alternatives) est mesurée. Ainsi, la quantité de données passant par le lien *wormhole* et la quantité de données passant par lien légitime sont comparées.
2. d'autre part, le nombre de routes établies passant par le lien *wormhole* et le nombre de routes passant par le lien légitime ont été évalués.

Ces deux métriques sont complémentaires et permettent de conclure sur la pertinence de notre mécanisme de sécurité contre l'attaque *wormhole*.

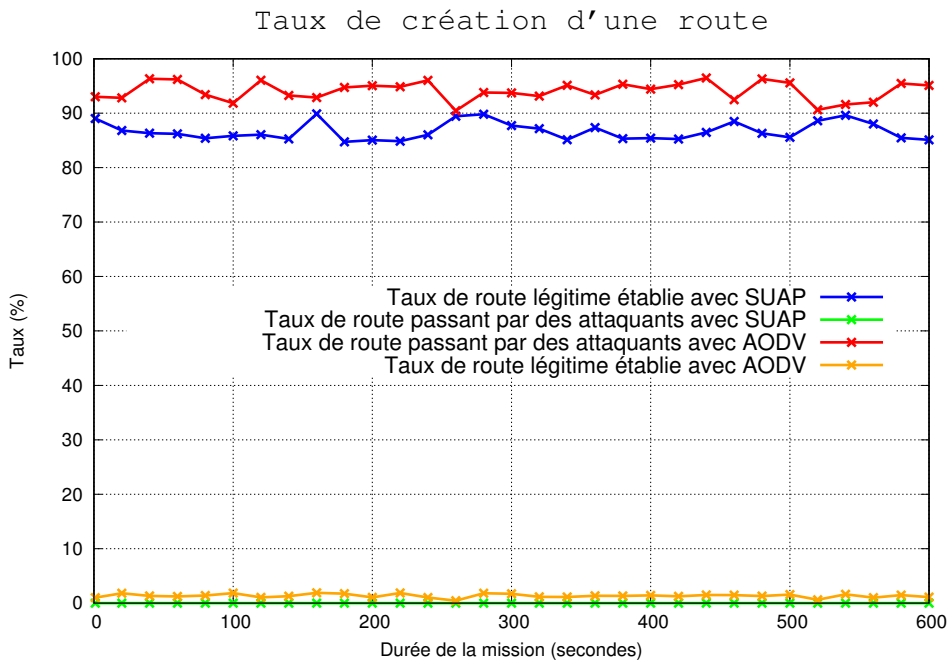

FIGURE 5.22 : Taux de création passant par les nœuds légitimes et les attaquants

Pour faciliter cette validation, nous avons décidé de simuler deux attaquants immobiles avec A1 proche de la station sol et A2 proche du drone le plus loin Dr4. Ce choix a été fait pour mettre les attaquants proches des nœuds cibles pour mieux évaluer l'impact de l'attaque.

Source	**Quantité de trafic de signalisation généré**	**Trafic vidéo généré**	**Destination**	**Quantité de trafic de signalisation réceptionné**	**Quantité de trafic de données réceptionné**
Dr4	146783 octets	23 02 289 ko	Dr3	146783 octets	23 02 111 ko
Dr3	140782 octets	23 02 111 ko	Dr2	140782 octets	23 00 011 ko
Dr2	142345 octets	23 00 011 ko	Dr1	142345 octets	22 98 245 ko
Dr1	130011 octets	22 98 245 ko	Station sol	140011 octets	22 97 988 ko
Dr4	146783 octets	23 02 289 ko	A1	146783 octets	**0 octet**
A1	146783 octets	**0 octet**	A2	146783 octets	**0 octet**

Tableau 5.20 : Synthèse des données envoyées et réceptionnées entre la communication Dr4 vers Station sol en utilisant le protocole SUAP

Sur les deux tableaux 5.20 et 5.21, nous présentons respectivement la quantité d'information échangée par les nœuds légitimes et par les attaquants. Ce que l'on peut remarquer c'est qu'en utilisant le protocole SUAP, le tunnel *wormhole* ne véhicule que des informations de routage des nœuds cibles. Cette réception des paquets de routage par les attaquants n'est pas importante, car ces messages ne contiennent pas d'informations secrètes relatives à la mission.

Nous remarquons également qu'aucune information de vidéo ne passe entre A1 et A2. Ici, la détection est faite par la station sol qui constate une anomalie entre sa distance relative avec le nœud Dr4 et le nombre de sauts dans le paquet de découverte de voisin qu'il reçoit de ce nœud. Nous rappelons qu'ici les paquets Hello sont protégés par un algorithme de *geographical leashes* qui compare le nombre de sauts et la distance relative entre voisins. Les paquets requêtes et réponses sont quant à eux protégées par une fonction de hachage. Ces résultats montrent donc la performance du protocole SUAP à choisir la route légitime malgré le fait qu'une route avec meilleure métrique est proposée sur le tunnel *wormhole*.

En comparant ce résultat au tableau 5.21 qui indique la quantité de données réceptionnée par les deux attaquants en utilisant le protocole AODV, nous constatons que le tunnel arrive à faire transiter de la vidéo provenant du nœud Dr4. Cette fois, les autres nœuds légitimes n'échangent que des trafics de signalisation et ne reçoivent qu'une partie infime de données utile (qui représente 0.18 % du trafic total). Ceci est dû au fait que la route qui passe par les deux attaquants est constamment choisie à cause d'une meilleure métrique en nombre de sauts. La valeur 0.18 % est expliquée par le fait que lorsque le nœud Dr4 est hors portée du nœud A2 à cause de sa mobilité, la route légitime est choisie pendant une courte durée pour véhiculer de la vidéo qui ne représente que 0.18 % de la quantité totale.

La figure 5.23 résume les résultats de taux de livraison de données sur les deux routes

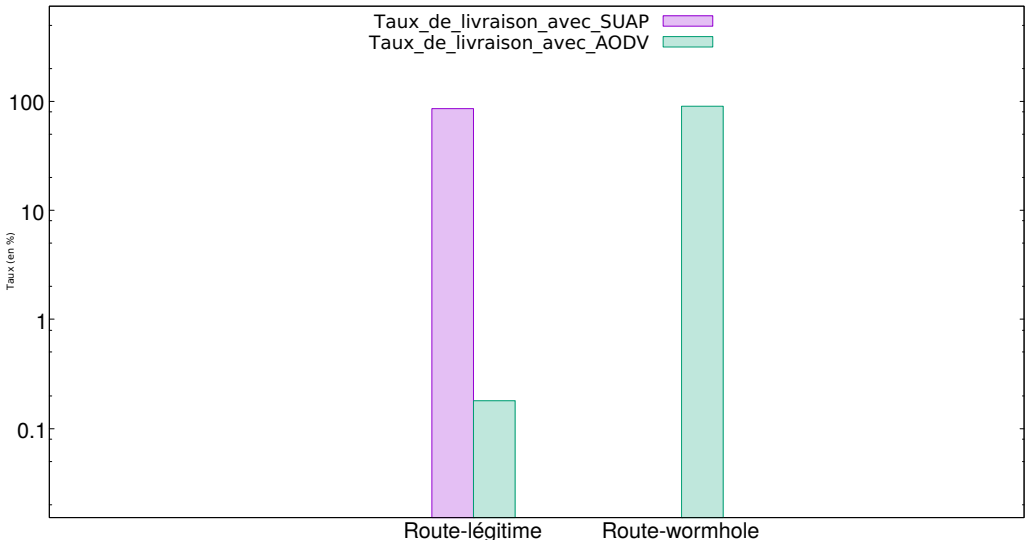

FIGURE 5.23 : Taux de livraison des paquets en présence d'une attaque *wormhole*

(légitimes et passant par les attaquants) illustrée dans les tableaux 5.20 et 5.21. Elle nous montre qu'avec le protocole SUAP, seule la route légitime est choisie pour véhiculer les paquets de données. Par l'intermédiaire de ces résultats, nous pouvons donc conclure que le protocole SUAP est robuste à l'attaque *wormhole*.

Source	**Quantité de trafic de signalisation généré**	**Trafic vidéo généré**	**Destination**	**Quantité de trafic de signalisation réceptionné**	**Quantité de trafic de données réceptionné**
Dr4	146783 octets	23 02 289 ko	Dr3	146783 octets	13450 ko
Dr3	140782 octets	13450 ko	Dr2	140782 octets	13450 ko
Dr2	142345 octets	13450 ko	Dr1	142345 octets	13450 ko
Dr1	130011 octets	13450 ko	Station sol	140011 octets	13450 ko
Dr4	146783 octets	23 02 289 ko	A1	146783 octets	22 98 839 ko
A1	146783 octets	22 98 839 ko	A2	146783 octets	22 98 839 ko

Tableau 5.21 : Synthèse des données envoyées et réceptionnées entre la communication Dr4 vers Station sol en utilisant le protocole AODV (sans sécurité)

Pour mieux situer ce résultat, le taux de nombre de routes établies sur la route légitime (Dr4, Dr3, Dr2, Dr1 et la station sol), et le taux de nombre de routes établies sur la route passant par le tunnel *wormhole* (Dr4, A1, A2 et la station sol) ont été mesurés. La figure 5.22 nous montre que le taux de création sur la route légitime est environ de 85 % pendant toute la durée de simulation tandis que pour le tunnel *wormhole*, elle est de 0 %. Le taux de création de routes avec le protocole SUAP n'atteint pas également les 100 %, car ici les mobilités des nœuds entrent en jeu en créant des fluctuations de lien entre chaque voisin. Ces valeurs sont à comparer avec le résultat qu'on a obtenu avec le protocole AODV que l'on illustre également sur la même figure. Nous remarquons que sans

le mécanisme de détection, le protocole trouve en moyenne 90 % sur la route *wormhole*. Ce taux est meilleur que le taux proposé par SUAP, car les deux attaquants sont immobiles et donc la fluctuation des liens sur la route est moindre que sur la route légitime. Ceci montre la performance de l'attaque *wormhole* qui peut proposer une meilleure performance suivant la topologie du réseau et suivant la métrique choisie. Ainsi, sans notre protocole de détection, les flux vidéos sont acheminés vers le tunnel *wormhole*.

Par ailleurs, avec l'utilisation du protocole AODV, le taux de création de routes passant par la route légitime est significativement moindre et proche de 0 %. Cette valeur n'atteint pas toutefois le 0 %, car à cause de la mobilité du nœud Dr4, il peut être hors portée de l'attaquant A2 pendant une courte durée et peut donc communiquer avec son voisin légitime Dr3. Toutefois, cet échange est négligeable, car le taux de création de routes sur le lien (Dr4, Dr3, Dr2, Dr1, station sol) est environ 0,18 % sur toute la communication.

Discussions

Dans cette sous-section, nous avons présenté la validation du mécanisme SUAP contre l'attaque *wormhole*. Les résultats obtenus montrent l'aptitude de notre approche à garder la route passant par les nœuds légitimes actifs et ignorent le chemin passant par le tunnel *wormhole*. Cela permet ainsi de garder un pourcentage de livraison de paquets de l'ordre de 85 % et qui est dans l'ordre de grandeur de performance que l'on a obtenu avec AODV en simulation (88 %). Nous remarquons également qu'en l'absence de l'approche de détection de l'attaque *wormhole*, le pourcentage de nombre de routes établies passant par le nœud légitime est très largement inférieur. Ces résultats montrent donc l'efficacité du protocole SUAP contre l'attaque *wormhole* comme spécifié dans le chapitre 4.

5.5 Conclusions

Dans ce chapitre, nous avons présenté la validation des partitions (routage et sécurisé) du protocole SUAP. Cette validation par évaluation de performance a été réalisée d'une part à travers un outil qui combine l'émulation (utilisation des machines virtuelles) et la simulation (l'outil OMNET++), et d'autre part en environnement réel en utilisant les drones DT18 de Delair-Tech. Rappelons que le protocole SUAP a été conçu en utilisant la méthodologie orientée modèle présente dans le chapitre 3. Dès lors, pour rester dans l'esprit de cette méthode de prototypage rapide, nous avons choisi d'utiliser ces outils d'expérimentation pour n'avoir qu'une version de code source du protocole SUAP qui puisse s'exécuter à la fois sur les machines virtuelles, mais aussi sur les systèmes embarqués finaux. En y insérant des traces réelles de déplacement des drones et en ajoutant un système Linux proche de celui utilisé par les drones, nous avons pu créer des scénarios concrets et réalistes. Les résultats que nous avons obtenus pour la validation des fonctions de routage montrent que

notre algorithme développé par l'approche orientée modèle répond aux différents services demandés par l'environnement ad hoc. Il offre des résultats acceptables pour le taux de livraison, le délai de bout en bout, l'overhead, le délai de rétablissement d'une route et la stabilité d'une route. En ce sens, nous pouvons dire que la partition de routage de notre architecture de communication répond au cahier des charges spécifié au début de la thèse.

