Sécurité des RFIDs actifs et applications

Amina Cherif

Thèse en cotutelle présentée pour obtenir le grade de docteur

($3^{\text{ème}}$ cycle LMD)

Université Mouloud Mammeri de Tizi Ouzou

Université de Limoges

Spécialité informatique

Option : Réseaux, Mobilité et Systèmes Embarqués

Sécurité des RFIDs actifs et applications

Par : **Amina CHERIF**

Sous la codirection de Malika BELKADI et Damien SAUVERON

Malika BELKADI, Maître de Conférences rang A, Université Mouloud Mammeri de Tizi Ouzou

Damien SAUVERON, Maître de Conférences Habilité à Diriger des Recherches, Université de Limoges

Membres du jury:

Président, Rapporteur interne : Mustapha Lalam, Prof. des Universités, Université Mouloud-Mammeri de Tizi Ouzou

Rapporteur externe : Serge Chaumette, Prof. des Universités, Université de Bordeaux

Rapporteur externe : Safia Nait-Bahloul, Prof. des Universités, Université d'Oran 1

Rapporteur interne, Examinateur : Emmanuel Conchon, Maître de conférences, Université de Limoges

Date de soutenance : 15 Mars 2021

Remerciements

Ce travail a été réalisé au sein du laboratoire de recherche en informatique (LARI) de l'Université Mouloud Mammeri de Tizi-Ouzou et du laboratoire (XLIM) de l'Université de Limoges.

Je tiens à remercier mes directeurs de thèse Malika BELKADI et Damien SAUVERON. Je les remercie non-seulement pour leurs contributions au travail effectué mais aussi pour leurs accueils, leurs indulgences à mes inexpériences et d'avoir mis à ma disposition des moyens de réussite de ce travail. Cette thèse n'aurait pas pu être accomplie sans leurs appuis inconditionnels.

Je remercie aussi le directeur du laboratoire de recherche (LARI) Monsieur Mustapha Lalam pour les conseils prodigués tout au long de ces années de travail. J'associe à ces remerciements tous les membres du laboratoire LARI ainsi que les tous les enseignants du département d'informatique et les membres du laboratoire XLIM.

J'exprime toute ma gratitude envers les membres du jury : Mustapha Lalam, Professeur à l'Université Mouloud Mammeri de Tizi-Ouzou, Serge Chaumette, Professeur à l'Université de Bordeaux, Safia Nait-Bahloul, Professeur à l'Université d'Oran 1, Emmanuel Conchon, Maître de Conférences à l'Université de Limoges, qui vont me faire l'honneur de juger mon travail. J'exprime tous mes remerciements à mes amis, dont l'aide

m'a été très précieuse notamment dans les moments de doute : Sonia, Fatma, Ryma, Shahrazed et Nabila.

Je remercie mes parents qui m'ont toujours incité à apprendre, ma mère la lumière de ma vie sans elle je ne serai pas arrivée où je suis aujourd'hui, mon père qui m'a encouragé et cru en moi et mes capacités. Comment ne pas remercier mes deux chers frères Yazid et Mohammed qui m'ont aidé et soutenu tout au long de mes études supérieurs, mon cher et tendre époux Younes, pour ses encouragements et sa patience, pour avoir toujours su me rassurer dans les moments difficiles, mon fils Adam ainsi que ma belle-famille.

Je tiens à rendre hommage à mon cher oncle « Vava Mohend » qui n'appartient plus à ce monde, cette personne merveilleuse qui a fortement cru en moi, m'a encouragé et a été toujours fière de moi, reposes en paix vava mohend.

Enfin, un grand merci à celles et ceux qui ont cru en moi et dont le nom n'apparaît pas dans cette page.