Ensuite, nous avons également réalisé des tests de simulation et test réels pour valider les mécanismes de sécurité du protocole SUAP. Notre architecture de test est constituée de trois drones et de trois stations sol dont une station qui est partie prenante des nœuds de communication du réseau. Notre configuration de test a abouti à un système de communication composé d'une flotte de drones coopératifs pilotés par des stations sol. Dans le but de valider les performances du protocole SUAP, nous avons réalisé une campagne de test. Les résultats évoqués montrent que SUAP sécurise la communication entre les nœuds en assurant l'authentification des messages. Il permet également de détecter un tunnel *wormhole*. Il assure que seuls les nœuds légitimes transitent les paquets de données. De plus, il respecte la contrainte temporelle du trafic temps réel échangé en proposant des délais de rétablissement de route et de délai de bout en bout acceptable. Les performances du protocole SUAP sont amplement acceptables en comparaison avec les performances du protocole AODV présenté dans le chapitre précédent. Les valeurs obtenues permettent de déployer une architecture de communication sécurisée d'une flotte de drones.

Également, nous avons donc mis en évidence l'utilisation de la méthodologie orientée modèle en validant les partitions logicielles par des tests de performance et la conformité entre le cahier des charges et les différentes fonctionnalités du logiciel embarqué produit.

Chapitre 6

Conclusions et perspectives

Ce chapitre résume les différents travaux réalisés dans le cadre de cette thèse. On présente également les perspectives qui peuvent être traitées pour la continuité de cette thèse.

Contents

6.1 Travaux réalisés 191
6.1.1 Architecture de communication d'une flotte de drones 192
6.1.2 Besoin en sécurité dans un réseau UAANET 193
6.1.3 Application d'une méthodologie orientée modèle pour le développement du protocole SUAP 194
6.1.4 Validation fonctionnelle de la partition de routage et de sécurisation 194
6.2 Perspectives 195
6.2.1 Étude de performances complémentaires pour SUAP 196
6.2.2 Sécurité applicative des échanges en environnement UAANET . 196
6.2.3 Conception d'une infrastructure de gestion de clé adaptée au contexte UAANET 197

6.1 Travaux réalisés

Grâce au progrès de la miniaturisation des systèmes embarqués, les mini-drones sont maintenant pleinement opérationnels au niveau industriel et peuvent réaliser différents types de missions civiles. Ils offrent une solution à moindre coût pour les tâches complexes qui étaient auparavant réalisées par des avions ou des hélicoptères et qui se sont avérées plus coûteuses, mais aussi plus risquées pour les pilotes.

Pour permettre d'étendre la couverture d'exécution au cours de la mission ou pour contourner un obstacle, il est possible de déployer une flotte de drones (plutôt qu'un drone unique) dans laquelle plusieurs drones communiquent par échange de trafic de signalisation. Dans ce cas, un système de drones doit être créé. Un système de drones est l'ensemble des éléments permettant le vol d'un ou plusieurs aéronefs sans pilote. Il est constitué d'un essaim de drones, mais aussi des stations de contrôle matérielles et logicielles, des systèmes de communication, de positionnement et d'évitement de collision.

Le but de cette thèse était de proposer une architecture de communication sécurisée pour une flotte de drones. Pour atteindre cet objectif, notre étude s'est d'abord focalisée sur la mise en place d'une structure de communication fiable permettant l'échange des données en temps réel. Ensuite, dans un deuxième temps, nous nous sommes focalisés sur la proposition d'un protocole de routage sécurisé permettant l'authentification des paquets des nœuds légitimes de la flotte et qui puisse contrer un maximum d'attaques existantes dans la littérature des réseaux MANET. Ces systèmes de communication ont été développés par une méthodologie de prototypage rapide en utilisant une approche orientée modèle et des outils de vérification formelle. Ces outils méthodologiques ont été intégrés dans ce travail de thèse pour permettre de contribuer plus efficacement à la certification du système UAS

final.

6.1.1 Architecture de communication d'une flotte de drones

Le déploiement d'une flotte de drones pour des missions civiles coopératives nécessite une architecture de communication qui permet d'une part de faire communiquer les nœuds (les drones et la station sol) entre eux afin de véhiculer des trafics temps réel, et d'autre part de sécuriser la communication entre les nœuds communicants. Nous avons vu dans le chapitre 1 que plusieurs configurations d'architecture réseau peuvent être mises en place pour réaliser ce type d'opération. Parmi ces différentes possibilités, notre choix s'est tourné vers les réseaux MANET qui offre une solution garantissant une coordination aisée entre les drones de la flotte (la justification de ce choix est détaillée dans le chapitre 2). Le principe fondamental de cette architecture étant que chaque drone est responsable du maintien de la connectivité dans le réseau en échangeant des paquets de routage avec ses voisins.

Les réseaux ad hoc appliqués aux flottes de drones s'intitulent UAANET pour UAV Ad hoc Network. Ce type de réseau est considéré comme une sous-catégorie du réseau MANET. La mise en place de ce type de réseau nécessite un protocole de routage dynamique qui va prendre en compte la mobilité de chaque nœud et les contraintes de ressources pour mettre en place une route fiable entre chaque nœud. En effet, la communication entre les drones doit être réalisée avec une architecture de communication qui garantit la livraison des données ainsi que la prise en compte de la contrainte temporelle des trafics applicatifs. Pour sélectionner un protocole qui répond à notre cahier des charges, nous avons étudié dans ce travail de thèse l'ensemble des protocoles de routage existant dans le domaine des MANET. Dans ce contexte, nous rappelons que la proposition d'un protocole de routage pour le réseau UAANET doit dans ce cas considérer plusieurs paramètres qui sont l'absence totale d'une infrastructure fixe, la présence d'une topologie dynamique, la forte mobilité des nœuds, la contrainte temporelle forte des trafics applicatifs et la faible densité des nœuds. Par rapport à l'existant, notre choix préliminaire s'est tourné vers les protocoles réactifs, car nous nous intéressons à la réduction de la quantité des paquets de routage échangés étant donné le faible niveau de ressources disponible dans les réseaux UAANET. De plus, même s'il n'est pas préconisé d'utiliser une approche de routage réactif pour les applications temps réel, nos résultats qui sont détaillés dans le chapitre 4 ont montré que la faible densité des nœuds UAANET implique que le temps nécessaire pour établir une route est acceptable au regard de notre cahier des charges pour ce projet. Pour justifier notre choix, nous avons testé des protocoles de routage utilisé dans les résaux MANET grâce à un outil hybride d'émulation et de simulation qui incorpore l'environnement UAANET qui a été développé pour les besoins de ce travail de thèse. En effet, pour nous rapprocher des conditions réelles de l'environnement réel et pour permettre de tester en local notre

protocole de routage, nous avons créé au cours de la thèse un outil de test hybride qui combine l'utilisation de machines virtuelles et le simulateur OMNET++. La définition d'un protocole de test expérimental (détaillé dans le chapitre 4) permettant de tester des protocoles de routage sur des machines virtuelles et le simulateur OMNET++ a permis d'identifier les avantages du protocole AODV par rapport aux autres familles de protocole de routage.

6.1.2 Besoin en sécurité dans un réseau UAANET

Comme dans tout type de réseau ad hoc mobile, le réseau UAANET fait l'hypothèse que chaque nœud est bienveillant et qu'il coopère automatiquement aux règles établies dans le réseau. Néanmoins, cette hypothèse est difficile à tenir surtout dans un environnement hostile sans infrastructure fixe comme le réseau UAANET. Plusieurs raisons nécessitent de réfléchir à la problématique de sécurité pour cet environnement. Nous pouvons en particulier rappeler : l'utilisation des liens sans fil, l'utilisation d'un canal de communication vulnérable non sûr, un environnement non contrôlé qui est dû à l'absence d'une entité centrale qui pourrait contrôler l'entrée et la sortie des nœuds dans le réseau, la topologie dynamique qui rend difficile la distinction entre l'impact d'une attaque et une perte de connectivité du réseau dû aux vitesses des nœuds, et la limitation des ressources. Ces différentes raisons font que des attaques sophistiquées peuvent être réalisées dans les réseaux UAANET. Nous pouvons, par exemple, citer l'attaque *blackhole* ou l'attaque *wormhole* que nous avons plus particulièrement considérées dans notre étude.

Par ailleurs, pour développer l'approche adéquate de sécurisation à mettre en place pour ce type de réseau, nous avons conduit différentes analyses de risque portant sur les attaques pouvant être menées depuis la couche physique jusqu'à la couche applicative. Notre analyse a conclu sur l'intérêt de considérer la couche réseau spécifiquement, car cette couche de communication permet la connectivité des nœuds du réseau que nous considérons dans cette thèse. Il est donc important de proposer une solution d'authentification et d'intégrité à ce niveau avant de se focaliser sur les autres couches.

Pour répondre à ce besoin de sécurité, nous nous sommes tournés vers des algorithmes de routage sécurisés existants dans le domaine des MANET et nous avons choisi comme approches de les améliorer pour les rendre plus performants à l'environnement UAANET et plus robuste aux attaques spécifiques qui peuvent toucher cet environnement. L'étude (détaillé dans le chapitre 3) des protocoles existants nous a conduits à considérer le protocole SAODV qui permet d'assurer l'authentification des messages. Toutefois, ce protocole est vulnérable à l'attaque *wormhole* qui consiste à retransmettre un paquet sécurisé vers un tunnel *wormhole* sans modification. Ce type d'attaque peut par la suite interférer avec l'échange des données de charge utile ou encore diminuer la performance du protocole en triant ou en jetant les paquets de routage à retransmettre. Pour remédier à cela, nous

avons proposé dans le chapitre 4, le protocole SUAP qui garantit l'authentification et l'intégrité des messages respectivement par des mécanismes de signature numérique et de hachage. Ces derniers sont des mécanismes présents dans SAODV et que nous avons implémentés par modélisation pour contribuer à la certification du système UAS final. Pour contrer l'attaque *wormhole*, nous avons proposé une approche de détection et de prévention originale qui se base sur l'association entre le nombre de sauts et la distance relative entre deux nœuds. Nous avons également ajouté un autre mécanisme contre le *wormhole* qui consiste à prendre en compte l'identité du nœud suivant dans le calcul de la valeur de hash. Ces deux mécanismes sont complémentaires, car le premier mécanisme permet de sécuriser les paquets de maintenance de route tandis que le second est consacré aux paquets de découverte de route.

6.1.3 Application d'une méthodologie orientée modèle pour le développement du protocole SUAP

Dans le chapitre 4 et chapitre 5, nous avons introduit et utilisé une méthodologie de prototypage rapide pour développer notre protocole de routage SUAP. Ainsi, nous avons créé des modèles pour les différentes familles de fonctions du protocole. Cette mise en œuvre du protocole SUAP s'étend de la modélisation à l'aide de machines à états-transitions des protocoles jusqu'à leur utilisation sur un système UAS réel. Notre méthodologie a considéré en particulier la couche réseau avec les différents blocs de sécurité nécessaire pour assurer l'authentification et l'intégrité des paquets de routage, mais également les couches basses (physique et liaison) nécessaires pour l'échange de trames entre les nœuds voisins dans le réseau. De plus, le protocole SUAP a été vérifié formellement par des outils de vérification formelle inhérents à l'outil de conception utilisé (i.e. Matlab Simulink & Stateflow). Ces vérifications ont été élaborées pour contribuer à la certification du système de communication UAS.

6.1.4 Validation fonctionnelle de la partition de routage et de sécurisation

Dans le chapitre 5, nous avons évalué la performance des différentes familles de fonction du protocole SUAP (routage et sécurité). Cette validation a été effectuée par évaluation de performance à travers la combinaison d'une étude menée par simulation réseau et tests réels. L'évaluation de la fonction routage du protocole SUAP a été menée en comparant SUAP au protocole de référence AODV considéré dans ce travail de thèse. Cette évaluation a été menée en déployant en environnement réel de plusieurs drones afin de pouvoir vérifier les performances de notre protocole dans un cadre le plus réaliste possible. Pour la partie sécurité, elle a été réalisée dans une approche comparative avec un protocole non sécurisé pour étudier l'impact des attaques *blackhole* sur l'état du réseau. Cette partie de

l'évaluation a nécessité l'utilisation de simulation, mais aussi d'expérimentations réelles pour permettre la validation de l'ensemble des fonctionnalités de sécurité. Les attaques les plus complexes comme l'attaque *wormhole* n'ont pu être validées qu'en simulation étant donné qu'elle nécessite le déploiement d'un grand nombre de nœuds et nous n'avions pas assez de prototypes DT-18 disponibles pour faire cette campagne de tests. Il en résulte que les moyens de validation expérimentale mis en œuvre dans ce chapitre (la simulation et les expérimentations réelles) sont deux moyens tout à fait complémentaires pour l'évaluation de notre protocole de routage. En effet, la simulation facilite la réalisation d'études de scénarios plus complexes alors que l'expérimentation réelle met le système dans son environnement d'application réel entouré de tous paramètres dynamiques qui peuvent affecter ses performances.

Les résultats que nous avons obtenus aussi bien pour la validation des fonctions de routage que de sécurité nous montrent que le protocole s'adapte bien à l'environnement réel avec des valeurs de performance acceptable au regard du cahier des charges initialement défini pour ce projet de recherche. Nous avons en particulier constaté qu'en présence d'une faible densité de nœuds et avec une vitesse importante des nœuds, le taux de perte de paquets de données utiles est moindre en présence d'un ou de plusieurs attaquants. De plus, en matière de quantité de trafic de signalisation généré, l'overhead généré par SUAP dans le réseau reste faible et la bande passante disponible dans le réseau reste très majoritairement disponible pour l'échange des données utiles temps réel. Ceci est principalement explicable par le fait que SUAP est un protocole réactif et qu'il génère peu de paquets de signalisation. Toutefois, la topologie dynamique du réseau fait que l'on perd un lien en moyenne tous les 14.34 secondes. La durée de rétablissement de la route que nous avons relevé dans nos études de performance montre toutefois la résilience du protocole à trouver une route alternative rapidement dans le réseau. La vitesse de déplacement des nœuds a peu d'influence sur cette valeur. Ensuite, en ce qui concerne le délai moyen d'acheminement des paquets de données, il reste très faible, ceci étant principalement dû a la faible densité de nœuds de notre réseau UAANET expérimental. En effet, suivant la topologie déployée au cours des différents tests, nous n'effectuons qu'un ou deux sauts entre la source et la destination.

Ces différents résultats nous permettent de conclure que l'algorithme SUAP développé au cours de cette thèse offre une bonne adaptabilité vis-à-vis de la spécifiée du réseau UAANET.

6.2 Perspectives

Bien que cette thèse apporte des contributions dans le cadre des communications sécurisées pour les mini-drones civils, plusieurs pistes de travaux futurs peuvent être considérées

comme compléments de recherche à nos travaux.

6.2.1 Étude de performances complémentaires pour SUAP

Les études de performances des différentes fonctions du protocole SUAP présentées dans ce manuscrit ont été suffisantes pour valider notre étude et les fonctionnalités du protocole SUAP. Toutefois, nous avons identifié des améliorations potentielles à mettre en œuvre pour augmenter la phase d'évaluation de performances présentée dans le dernier chapitre de ce travail de thèse. Ces travaux n'ont pas été conduits dans le cadre de cette thèse principalement par manque de ressources matérielles spécifiques (nous n'avions pas la possibilité de faire voler un nombre plus important de drones DT-18), mais également pour des raisons de calendrier. Il est néanmoins intéressant de pointer les volets d'amélioration que nous envisageons.

1. Il serait intéressant d'augmenter le nombre de nœuds durant l'expérimentation réelle. Dans notre cas, nous avons utilisé que 4 nœuds communicants, mais il serait judicieux d'étudier le comportement du protocole SUAP avec une plus grande densité des nœuds. Cela nous permet aussi de vérifier en environnement réel l'attaque *wormhole*.
2. Pour la validation de la partition de sécurisation, il aurait été intéressant de créer un attaquant mobile qui puisse directement interagir avec les drones en l'air afin de mesurer l'impact des mécanismes d'authentification mis en place. Il serait, par exemple, intéressant de déployer un drone attaquant qui volerait dans la zone de connexion d'un nœud cible et échange des paquets de routage fictifs.
3. Pour des contraintes de disponibilité matérielles, il n'a pas été possible au cours de cette thèse de valider en environnement réel la proposition contre l'attaque *wormhole*. Il serait alors intéressant de déployer une flotte de drones pour tester ce mécanisme.

6.2.2 Sécurité applicative des échanges en environnement UAANET

L'étude faite au cours de cette thèse s'est consacrée majoritairement sur la couche réseau par des mécanismes d'authentification et d'intégrités des trafics de signalisation pour le routage dans le réseau. C'est-à-dire qu'on s'assure que la route utilisée est intègre, mais la question de sécurité associée aux paquets des données n'a pas été traitée. Ces données peuvent être divulguées s'il y a un attaquant qui fait une écoute illicite (Man in the middle) sur le médium sans fil. Dans ce cas, il faut alors assurer sa confidentialité par des techniques cryptographiques.

Nous pouvons poser la question de l'intérêt de cette couche de sécurité au niveau applicatif en termes de besoin utilisateur. La réponse à cette question est affirmative, car si nous nous intéressons au cahier des charges défini avec l'industriel dans ce travail de thèse, il arrive souvent que Delair-Tech fasse une surveillance d'une zone industrielle ayant un caractère

confidentiel. Dans ce cas, ce type de données ne doit pas être divulgué à des tiers, car il contient tous les points stratégiques d'accès à la zone.

Dans la littérature, les propositions qui existent se basent généralement sur l'utilisation d'une cryptographie hybride en chiffrant les données à leur sortie du nœud émetteur et ensuite en les déchiffrant côté récepteur [HNA11]. Toutefois, dans le cadre de notre application UAANET il serait nécessaire de considérer la question de la contrainte temporelle forte du trafic applicatif. Ceci n'a pas été abordé spécifiquement dans la littérature que nous avons analysée. Ces solutions ne sont donc pas utilisables clé en main pour un contexte de données temps réel. Ce domaine de recherche semble donc une piste envisageable à la suite de nos travaux dans laquelle il serait intéressant de trouver une solution de sécurité applicative pour assurer la confidentialité des données utiles tout en garantissant leur caractère temps réel.

6.2.3 Conception d'une infrastructure de gestion de clé adaptée au contexte UAANET

Pour assurer une communication sécurisée entre les drones et la station de base, le recours à un protocole cryptographique est essentiel. Nous avons vu dans le manuscrit les mécanismes de signature numérique et de hachage qui se basent sur des procédés cryptographiques. Toutefois, il ne faut pas oublier que recourir à ces méthodes implique de concevoir une organisation de gestion de clés cryptographiques pour assurer la cohérence et l'efficacité d'un tel système.

Dans le cas particulier du réseau UAANET, aucune proposition n'a été à ce jour avancée. Nous pouvons alors faire une analogie de ce qui a été fait dans les réseaux MANET et étudier leurs adaptabilités dans le contexte d'un réseau ad hoc de drones. Sa mise en place doit notamment faire face aux spécificités de ce type de réseau. Ainsi, au cours de cette thèse, nous nous sommes posé la question de proposer une infrastructure de gestion de clé. L'étude de l'état de l'art a été effectuée et des propositions préliminaires ont été étudiées analytiquement. Pour cela, nous avons fait une étude synthétique des propositions actuelles de PKI pour les réseaux ad hoc. Généralement, l'implémentation d'une PKI classique dans le contexte d'un réseau MANET n'est pas envisageable pour plusieurs raisons. Tout d'abord, il manque une infrastructure fixe. De plus, une autorité de certification centralisée constitue un point de défaillance unique. Il faut aussi prendre en compte le fait que dans un réseau mobile tel qu'un réseau de drones, les nœuds sont en déplacement constant et la topologie du réseau varie en conséquence. Pour répondre à ces problèmes, différents types d'implémentation ont été proposés dans la littérature. Nous pouvons distinguer quatre catégories : les PKI partiellement distribuées, les PKI entièrement distribuées, les PKI basées sur des chaînes de certificats et enfin les PKI basées sur le clustering.

Nous pouvons dire que de nombreuses architectures de PKI existent ou ont du moins été imaginées pour la famille de réseaux MANET. Celles-ci ont toutes leurs particularités, tout en gardant un objectif commun. Cependant, aucune de ces propositions ne correspond exactement au besoin du réseau UAANET. En s'appuyant sur les mécanismes existants, il serait intéressant de chercher à déterminer un modèle de PKI propre au besoin du réseau UAANET. Nous pouvons citer par exemple, en matière d'exigence spécifique pour les réseaux UAANET, qu'il serait primordial de s'assurer que le système de PKI minimise les échanges entre les aéronefs et avec la station sol. Ces échanges sont en effet coûteux et peuvent augmenter le temps de latence entre le moment où le drone entre dans le réseau et le moment ou la station de base peut communiquer avec lui. Par ailleurs, une des spécificités du réseau UAANET que l'on pourrait exploiter pour mettre en place une PKI est le fait que la station sol peut être immobile et garder sa position initiale pendant toute la durée de la mission. Dans ce cas, il serait alors intéressant de considérer la station de base comme une autorité de certification. Le drone qui vient se connecter au réseau doit dans un premier temps établir des échanges avec la station pour s'authentifier.

Ce travail de thèse a permis la proposition d'une architecture de communication sécurisée d'une flotte de drones. Les propositions que nous avons avancées concernent l'authentification et l'intégrité des messages. Nous avons également proposé une technique de détection de l'attaque *wormhole*. Les résultats obtenus nous permettent de conclure que notre protocole SUAP maintient la performance réseau souhaitée en cas d'attaque qui modifie les paquets légitimes ou qui génère de faux paquets. Par ailleurs, ce travail de thèse a montré qu'en utilisant l'approche orientée modèle, il est possible de produire rapidement des logiciels répondants à de contraintes fortes de certification et d'évaluation. En outre, la modélisation de protocole de sécurité nous a permis de créer deux modèles à double usage : la validation des partitions du protocole et la génération automatique de code source. Ces propriétés font partie des éléments importants pour la certification du système final.

À part les travaux effectués pour l'élaboration du protocole de routage SUAP, l'ensemble des perspectives que nous avons présenté ouvre la voie à de nouvelles recherches pour créer un nouveau protocole de routage sécurisé et améliorer la fiabilité et la sécurité de l'architecture de communication d'une flotte de drones, tout en contribuant à sa certification.

Publications

Article dans une revue acceptée

— Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, Nicolas Larrieu. Survey on UAANET Routing Protocols and Network Security Challenges. Ad Hoc & Sensor Wireless Networks, PKP Publishing ServicesNetwork 2017. <hal-01465993>

Conférences internationales

— Jean-Aimé Maxa, Gilles Roudière, Nicolas Larrieu. Emulation-Based Performance Evaluation of Routing Protocols for Uaanets. Nets4Aircraft 2015, May 2015, Sousse, Tunisia. Springer, LNCS (9066), pp.227-240, Nets4Cars/Nets4Trains/Nets4Aircraft 2015. <hal-01132516>

— Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, Nicolas Larrieu. Secure Routing Protocol Design for UAV Ad Hoc Networks. DASC 2016, IEEE/AIAA 34th Digital Avionics Systems Conference, Sep 2015, Prague, Czech Republic. <hal-01166852>

— Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, Nicolas Larrieu. Joint model-driven design and real experiment-based validation for a secure UAV ad hoc network routing protocol. ICNS 2016, 2016 , Apr 2016, Herndon, United States. pp 1E2-1 - 1E2- 16 /, 2016 (ICNS) <10.1109/ICNSURV.2016.7486324>. <hal-01292300>
Best paper of Track Cyber security award

— Emerson A Marconato, Jean-Aimé Maxa, Daniel F Pigatto, Kalinka R L J Castelo Branco, Jean-Aimé Maxa, Nicolas Larrieu, IEEE 802.11n vs. IEEE 802.15.4 : a study on Communication QoS to provide Safe FANETs. 46th annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2016) , Jun 2016, Toulouse, France. IEEE, DSN 2016, IEEE/IFIP International Conference on Dependable Systems and Networks. <hal-01318385v2>

— Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, Nicolas Larrieu. Extended Verification of Secure UAANET Routing Protocol. DASC 2016, IEEE/AIAA 35th Digital Avionics Systems Conference, Sep 2016, Sacramento, United States. <hal-01365933>

Conférence nationale

— Jean-Aimé Maxa. Model-driven approach to design a secure routing protocol for UAV Adhoc networks. EDSYS 2015, 15ème Congrès des doctorants, May 2015, Toulouse, France. <hal-01166854>

Poster

— Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, Nicolas Larrieu. How to improve routing protocol security in a RPAS swarm ?. 4th AETOS International Workshop on "Research Challenges for Future RPAS/UAV Systems", Oct 2016, Mérignac, France. <hal-01362115>
Best Poster Award Winner

Rapports techniques

— Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, Nicolas Larrieu, Gilles Roudière, Rita Zgheib, Cahier des charges : Architecture de communication securisée pour une flotte de drones. 2014. <hal-01010315>

— Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, Nicolas Larrieu, Gilles Roudière, Rita Zgheib, Choix d'un protocole de routage pour le projet SUANETs (Security of UAV Ad hoc Networks). 2014. <hal-01010319>

Chapitre A

Annexe A

A.1 Outils de simulation et d'émulation

L'outil de test utilisé pour la comparaison des protocoles de routage combine l'approche de simulation et d'émulation.

D'une part, l'étude par simulation permet d'inclure de nombreux paramètres liés à l'environnement et la mobilité, mais ne permet d'évaluer que l'implémentation du protocole réalisée pour le simulateur. Une implémentation pour un système d'exploitation ne peut pas être testée avec cette solution.

D'autre part, l'approche par émulation permet d'évaluer une implémentation pour un système d'exploitation standard. Dans notre étude, il était nécessaire d'évaluer le fonctionnement du protocole en local avant d'aller sur le terrain. Ainsi, l'émulation expose une solution d'évaluation de protocoles de routage ad hoc à partir de machines virtuelles. Le but étant de pouvoir évaluer une implémentation pour un système d'exploitation standard sans avoir à réaliser un test en environnement réel.

A.1.1 Mesure passive

Effectuer une mesure passive qui consiste à capturer un maximum d'information en un point du réseau sans altérer le trafic permet d'avoir des données précises concernant l'ensemble des paquets évoluant sur le réseau virtuel. Afin de capturer le trafic passant sur chacune des interfaces, nous avons utiliser *tcpdump* [tcp14]. Le choix a été fait de capturer l'ensemble des paquets passés par chaque interface, la sélection des paquets considérés comme intéressants pouvant s'effectuer a posteriori. Il est important de noter que le module utilisé pour bloquer le trafic ne peut empêcher un paquet d'être capturé par *tcpdump*, il ne peut qu'empêcher le traitement par les couches supérieures du modèle OSI.

Pour analyser les captures enregistrées par *tcpdump* nous avons utilisé *Wireshark*[1]. C'est un outil d'analyse performant permettant d'extraire plusieurs métriques d'une ou plusieurs traces réseau. Il est possible d'extraire d'une trace le trafic qui nous intéresse et d'en calculer le débit.

Il est important de constater que le fonctionnement de *VirtualBox* nous force, à moins de considérer la réécriture des drivers, à utiliser les interfaces en mode promiscuité. *tcpdump* capture donc l'ensemble du trafic passant sur le médium, ce qui dans notre cas nous permet d'évaluer l'ensemble de le l'*overhead* créé par chacun des protocoles sur tout le réseau.

A.1.2 Mesure active

Étant donné que nous utilisons des machines virtuelles, la génération du trafic peut être effectuée avec n'importe quelle application. Il est alors possible d'évaluer directement la qualité du ou des services utilisés.

Dans notre cas, la génération est effectuée grâce à un script python cherchant à créer un trafic similaire à un scénario d'utilisation réaliste. Ce choix est motivé par la possibilité d'ajouter aux données une charge utile : la date d'émission et un identifiant incrémenté sont ajoutés en entête. Celle-ci nous permet d'évaluer en réception le délai de bout en bout ainsi que les pertes ayant eu lieu.

A.1.3 Introduction au framework *Virtualmesh*

Le framework *VirtualMesh* a été proposé en 2010 par Reto Gantenbein [Gan10]. Ce framework est un outil permettant d'interfacer un système Linux avec une simulation *OMNeT++* [omn14].

OMNeT++ est un simulateur réseau codé en C++. C'est un simulateur modulaire, permettant, à l'aide du framework *INET* [SKKS11], la simulation simple d'un grand nombre de systèmes et protocoles normalisés.

L'utilisation standard de *VirtualMesh* sur un système consisterait en ces quelques étapes :

— Une interface sans-fil virtuelle est créée sur le système Linux que l'on souhaite connecter à la simulation.

— La simulation *OMNeT++* (composée au moins des modules proposés par le framework) est lancée.

— Un outil, lancé sur le système Linux, permet de lier l'interface virtuelle à la simulation. Des sockets UDP sont ici utilisés, permettant de connecter à la simulation des

1. http ://www.wireshark.org/

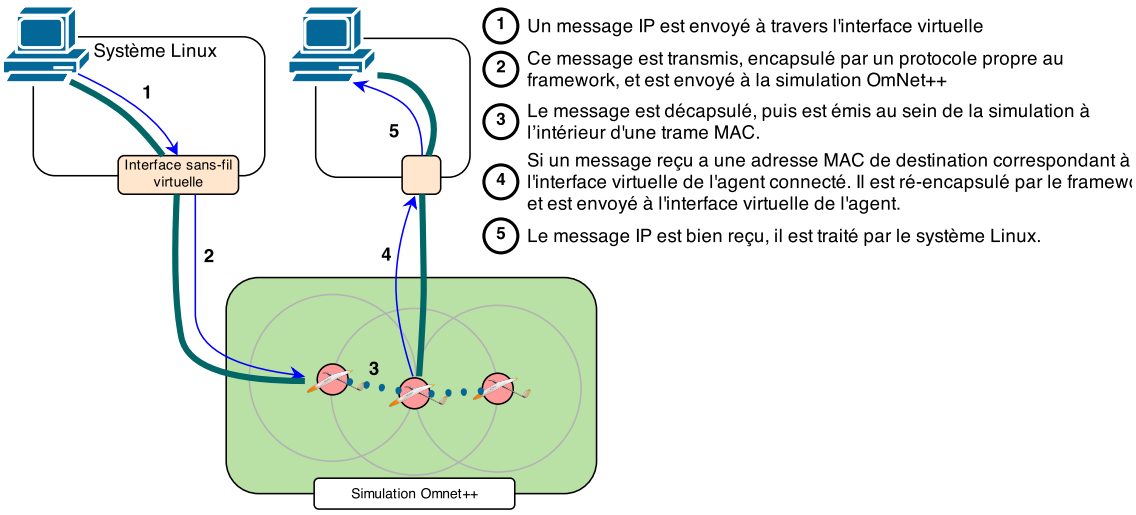

FIGURE A.1 : Envoi d'un paquet IP à travers le framework *Virtualmesh*

systèmes émulés ou réels.

Ainsi, lorsque ces étapes ont été réalisées, un nouvel agent apparaît dans la simulation. Lorsqu'un paquet est émis au niveau d'une interface virtuelle, celui-ci est envoyé à la simulation par l'intermédiaire du socket UDP, qui le relaie alors en fonction des paramètres de l'environnement physique de la simulation (portée des agents, atténuation, puissance d'émission, etc.). À l'inverse, un paquet reçu par l'un des agents de la simulation est relayé à l'interface virtuelle. Une illustration de ce système est visible sur la figure A.1.

En résumé, ce framework permet de simuler tous les aspects liés à l'environnement physique des nœuds (déplacement des agents, accès au médium ...).

Intégration de Virtualmesh au protocole expérimental

Afin de faire évoluer notre protocole expérimental, nous avons choisi d'inclure le framework *Virtualmesh* au protocole expérimental. Ceci pour permettre d'inclure deux paramètres supplémentaires à l'évaluation que sont l'accès au médium et la mobilité. La figure A.2 est une copie d'écran du protocole expérimental en pleine action.

A.1.4 Prise en compte de la mobilité dans le système

Afin d'obtenir un scénario de connectivité plus réaliste, l'inclusion de la mobilité dans le protocole expérimental est devenue nécessaire. Cela a eu une conséquence sur le protocole expérimental : les déconnexions totales -c'est-à-dire quand il n'existe aucun chemin possible de la source à la destination- sont devenues inévitables. On obtient donc parfois dans les fichiers de résultats des temps de déconnexion assez longs qui ne sont pas causés par les

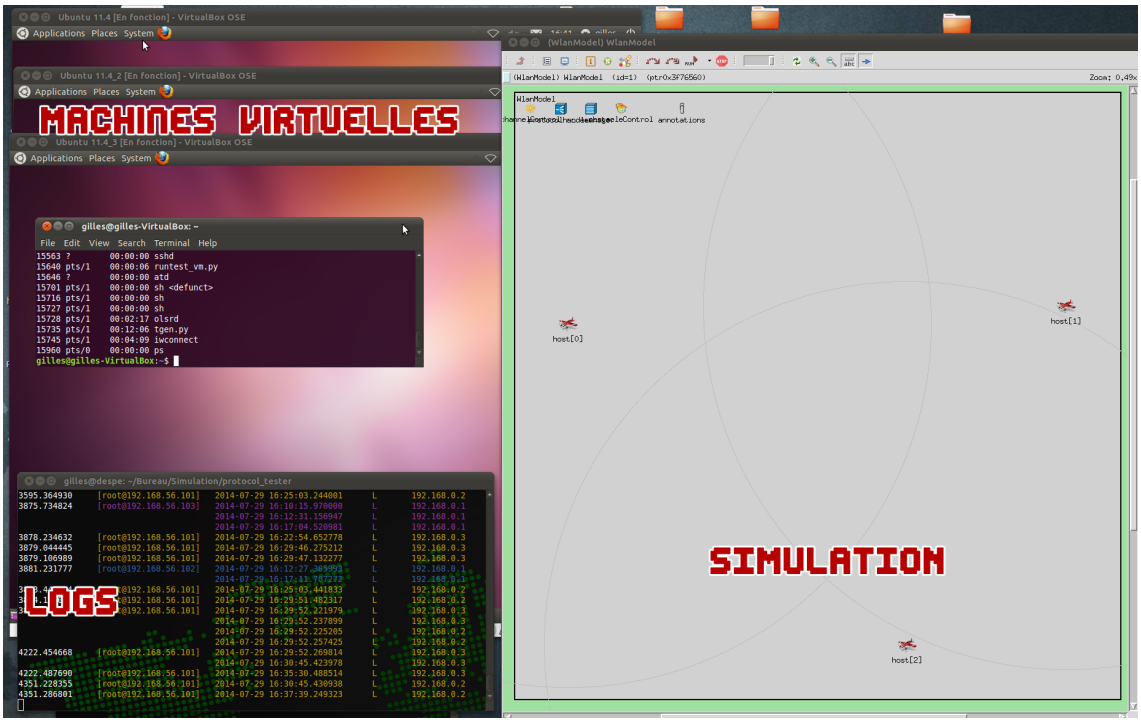

FIGURE A.2 : Capture d'écran du nouveau protocole expérimental en fonctionnement

mécanismes de réparation de route des protocoles de routage.

Par conséquent, de manière à obtenir un scénario le plus réalise possible, nous avons utilisé des traces fournies par Delair-Tech pour recréer un scénario de mobilité correspondant au scan d'une zone. À partir d'une trace réelle et grâce à des rotations et changements d'échelle, nous avons recréé un scénario avec 3 drones en scannant une zone en collaboration. Les traces résultantes sont illustrées dans la figure A.3.

Ainsi les mouvements des drones simulés sont aussi réalistes que cela nous était possible.

Accès au médium

Virtualmesh, à travers les fonctionnalité proposée par *OMNeT++* et *INET* permettent de simuler un accès au médium réaliste (avec collisions, atténuation du signal ...). Après quelques tests, nous nous sommes aperçus qu'un modèle réaliste au niveau MAC imposait de nombreuses pertes, rendant l'évaluation du protocole de routage difficile. En effet, les pertes occasionnelles causées par la couche MAC permettaient difficilement d'évaluer les pertes causées par le protocole de routage. Nous avons donc remplacé la couche MAC par une couche MAC idéale, où chaque interface est caractérisée par un débit et une portée.

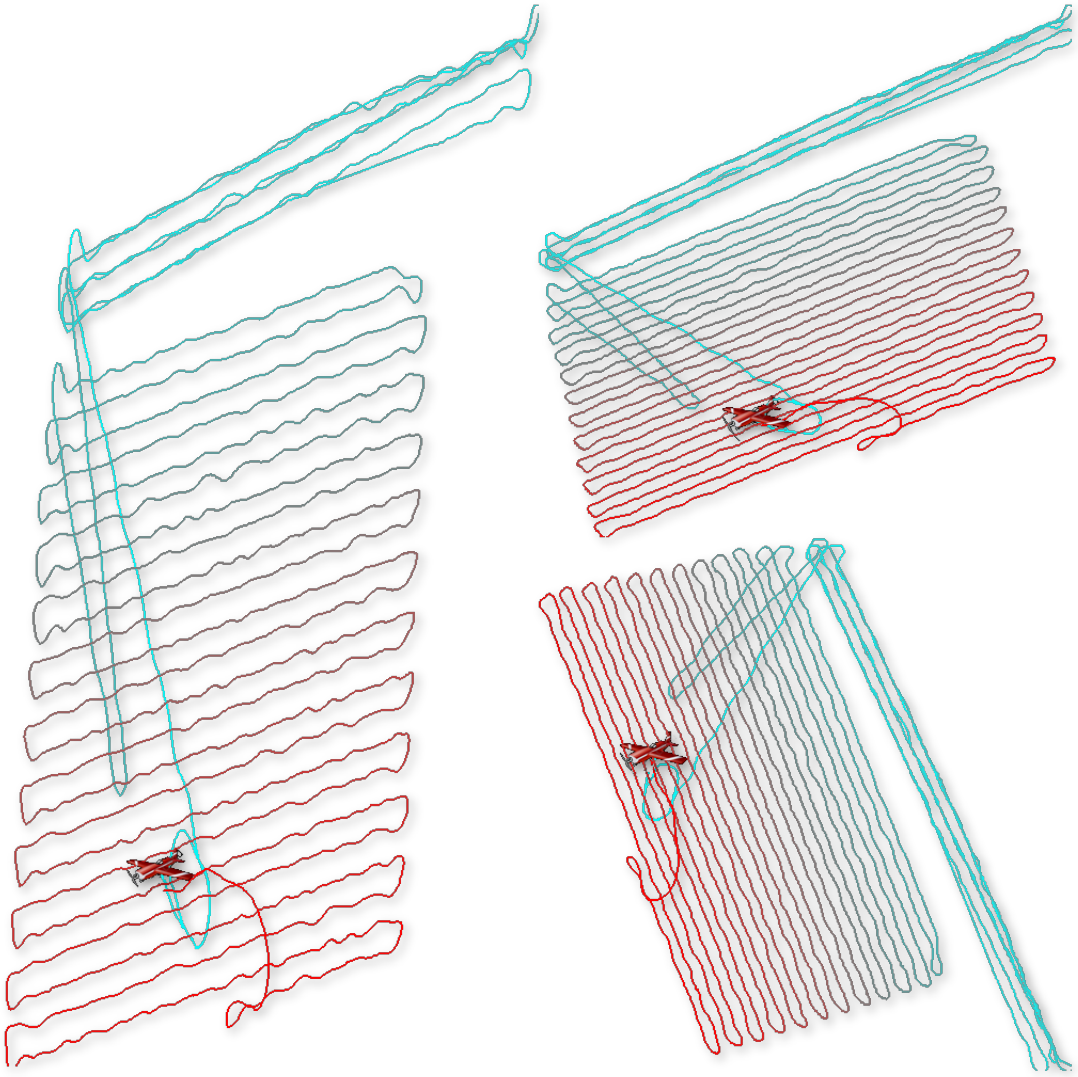

FIGURE A.3 : Traces de trois drones scannant une zone

Chapitre B

Annexe B

B.1 Vérification formelle avec AVISPA

B.1.1 Quelques outils de vérification formelle existants

Généralement, les outils de vérification formelle reposent sur l'une des deux classes techniques suivantes : les techniques algorithmiques appelées « presse-bouton » et les techniques dites de *preuve*. Dans la technique de preuve, il y a deux approches, la méthode du raisonnement logique [PBP+10] et l'approche par preuve de théorèmes. La méthode du raisonnement logique consiste à décrire les principes de communication et les évaluer grâce à des règles de déduction. Ces raisonnements logiques permettent d'affirmer si un protocole est sécurisé ou pas. En ce qui concerne l'approche par preuve de théorèmes, elle se base sur un raisonnement mathématique formel. Les outils se basant sur cette approche utilisent la HOL(High Order Method) [XK01] et la FOL (First Order Method) [BBC11]. Cette méthode permet d'obtenir une preuve formelle, mais n'est pas totalement automatisée.

Dans la technique de « presse-bouton », il est commun de s'intéresser exclusivement à l'exploration d'états, c'est-à-dire à la recherche de toutes les exécutions possibles du protocole. La vérification dite « non bornée » considère une infinité de principes et une infinité de sessions. L'espace des états généré par l'analyse d'un protocole peut être infini. La vérification dite « bornée » stipule que le nombre de sessions est une constante. L'objectif de la vérification est alors de chercher un état où les mécanismes de sécurité sont attaqués et génère ainsi un exemple d'attaque. C'est sur cette approche que reposent la plupart des outils de vérification automatique. De toutes les approches évoquées, nous allons utiliser cette dernière. En effet, les techniques de preuve nécessitent une approche de théories mathématiques et cryptographiques pour établir les démonstrations de propriétés et pour formaliser les résultats. Parmi les outils qui se basent sur cette méthode, nous pouvons citer : ISABELLE [NPW02] et COQ [DFH+92]. Il y également l'outil SPIN [Hol03] qui

Outil	Open Source	Falsification	Borné	Non borné	Terminaison
OFMC	✓	✓	✓	✗	✓
CL-Atse	✓	✓	✓	✗	✓
SATMC	✓	✓	✓	✗	✓
TA4SP	✓	✓	✗	✓	✓
AVISPA	✓	✓	✓	✓	✓
FDR/Casper	✓	✓	✓	✗	✓
HERMES	✓	✓	✗	✓	✓
Interrogator	✗	✓	✓	✗	✓
NRL Analyzer	✗	✗	✗	✓	✓
Brutus	✗	✓	✓	✗	✓
ProVerif	✓	✓	✗	✓	✓
Athena	✗	✓	✓	✓	✓
Scyther	✓	✓	✓	✓	✓

Tableau B.1 : Comparaison des outils de vérification formelle

est utilisé pour la vérification[1] des modèles de spécification (Model checking).

Une comparaison d'outils de vérification formelle existant est montré par la figure B.1 en nous appuyant sur les travaux présentés dans [PBP+10].

B.1.2 Architecture logicielle de l'outil AVISPA

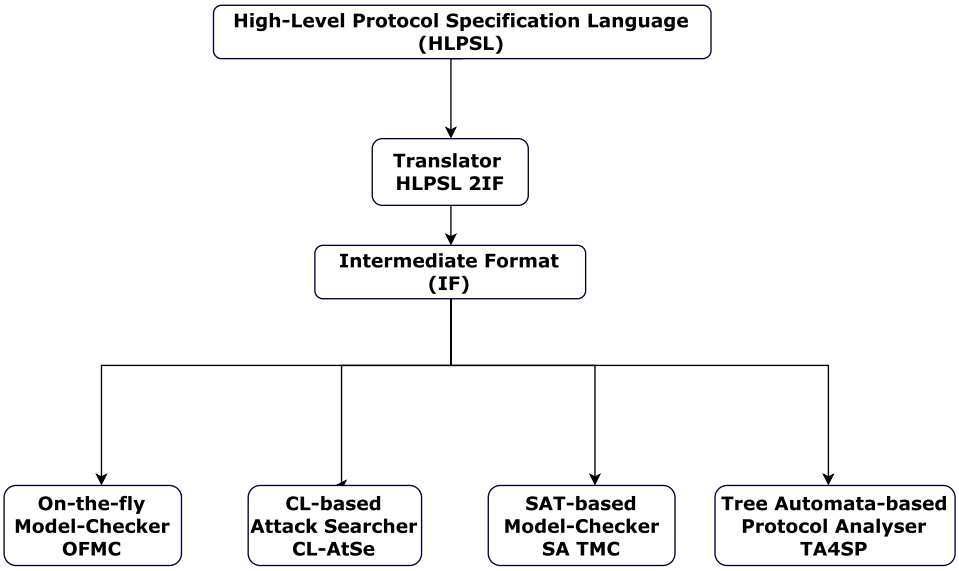

FIGURE B.1 : Architecture logicielle d'AVISPA

Les différents back-ends dans AVISPA sont cités ci-dessous :

1. Ici, il s'agit surtout de la vérification des fonctions de sécurité contrairement à la vérification de conception que l'on à vue au chapitre précédent

— **OFMC (On the Fly Model Checker)** : utilisé pour un protocole dans lequel des propriétés algébriques sont à vérifier

— **CL-AtSe** : qui est basé sur les contraintes. Il simplifie le protocole à vérifier en éliminant les états redondants par application de propriétés heuristiques. Sa modularité permet d'intégrer facilement d'autres spécifications de fonctions cryptographiques.

— **SATMC** : utilise un état transitoire et recherche les éventuelles violations d'un protocole donné. La sortie est une représentation mathématique de la violation et la transforme en attaque.

— **T4SP** montre la vulnérabilité d'un protocole en faisant une estimation des capacités de l'intrus.

B.1.3 Simulation d'intrus dans AVISPA. Attaque de type confidentialité

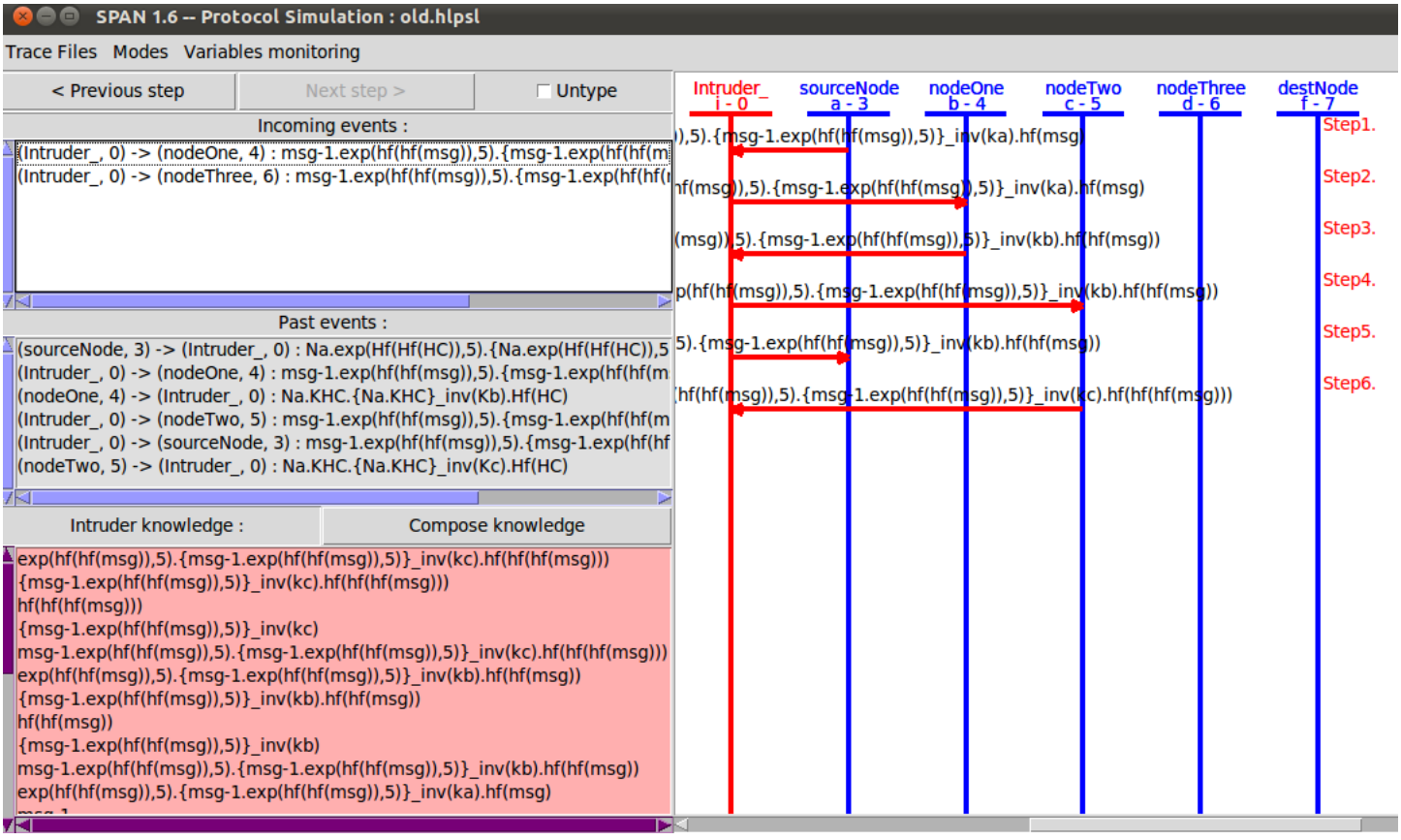

FIGURE B.2 : Simulation d'intrus dans AVISPA

Bibliographie

[ABB+05] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *International Conference on Computer Aided Verification*, pages 281–285. Springer, 2005.

[ABC+] Raja Naeem Akram, Pierre-François Bonnefoi, Serge Chaumette, Konstantinos Markantonakis, and Damien Sauveron. Improving security of autonomous uavs fleets by using new specific embedded secure elements a position paper.

[ACH+04] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. Mitigating byzantine attacks in ad hoc wireless networks. *Department of Computer Science, Johns Hopkins University, Tech. Rep. Version*, 1, 2004.

[ACJ+03] Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Muhlethaler, and Daniele Raffo. Securing the olsr protocol. In *Proceedings of Med-Hoc-Net*, pages 25–27, 2003.

[AD10a] Abdel Ilah Alshbatat and Liang Dong. Adaptive mac protocol for uav communication networks using directional antennas. In *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, pages 598–603. IEEE, 2010.

[AD10b] Abdel Ilah Alshbatat and Liang Dong. Cross layer design for mobile ad-hoc unmanned aerial vehicle communication networks. In *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, pages 331–336. IEEE, 2010.

[AD11] Abdel Ilah Alshabtat and Liang Dong. Low latency routing algorithm for unmanned aerial vehicles ad-hoc networks. 5(8) :629 – 635, 2011.

[AFOTH06] Jim Alves-Foss, Paul W Oman, Carol Taylor, and W Scott Harrison. The MILS architecture for high-assurance embedded systems. *International journal of embedded systems*, 2(3-4) :239–247, 2006.

[AK07] Fadi A Aloul and Nagaragan Kandasamy. Sensor deployment for failure diagnosis in networked aerial robots : a satisfiability-based approach. In *Theory and Applications of Satisfiability Testing–SAT 2007*, pages 369–376. Springer, 2007.

[AKR07] Rehan Akbani, Turgay Korkmaz, and GVS Raju. Heap : hop-by-hop efficient authentication protocol for mobile ad-hoc networks. In *Proceedings of the 2007 spring simulaiton multiconference-Volume 1*, pages 157–165. Society for Computer Simulation International, 2007.

[AP05] Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In *Cryptographers' Track at the RSA Conference*, pages 191–208. Springer, 2005.

[ARP96] SAE ARP. 4754. *Certification considerations for highly-integrated or complex aircraft systems*, 1996.

[AS16] KS Arathy and CN Sminesh. A novel approach for detection of single and collaborative black hole attacks in manet. *Procedia Technology*, 25 :264–271, 2016.

[AWS06] Jinthana Ariyakhajorn, Pattana Wannawilai, and Chanboon Sathitwiriyawong. A comparative study of random waypoint and gauss-markov mobility models in the performance evaluation of manet. In *2006 International Symposium on Communications and Information Technologies*, pages 894–899. IEEE, 2006.

[Bac05] Marko Bacic. On hardware-in-the-loop simulation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 3194–3198. IEEE, 2005.

[BAD+04] Timothy X Brown, Brian Argrow, Cory Dixon, Sheetalkumar Doshi, Roshan-George Thekkekunnel, and Daniel Henkel. Ad hoc uav ground network (augnet). In *AIAA 3rd Unmanned Unlimited Technical Conference*, pages 1–11, 2004.

[BAFF16] Paulo Bartolomeu, Muhammad Alam, Joaquim Ferreira, and José Fonseca. Survey on low power real-time wireless mac protocols. *Journal of Network and Computer Applications*, 75 :293–316, 2016.

[BAGL14] Ouns Bouachir, Alinoe Abrassart, Fabien Garcia, and Nicolas Larrieu. A mobility model for uav ad hoc network. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 383–388. IEEE, 2014.

[BBC11] Stephen Becker, Jérôme Bobin, and Emmanuel J Candès. Nesta : A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, 4(1) :1–39, 2011.

[BBF+00] Gérard Berry, Amar Bouali, Xavier Fornari, Emmanuel Ledinot, Eric Nassor, and Robert De Simone. Esterel : A formal method applied to avionic software development. *Science of Computer Programming*, 36(1) :5–25, 2000.

[BBF+13] Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, and Philippe Schnoebelen. *Systems and software verification : model-checking techniques and tools*. Springer Science & Business Media, 2013.

[BCC+03] Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérome Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. A static analyzer for large safety-critical software. In *ACM SIGPLAN Notices*, volume 38, pages 196–207. ACM, 2003.

[BCPQ01] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group diffie-hellman key exchange. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 255–264. ACM, 2001.

[Bel06] Ron Bell. Introduction to iec 61508. In *Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55*, pages 3–12. Australian Computer Society, Inc., 2006.

[BGLG13] Ouns Bouachir, Fabien Garcia, Nicolas Larrieu, and Thierry Gayraud. Ad hoc network qos architecture for cooperative unmanned aerial vehicles (uavs). In *Wireless Days (WD), 2013 IFIP*, pages 1–4. IEEE, 2013.

[BH07] T Baghai and V Hildeman. Avionics certification : A complete guide to do-178 (software), do-254 (hardware). *United States*, 2007.

[BLB02] Sonja Buchegger and J-Y Le Boudec. Nodes bearing grudges : Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on*, pages 403–410. IEEE, 2002.

[BLT02] B Bellur, M Lewis, and F Templin. An ad-hoc network for teams of autonomous vehicles. In *Proceedings of the First Annual Symposium on Autonomous Intelligence Networks and Systems*. Citeseer, 2002.

[BST13] Ilker Bekmezci, Ozgur Koray Sahingoz, and ŞAmil Temel. Flying ad-hoc networks (fanets) : A survey. *Ad Hoc Networks*, 11(3) :1254–1270, 2013.

[BTJBL12] Rosana TV Braga, Onofre Trindade Jr, Kalinka RLJ Branco, and Jaejoon Lee. Incorporating certification in feature modelling of an unmanned aerial vehicle product line. In *Proceedings of the 16th International Software Product Line Conference-Volume 1*, pages 249–258. ACM, 2012.

[CCC10] HaiYang Chao, YongCan Cao, and YangQuan Chen. Autopilots for small unmanned aerial vehicles : a survey. *International Journal of Control, Automation and Systems*, 8(1) :36–44, 2010.

[Cer01] Iliano Cervesato. The dolev-yao intruder is the most powerful attacker. In *16th Annual Symposium on Logic in Computer Science—LICS*, volume 1, 2001.

[CHD13] Xiuzhen Cheng, Xiao Huang, and Ding-Zhu Du. *Ad hoc wireless networking*, volume 14. Springer Science & Business Media, 2013.

[CHJ+10] Stephen Cameron, S Hailes, S Julier, S McClean, G Parr, N Trigoni, M Ahmed, G McPhillips, R De Nardi, J Nie, et al. Suaave : Combining aerial robots and wireless networking. In *25th Bristol International UAV Systems Conference*, pages 1–14, 2010.

[CHKV06] Chen-Mou Cheng, Pai-Hsiang Hsiao, HT Kung, and Dario Vlah. Performance measurement of 802.11 a wireless links from uav to ground nodes with various antenna orientations. In *Proceedings of 15th International Conference on Computer Communications and Networks*, pages 303–308. IEEE, 2006.

[CJ03] Thomas Clausen and Philippe Jacquet. Optimized link state routing protocol (olsr). Technical report, 2003.

[CLM+11] S Chaumette, R Laplace, C Mazel, R Mirault, A Dunand, Y Lecoutre, and J-N Perbet. Carus, an operational retasking application for a swarm of autonomous uavs : First return on experience. In *MILITARY COMMUNICATIONS CONFERENCE, 2011-MILCOM 2011*, pages 2003–2010. IEEE, 2011.

[CLMG09] Serge Chaumette, Rémi Laplace, Christophe Mazel, and Aurélien Godin. Secure cooperative ad hoc applications within uav fleets position paper. In *MILCOM 2009-2009 IEEE Military Communications Conference*, pages 1–7. IEEE, 2009.

[CM12] Bow-Nan Cheng and Steven Moore. A comparison of manet routing protocols on airborne tactical networks. In *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pages 1–6. IEEE, 2012.

[CSC11] Jin-Hee Cho, Ananthram Swami, and Ray Chen. A survey on trust management for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 13(4) :562–583, 2011.

[CYL+13] Yegui Cai, F Richard Yu, Jun Li, Yifeng Zhou, and Louise Lamont. Medium access control for unmanned aerial vehicle (uav) ad-hoc networks with full-duplex radios and multipacket reception capability. *IEEE Transactions on Vehicular Technology*, 62(1) :390–394, 2013.

[DB05] Saman Desilva and Rajendra V Boppana. Mitigating malicious control packet floods in ad hoc networks. In *IEEE Wireless Communications and Networking Conference, 2005*, volume 4, pages 2112–2117. IEEE, 2005.

[DDLW09] Kai Daniel, Bjoern Dusza, Andreas Lewandowski, and Christian Wietfeld. Airshield : A system-of-systems muav remote sensing architecture for disaster response. In *Systems Conference, 2009 3rd Annual IEEE*, pages 196–200. IEEE, 2009.

[Del15] Delair Tech. www.delair-tech.com, 2015. [Online ; accessed 09-octobre-2015].

[DFH+92] Gilles Dowek, Amy Felty, Hugo Herbelin, Gérard Huet, Chetan Murthy, Catherin Parent, Christine Paulin-Mohring, and Benjamin Werner. *The COQ Proof Assistant : User's Guide : Version 5.6*. INRIA, 1992.

[Dij68] Edsger W Dijkstra. Letters to the editor : go to statement considered harmful. *Communications of the ACM*, 11(3) :147–148, 1968.

[DLA02] Hongmei Deng, Wei Li, and Dharma P Agrawal. Routing security in wireless ad hoc networks. *IEEE Communications magazine*, 40(10) :70–75, 2002.

[Dro15] droneregulationdgac. http://www.developpement-durable.gouv.fr/Quelle-place-pour-les-drones-dans,45924.html, 2015. [Online ; accessed 23-novembre-2016].

[DT116] dt18. http://dronelife.com/cms/product/DT-18, 2016. [Online ; accessed 24-novembre-2016].

[DVP08] K Dalamagkidis, KP Valavanis, and LA Piegl. On unmanned aircraft systems issues, challenges and operational restrictions preventing integration into the national airspace system. *Progress in Aerospace Sciences*, 44(7) :503–519, 2008.

[EBPQ14] Richard Gilles Engoulou, Martine Bellaïche, Samuel Pierre, and Alejandro Quintero. Vanet security surveys. *Computer Communications*, 44 :1–13, 2014.

[EFL+09] Jack Elston, Eric W Frew, Dale Lawrence, Peter Gray, and Brian Argrow. Net-centric communication and control for a heterogeneous unmanned aircraft system. *Journal of Intelligent and Robotic Systems*, 56(1-2) :199–232, 2009.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 10–18. Springer, 1984.

[ErJ01] D Eastlake 3rd and Paul Jones. Us secure hash algorithm 1 (sha1). Technical report, 2001.

[ESD13] Raymond G Estrada, Gen Sasaki, and Eric Dillaber. Best practices for developing do-178 compliant software using model-based design. In *AIAA Infotech@ Aerospace (I@ A) Conference*, page 4566, 2013.

[EvHHW08] Aysegul Tuysuz Erman, Lodewijk van Hoesel, Paul Havinga, and Jian Wu. Enabling mobility in heterogeneous wireless sensor networks cooperating with uavs for mission-critical management. *IEEE Wireless Communications*, 15(6) :38–46, 2008.

[FB09] Eric W Frew and Timothy X Brown. Networking issues for small unmanned aircraft systems. *Journal of Intelligent and Robotic Systems*, 54(1-3) :21–37, 2009.

[FD07] Bo Fu and Luiz A DaSilva. A mesh in the sky : A routing protocol for airborne networks. In *MILCOM 2007-IEEE Military Communications Conference*, pages 1–7. IEEE, 2007.

[FHS07] J Hope Forsmann, Robert E Hiromoto, and John Svoboda. A time-slotted on-demand routing protocol for mobile ad hoc unmanned vehicle systems. In *Defense and Security Symposium*, pages 65611P–65611P. International Society for Optics and Photonics, 2007.

[Fra15] UAV France. Regulation. 2015. 2015.

[FTS10] Zubair Muhammad Fadlullah, Tarik Taleb, and Marcus Schöller. Combating against security attacks against mobile ad hoc networks (manets). *Security of Self-Organizing Networks : MANET, WSN, WMN, VANET*, 173, 2010.

[FW12] Rachel L Finn and David Wright. Unmanned aircraft systems : Surveillance, ethics and privacy in civil applications. *Computer Law & Security Review*, 28(2) :184–194, 2012.

[GA05] Lei Guang and Chadi Assi. On the resiliency of mobile ad hoc networks to mac layer misbehavior. In *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 160–167. ACM, 2005.

[Gan10] Reto Gantenbein. *VirtualMesh : An Emulation Framework For Wireless Mesh Networks in OMNeT++*. PhD thesis, University of Bern, 2010.

[GAY07] Lei Guang, Chadi Assi, and Yinghua Ye. Dream : A system for detection and reaction against mac layer misbehavior in ad hoc networks. *Computer communications*, 30(8) :1841–1853, 2007.

[GGG16] Vikrant Gokhale, SK Ghosh, and Arobinda Gupta. Classification of attacks on wireless mobile ad hoc networks and vehicular ad hoc networks. *Security of Self-Organizing Networks*, page 195, 2016.

[GH03] Henri Gilbert and Helena Handschuh. Security analysis of sha-256 and sisters. In *International Workshop on Selected Areas in Cryptography*, pages 175–193. Springer, 2003.

[GH04] Henri Gilbert and Helena Handschuh. Security analysis of sha-256 and sisters. In *Selected areas in cryptography*, pages 175–193. Springer, 2004.

[GJV15] Lav Gupta, Raj Jain, and Gabor Vaszkun. Survey of important issues in uav communication networks. 2015.

[GR93] Andrew V Goldberg and Tomasz Radzik. A heuristic improvement of the bellman-ford algorithm. *Applied Mathematics Letters*, 6(3) :3–6, 1993.

[Gra] T Graf. Netlink library (libnl). 2011. *Dostupné z : http ://www. infradead. org/˜ tgr/libnl/doc/core. html*.

[GSV10] Anuj K Gupta, Harsh Sadawarti, and Anil K Verma. Performance analysis of aodv, dsr & tora routing protocols. *International Journal of Engineering and Technology*, 2(2) :226, 2010.

[Gui14] Getting Started Guide. R2014b. mathworks, 2014.

[HB07] Vance Hilderman and Tony Baghi. *Avionics certification : a complete guide to DO-178 (software), DO-254 (hardware).* Avionics Communications, 2007.

[HE04] Lingxuan Hu and David Evans. Using directional antennas to prevent wormhole attacks. In *NDSS*, 2004.

[HFB09] Jerome Harri, Fethi Filali, and Christian Bonnet. Mobility models for vehicular ad hoc networks : a survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 11(4) :19–41, 2009.

[HH99] B Harris and R Hunt. Tcp/ip security threats and attack methods. *Computer Communications*, 22(10) :885–897, 1999.

[HJP03] Yih-Chun Hu, David B Johnson, and Adrian Perrig. Sead : Secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad hoc networks*, 1(1) :175–192, 2003.

[HLZF15] Sabine Hauert, Severin Leven, Jean-Christophe Zufferey, and Dario Floreano. The swarming micro air vehicle network (smavnet) project, 2015.

[HMBT07] MT Hyland, Barry E Mullins, Rusty O Baldwin, and Michael A Temple. Simulation-based performance evaluation of mobile ad hoc routing protocols in a swarm of unmanned aerial vehicles. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 249–256. IEEE, 2007.

[HNA11] Ayman A Hanafy, Sherif H Noureldin, and Marianne A Azer. Immunizing the saodv protocol against routing information disclosure. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, pages 330–334. IEEE, 2011.

[Hol03] Gerard Holzmann. *Spin model checker, the : primer and reference manual.* Addison-Wesley Professional, 2003.

[HP04] Yih-Chun Hu and Adrian Perrig. A survey of secure wireless ad hoc routing. *IEEE Security & Privacy*, 2(3) :28–39, 2004.

[HPFS02] Russell Housley, W Polk, Warwick Ford, and David Solo. Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. Technical report, 2002.

[HPJ03] Yih-Chun Hu, Adrian Perrig, and David B Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 30–40. ACM, 2003.

[HPJ05] Yih-Chun Hu, Adrian Perrig, and David B Johnson. Ariadne : A secure on-demand routing protocol for ad hoc networks. *Wireless networks*, 11(1-2) :21–38, 2005.

[HZSS05] Dieter Hausamann, Werner Zirnig, Gunter Schreier, and Peter Strobl. Monitoring of gas pipelines-a civil uav application. *Aircraft Engineering and Aerospace Technology*, 77(5) :352–360, 2005.

[JHM07] David Johnson, Y Hu, and D Maltz. The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4. Technical report, 2007.

[JK09] Mohit Jain and Himanshu Kandwal. A survey on complex wormhole attack in wireless ad hoc networks. In *Proceedings of International Conference on Advances in Computing, Control, and Telecommunication Technologies (ACT'09)*, pages 28–29, 2009.

[JMB+01] David B Johnson, David A Maltz, Josh Broch, et al. Dsr : The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad hoc networking*, 5 :139–172, 2001.

[JMLW08] Zhen Jiang, Junchao Ma, Wei Lou, and Jie Wu. An information model for geographic greedy forwarding in wireless ad-hoc sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 825–833. IEEE, 2008.

[JSDA12] Ahmad Y Javaid, Weiqing Sun, Vijay K Devabhaktuni, and Mansoor Alam. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 585–590. IEEE, 2012.

[Kir11] Paul Kirchner. Improved generalized birthday attack. *IACR Cryptology ePrint Archive*, 2011 :377, 2011.

[KJG12] Mina Vajed Khiavi, Shahram Jamali, and Sajjad Jahanbakhsh Gudakahriz. Performance comparison of aodv, dsdv, dsr and tora routing protocols in manets. *International Research Journal of Applied and Basic Sciences*, 3(7) :1429–1436, 2012.

[KJT08] Liu Kesheng, Zhang Jun, and Zhang Tao. The clustering algorithm of uav networking in near-space. In *Antennas, Propagation and EM Theory, 2008. ISAPE 2008. 8th International Symposium on*, pages 1550–1553. IEEE, 2008.

[KK00] Brad Karp and Hsiang-Tsung Kung. Gpsr : Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[KNM08] Rashid Hafeez Khokhar, Md Asri Ngadi, and Satria Mandala. A review of current routing attacks in mobile ad hoc networks. *International Journal of Computer Science and Security*, 2(3) :18–29, 2008.

[KNN+07] Bounpadith Kannhavong, Hidehisa Nakayama, Yoshiaki Nemoto, Nei Kato, and Abbas Jamalipour. A survey of routing attacks in mobile ad hoc networks. *IEEE Wireless Communications*, 14(5) :85–91, 2007.

[KNT06] Erik Kuiper and Simin Nadjm-Tehrani. Mobility models for uav group reconnaissance applications. In *2006 International Conference on Wireless and Mobile Communications (ICWMC'06)*, pages 33–33. IEEE, 2006.

[KNT11] Erik Kuiper and Simin Nadjm-Tehrani. Geographical routing with location service in intermittently connected manets. *IEEE Transactions on Vehicular Technology*, 60(2) :592–604, 2011.

[KS14] Rupinder Kaur and Parminder Singh. Black hole and greyhole attack in wireless mesh network. *Volume 3 Issue 10–October 2014*, 99(109) :41, 2014.

[KV05] Pradeep Kyasanur and Nitin H Vaidya. Selfish mac layer misbehavior in wireless networks. *IEEE transactions on mobile computing*, 4(5) :502–516, 2005.

[KWG+12] Alan Kim, Brandon Wampler, James Goppert, Inseok Hwang, and Hal Aldridge. Cyber attack vulnerabilities analysis for unmanned aerial vehicles. *Infotech@ Aerospace*, 2012.

[Lau05] Michael Lauer. Building embedded linux distributions with bitbake and openembedded. In *Proceedings of the Free and Open Source Software Developers' European Meeting (FOSDEM), Brussels, Belgium*, 2005.

[LC08] Byeong Gi Lee and Sunghyun Choi. *Broadband wireless access and local networks : mobile WiMAX and WiFi*. Artech House, 2008.

[LL12] Yan Li and Xiling Luo. Cross layer optimization for cooperative mobile adhoc uav network. *International Journal of Digital Content Technology and its Applications*, 6(18) :367, 2012.

[LLF04] Wenjing Lou, Wei Liu, and Yuguang Fang. Spread : Enhancing data confidentiality in mobile ad hoc networks. In *INFOCOM 2004. Twenty-third*

AnnualJoint Conference of the IEEE Computer and Communications Societies, volume 4, pages 2404–2413. IEEE, 2004.

[LML05] Ben Leong, Sayan Mitra, and Barbara Liskov. Path vector face routing : Geographic routing with local face information. In *Network Protocols, 2005. ICNP 2005. 13th IEEE International Conference on*, pages 12–pp. IEEE, 2005.

[LOBH07] Brian B Luu, Barry J O'Brien, David G Baran, and Rommie L Hardy. A soldier-robot ad hoc network. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pages 558–563. IEEE, 2007.

[LP99] Johan Lilius and Iván Porres Paltor. Formalising uml state machines for model checking. In *International Conference on the Unified Modeling Language*, pages 430–444. Springer, 1999.

[LP09] Abdesselam Lakehal and Ioannis Parissis. Structural coverage criteria for lustre/scade programs. *Software Testing, Verification and Reliability*, 19(2) :133–154, 2009.

[LSHK12] Yi Li, Marc St-Hilaire, and Thomas Kunz. Improving routing in networks of uavs via scoped flooding and mobility prediction. In *Wireless Days (WD), 2012 IFIP*, pages 1–6. IEEE, 2012.

[LSLY12] Lin Lin, Qibo Sun, Jinglin Li, and Fangchun Yang. A novel geographic position mobility oriented routing strategy for uavs. *Journal of Computational Information Systems*, 8(2) :709–716, 2012.

[LSWY12] Lin Lin, Qibo Sun, Shangguang Wang, and Fangchun Yang. A geographic mobility prediction routing protocol for ad hoc uav network. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 1597–1602. IEEE, 2012.

[LZL13] Jun Li, Yifeng Zhou, and Lisa Lamont. Communication architectures and protocols for networking unmanned aerial vehicles. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 1415–1420. IEEE, 2013.

[LZW+09] Qing Li, Meiyuan Zhao, Jesse Walker, Yih-Chun Hu, Adrian Perrig, and Wade Trappe. Sear : a secure efficient ad hoc on demand routing protocol for wireless networks. *Security and Communication Networks*, 2(4) :325–340, 2009.

[MA15] Bastien Mancini and Jean-François Aumont. Mapping quickly and with accuracy. *GeoInformatics*, 18(6) :6, 2015.

[Mac99] Joseph Macker. Mobile ad hoc networking (manet) : Routing protocol performance issues and evaluation considerations. 1999.

[MAMS11] Nicholas Martin, Yamin Al-Mousa, and Nirmala Shenoy. An integrated routing and medium access control framework for surveillance networks of mobile devices. In *International Conference on Distributed Computing and Networking*, pages 315–327. Springer, 2011.

[Man14] Bastien Mancini. The great reconnaissance : A uav project in niger. *GeoInformatics*, 17(6) :6, 2014.

[Mar09] Douglas Marshall. Unmanned aerial systems and international civil aviation organization regulations. *NDL Rev.*, 85 :693, 2009.

[MB15] Koppány Máthé and Lucian Buşoniu. Vision and control for uavs : A survey of general methods and of inexpensive platforms for infrastructure inspection. *Sensors*, 15(7) :14887–14916, 2015.

[MG98] Cheryl Madson and Rob Glenn. The use of hmac-md5-96 within esp and ah. 1998.

[MGFM06] Giannis F Marias, Panagiotis Georgiadis, D Flitzanis, and K Mandalas. Cooperation enforcement schemes for manets : A survey. *Wireless Communications and Mobile Computing*, 6(3) :319–332, 2006.

[Mih07] Preda Mihailescu. The fuzzy vault for fingerprints is vulnerable to brute force attack. *arXiv preprint arXiv :0708.2974*, 2007.

[Mil85] Dave L Mills. Network time protocol (ntp). *Network*, 1985.

[Mis04] Ajay R Mishra. *Fundamentals of cellular network planning and optimisation : 2G/2.5 G/3G... evolution to 4G*. John Wiley & Sons, 2004.

[MLD+13] Yannick Moy, Emmanuel Ledinot, Hervé Delseny, Virginie Wiels, and Benjamin Monate. Testing or formal verification : Do-178c alternatives and industrial experience. *IEEE software*, 30(3) :50–57, 2013.

[MML16a] Jean-Aimé Maxa, Mohamed Slim Ben Mahmoud, and Nicolas Larrieu. Extended verification of secure uaanet routing protocol. In *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, pages 1–16. IEEE, 2016.

[MML16b] Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, and Nicolas Larrieu. How to improve routing protocol security in a rpas swarm ? In *4th AETOS International Workshop on" Research Challenges for Future RPAS/UAV Systems"*, 2016.

[MML17a] Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, and Nicolas Larrieu. Survey on uaanet routing protocols and network security challenges. *Ad Hoc & Sensor Wireless Networks*, 2017.

[MML17b] Jean-Aimé Maxa, Mohamed-Slim Ben Mahmoud, and Nicolas Larrieu. Survey on uaanet routing protocols and network security challenges. In *Adhoc - Sensors Networks*, 2017.

[MMS09] Misagh Mohammadizadeh, Ali Movaghar, and Seyad Mohammad Safi. Seaodv : secure efficient aodv routing protocol for manets networks. In *Proceedings of the 2nd International Conference on Interaction Sciences : Information Technology, Culture and Human*, pages 940–944. ACM, 2009.

[MNS08] Viren Mahajan, Maitreya Natu, and Adarshpal Sethi. Analysis of wormhole intrusion attacks in manets. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008.

[MRL15] Jean-Aimé Maxa, Gilles Roudiere, and Nicolas Larrieu. Emulation-based performance evaluation of routing protocols for uaanets. In *Communication Technologies for Vehicles*, pages 227–240. Springer, 2015.

[MWH01] Martin Mauve, Jorg Widmer, and Hannes Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE network*, 15(6) :30–39, 2001.

[NGGC12] Yanchao Niu, Deyun Gao, Shuai Gao, and Ping Chen. A robust localization in wireless sensor networks against wormhole attack. *JNW*, 7(1) :187–194, 2012.

[NMSK01] Hanae Nozaki, Masahiko Motoyama, Atsushi Shimbo, and Shinichi Kawamura. Implementation of rsa algorithm based on rns montgomery multiplication. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 364–376. Springer, 2001.

[NN08] Hoang Lan Nguyen and Uyen Trang Nguyen. A study of different types of attacks on multicast in mobile ad hoc networks. *Ad Hoc Networks*, 6(1) :32–46, 2008.

[NPW02] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL : a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.

[NS11] René Ndoundam and Juvet Karnel Sadie. Collision-resistant hash function based on composition of functions. *arXiv preprint arXiv :1108.1478*, 2011.

[omn14] OMNeT++, 2014.

[OP05] Bob O'hara and Al Petrick. *IEEE 802.11 handbook : a designer's companion*. IEEE Standards Association, 2005.

[OTBL02] Richard G Ogier, Fred L Templin, Bhargav Bellur, and Mark G Lewis. Topology broadcast based on reverse-path forwarding (tbrpf). *draft-ietf-manettbrpf-05. txt*, 1, 2002.

[Pap03] UAV Paparazzi. Url http ://paparazzi. enac. fr/wiki, 2003.

[PB01] Charles E Perkins and Pravin Bhagwat. Dsdv routing over a multihop wireless network of mobile computers. In *Ad hoc networking*, pages 53–74. Addison-Wesley Longman Publishing Co., Inc., 2001.

[PBP+10] Reema Patel, Bhavesh Borisaniya, Avi Patel, Dhiren Patel, Muttukrishnan Rajarajan, and Andrea Zisman. Comparative analysis of formal model checking tools for security protocol verification. In *International Conference on Network Security and Applications*, pages 152–163. Springer, 2010.

[PBRD+03a] Charles Perkins, E Belding-Royer, Samir Das, et al. Rfc 3561-ad hoc on-demand distance vector (aodv) routing. *Internet RFCs*, pages 1–38, 2003.

[PBRD03b] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003.

[PD12] Hrituparna Paul and Prodipto Das. Performance evaluation of manet routing protocols. *International Journal of Computer Science Issues (IJCSI)*, 9(4), 2012.

[Pet11] Fabien AP Petitcolas. Kerckhoffs' principle. In *Encyclopedia of cryptography and security*, pages 675–675. Springer, 2011.

[PGC00] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing : A routing scheme for ad hoc wireless networks. In *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, volume 1, pages 70–74. IEEE, 2000.

[PH02] Panos Papadimitratos and Zygmunt J Haas. Secure routing for mobile ad hoc networks. In *the SCS Commnication Networks and Distributed Systems Modeling and Simulation Conference (CNDS), San Antonio, TX, January 27-31, 2002*, pages 193–204, 2002.

[PJ03] A Perrig and David B Johnson. Packet leashes : A defense against wormhole attacks in wireless ad hoc networks. In *IEEE INFOCOM*, pages 1976–1986, 2003.

[Pon08] Marie-Noëlle Pons. Analyse du cycle de vie : Comment choisir un logiciel. *Techniques de l'ingénieur. Environnement*, 1(G6350), 2008.

[Pot12] Bill Potter. Complying with do-178c and do-331 using model-based design. *MathWorks, Inc. 12AEAS-0090*, 2012.

[PPP15] Anal Patel, Nimisha Patel, and Rajan Patel. Defending against wormhole attack in manet. In *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, pages 674–678. IEEE, 2015.

[PST+02] Adrian Perrig, Robert Szewczyk, Justin Douglas Tygar, Victor Wen, and David E Culler. Spins : Security protocols for sensor networks. *Wireless networks*, 8(5) :521–534, 2002.

[PT03] Adrian Perrig and JD Tygar. Tesla broadcast authentication. In *Secure Broadcast Communication*, pages 29–53. Springer, 2003.

[PZV+06] Matthew Pirretti, Sencun Zhu, Narayanan Vijaykrishnan, Patrick McDaniel, Mahmut Kandemir, and Richard Brooks. The sleep deprivation attack in sensor networks : Analysis and methods of defense. *International Journal of Distributed Sensor Networks*, 2(3) :267–287, 2006.

[RBB09] AJ Roscoe, SM Blair, and Graeme M Burt. Benchmarking and optimisation of simulink code using real-time workshop and embedded coder for inverter and microgrid control applications. In *Universities Power Engineering Conference (UPEC), 2009 Proceedings of the 44th International*, pages 1–5. IEEE, 2009.

[RC05] Ashish Raniwala and Tzi-cker Chiueh. Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 2223–2234. IEEE, 2005.

[RGDW10] Sebastian Rohde, Niklas Goddemeier, Kai Daniel, and Christian Wietfeld. Link quality dependent mobility strategies for distributed aerial sensor networks. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1783–1787. IEEE, 2010.

[RHA04] Maxim Raya, Jean-Pierre Hubaux, and Imad Aad. Domino : a system to detect greedy behavior in ieee 802.11 hotspots. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 84–97. ACM, 2004.

[Rie13] Leanna Rierson. *Developing safety-critical software : a practical guide for aviation software and DO-178C compliance*. CRC Press, 2013.

[Riv92] Ronald Rivest. The md5 message-digest algorithm. 1992.

[RJB04] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified modeling language reference manual, the*. Pearson Higher Education, 2004.

[RKT13] Stefano Rosati, Karol Krużelecki, and Louis Traynard. Speed-aware routing for uav ad-hoc networks. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 1367–1373. IEEE, 2013.

[RMA+08] Shravan Rayanchu, Arunesh Mishra, Dheeraj Agrawal, Sharad Saha, and Suman Banerjee. Diagnosing wireless packet losses in 802.11 : Separating collision from weak signal. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[RR12] Laurent Reynaud and Tinku Rasheed. Deployable aerial communication networks : challenges for futuristic applications. In *Proceedings of the 9th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 9–16. ACM, 2012.

[Sah14] Ozgur Koray Sahingoz. Networking models in flying ad-hoc networks (fanets) : Concepts and challenges. *Journal of Intelligent & Robotic Systems*, 74(1-2) :513–527, 2014.

[Sal13] Arto Salomaa. *Public-key cryptography*. Springer Science & Business Media, 2013.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3) :161–174, 1991.

[SCZ11] Hangguan Shan, Ho Ting Cheng, and Weihua Zhuang. Cross-layer cooperative mac protocol in distributed wireless networks. *IEEE Transactions on Wireless Communications*, 10(8) :2603–2615, 2011.

[Sen10] Jaydip Sen. *Reputation-and trust-based systems for wireless self-organizing networks*. Aurbach Publications, CRC Press, USA, 2010.

[SGB11] Thomas Staub, Reto Gantenbein, and Torsten Braun. Virtualmesh : an emulation framework for wireless mesh and ad hoc networks in omnet++. *Simulation*, 87(1-2) :66–81, 2011.

[SH10] Lu Song and Ting-Lei Huang. A summary of key technologies of ad hoc networks with uav node. In *Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on*, pages 944–949. IEEE, 2010.

[Ski90] S Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics : Combinatorics and Graph Theory with Mathematica, Reading, MA : Addison-Wesley*, pages 225–227, 1990.

[SKKS11] Till Steinbach, Hermand Dieumo Kenfack, Franz Korf, and Thomas C Schmidt. An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 375–382. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.

[SL12] Nanxiang Shi and Xiling Luo. A novel cluster-based location-aided routing protocol for uav fleet networks. *International Journal of Digital Content Technology and its Applications*, 6(18) :376, 2012.

[SLD+05] Kimaya Sanzgiri, Daniel LaFlamme, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M Belding-Royer. Authenticated routing for ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 23(3) :598–610, 2005.

[SN93] Matlab Simulink and MA Natick. The mathworks, 1993.

[SSHK+12] Rostam Shirani, Marc St-Hilaire, Thomas Kunz, Yifeng Zhou, Jun Li, and Louise Lamont. Combined reactive-geographic routing for unmanned aeronautical ad-hoc networks. In *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 820–826. IEEE, 2012.

[ST10] Achudhan Sivakumar and Colin Keng-Yan Tan. Uav swarm coordination using cooperative control for establishing a wireless communications backbone. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems : volume 3-Volume 3*, pages 1157–1164. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[SZ08] Hang Su and Xi Zhang. Cross-layer based opportunistic mac protocols for qos provisionings over cognitive radio wireless networks. *IEEE Journal on Selected Areas in Communications*, 26(1) :118–129, 2008.

[Szy04] Michael Szydlo. Merkle tree traversal in log space and time. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 541–554. Springer, 2004.

[TB15] Samil Temel and Ilker Bekmezci. Lodmac : Location oriented directional mac protocol for fanets. *Computer Networks*, 83 :76–84, 2015.

[tcp14] tcpdump & libpcap, 2014.

[TS07] Latha Tamilselvan and V Sankaranarayanan. Prevention of blackhole attack in manet. In *The 2nd International Conference on Wireless Broadband and Ultra Wideband Communications (AusWireless 2007)*, pages 21–21. IEEE, 2007.

[Var13] Antoine Varet. *Conception, Mise en œuvre et évaluation d'un routeur embarqué pour l'avionique de nouvelle génération*. PhD thesis, INSA de Toulouse, 2013.

[VL12] Antoine Varet and Nicolas Larrieu. Design and development of an embedded aeronautical router with security capabilities. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2012*, pages E1–1. IEEE, 2012.

[VMA14] Danil S Vasiliev, Daniil S Meitis, and Albert Abilov. Simulation-based comparison of aodv, olsr and hwmp protocols for flying ad hoc networks. In *International Conference on Next Generation Wired/Wireless Networking*, pages 245–252. Springer, 2014.

[Wat08] Jon Watson. Virtualbox : bits and bytes masquerading as machines. *Linux Journal*, 2008(166) :1, 2008.

[WCWC07] Bing Wu, Jianmin Chen, Jie Wu, and Mihaela Cardei. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless network security*, pages 103–135. Springer, 2007.

[WGWW10] Wei Wang, Xiaohong Guan, Beizhan Wang, and Yaping Wang. A novel mobility model based on semi-random circular movement in mobile ad hoc networks. *Information Sciences*, 180(3) :399–413, 2010.

[XK01] Dongbin Xiu and George Em Karniadakis. A semi-lagrangian high-order method for navier–stokes equations. *Journal of computational physics*, 172(2) :658–684, 2001.

[YCBE10] Evşen Yanmaz, Carmelo Costanzo, Christian Bettstetter, and Wilfried Elmenreich. A discrete stochastic process for coverage analysis of autonomous uav networks. In *2010 IEEE Globecom Workshops*, pages 1777–1782. IEEE, 2010.

[YDZZ05] Ping Yi, Zhoulin Dai, Shiyong Zhang, and Yiping Zhong. A new routing attack in mobile ad hoc networks. *International Journal of Information Technology*, 11(2) :83–94, 2005.

[YK04] Seung Yi and Robin Kravets. Moca : Mobile certificate authority for wireless ad hoc networks. 2004.

[YLY+04] Hao Yang, Haiyun Luo, Fan Ye, Songwu Lu, and Lixia Zhang. Security in mobile ad hoc networks : challenges and solutions. *IEEE wireless communications*, 11(1) :38–47, 2004.

[YZY13] Lan Yao, Zhibin Zhao, and Ge Yu. Detecting wormhole attacks in wireless sensor networks using hop count analysis. *International Journal on Smart Sensing & Intelligent Systems*, 6(1), 2013.

[ZA02a] Manel Guerrero Zapata and Nadarajah Asokan. Securing ad hoc routing protocols. In *Proceedings of the 1st ACM workshop on Wireless security*, pages 1–10. ACM, 2002.

[ZA02b] Eustathia Ziouva and Theodore Antonakopoulos. Csma/ca performance under high traffic conditions : throughput and delay analysis. *Computer communications*, 25(3) :313–321, 2002.

[Zap02] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector routing. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(3) :106–107, 2002.

[Zap08] Manel Guerrero Zapata. Key management in wireless mesh networks. *book chapter in : Security in Wireless Mesh Networks, Y. Zhang et al.(eds.)*, pages 323–346, 2008.

[ZR03] Jing Zhu and Sumit Roy. Mac for dedicated short range communications in intelligent transport system. *IEEE Communications Magazine*, 41(12) :60–67, 2003.

[ZR13] Jun Du Zhai, Zhongqiang and Yong Ren. The application and improvement of temporally ordered routing algorithm in swarm network with unmanned aerial vehicle nodes. In *ICWMC 2013, The Ninth International Conference on Wireless and Mobile Communications*, 2013.

[ZT11] Xin Zhou and Xiaofei Tang. Research and implementation of rsa algorithm for encryption and decryption. In *Strategic Technology (IFOST), 2011 6th International Forum on*, volume 2, pages 1118–1121. IEEE, 2011.

[ZWL+14] Yi Zheng, Yuwen Wang, Zhenzhen Li, Li Dong, Yu Jiang, and Hong Zhang. A mobility and load aware olsr routing protocol for uav mobile ad-hoc networks. In *Information and Communications Technologies (ICT 2014), 2014 International Conference on*, pages 1–7. IET, 2014.

[ZWN04] Yihong Zhou, Dapeng Wu, and Scott M Nettles. Analyzing and preventing MAC-layer denial of service attacks for stock 802.11 systems. In *Workshop on BWSA, Broadnets*, 2004.

[ZZ11] Chunhua Zang and Shouhong Zang. Mobility prediction clustering algorithm for uav networking. In *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, pages 1158–1161. IEEE, 2011.

[ZZZ08] Ke Zhang, Wei Zhang, and Jia-Zhi Zeng. Preliminary study of routing and date integrity in mobile ad hoc uav network. In *2008 International Conference on Apperceiving Computing and Intelligence Analysis*, 2008.

[BS T13] İlker Bekmezci, Ozgur Koray Sahingoz, and Şamil Temel. Flying ad-hoc networks (fanets) : A survey. *Ad Hoc Networks*, 11(3) :1254 – 1270, 2013.