Résumé

Au cours des 30 dernières années, les dispositifs RFID actifs sont passés de simples dispositifs d'identification (tags) à des nœuds autonomes qui, en prime, collectent (à partir de l'environnement ou d'autres sources) et échangent des données. En conséquence, le spectre de leurs applications s'est largement étendu, passant de la simple identification à la surveillance et à la localisation en temps réel. Ces dernières années, grâce à leurs avantages, l'utilisation de nœuds RFID actifs pour la collecte de données mobiles a suscité une attention particulière. En effet, dans la plupart des scénarios, ces nœuds sont déployés dans des environnements adverses. Les données doivent donc être stockées et transmises de manière sécurisée pour empêcher toute attaque par des adversaires actifs : même si les nœuds sont capturés, la confidentialité des données doit être assurée. Toutefois, en raison des ressources limitées des nœuds en termes d'énergie, de stockage et/ou de calcul, la solution de sécurité utilisée doit être légère.

Cette thèse est divisée en deux parties. Dans la première, nous étudierons en détail l'évolution des nœuds RFID actifs et leur sécurité. Nous présenterons ensuite, dans la seconde partie, un nouveau protocole sans serveur permettant à des MDC (collecteurs de données mobiles), par exemple des drones, de collecter en toute sécurité des données provenant de nœuds RFID actifs mobiles et statiques afin de les transmettre ultérieure-

ment à un tiers autorisé. L'ensemble de la solution proposée garantit la confidentialité des données à chaque étape (de la phase de mesure, avant la collecte des données par le MDC, une fois les données collectées par le MDC et lors de la livraison finale), tout en satisfaisant les exigences de faible consommation des ressources (calcul, mémoire, etc.) des entités impliquées. Pour évaluer l'adéquation du protocole aux exigences de performance, nous l'implémenterons sur les dispositifs de sécurité les plus limités en ressources c'est-à-dire à base de processeur de cartes à puce pour prouver qu'il est efficace même dans les pires conditions. De plus, pour prouver que le protocole satisfait aux exigences de sécurité, nous l'analyserons à l'aide de jeux de sécurité et également d'outils de vérification formelle : AVISPA et ProVerif.

Mots clés : *Nœuds RFID actifs, Protocole de collecte de données mobiles, Protocole sans serveur, Cryptographie légère, Confidentialité des données.*

Abstract

Over the 30 last years, active RFID devices have evolved from nodes dedicated to identification to autonomous nodes that, in addition, sense (from environment or other sources) and exchange data. Consequently, the range of their applications has rapidly grown from identification only to monitoring and real time localisation. In recent years, thanks to their advantages, the use of active RFID nodes for mobile data collection has attracted significant attention. However, in most scenarios, these nodes are unattended in an adverse environments, so data must be securely stored and transmitted to prevent attack by active adversaries: even if the nodes are captured, data confidentiality must be ensured. Furthermore, due to the scarce resources available to nodes in terms of energy, storage and/or computation, the used security solution has to be lightweight.

This thesis is divided in two parts. In the first, we will study in details the evolution of active RFID nodes and their security. We will then, present, in the second part, a new serverless protocol to enable MDCs (Mobile Data Collectors), such as drones, to securely collect data from mobile and static Active RFID nodes and then deliver them later to an authorized third party. The whole solution ensures data confidentiality at each step (from the sensing phase, before data collection by the MDC, once data have been collected by MDC, and during final delivery) while fulfilling the lightweight requirements

for the resource-limited entities involved. To assess the suitability of the protocol against the performance requirements, we will implement it on the most resource-constrained secure devices to prove its efficiency even in the worst conditions. In addition, to prove the protocol fulfills the security requirements, we will analyze it using security games and we will also formally verify it using the AVISPA and ProVerif tools.

Key Words: *Active RFID nodes, Mobile data collection protocol, Serverless protocol, Lightweight cryptography, Data confidentiality.*

Table des matières

Table des figures

Liste des tableaux

Glossaire

APDU « Application Protocol Data Unit ».

AR « Active RFID » ou RFID actif.

BS « Base Station » ou station de base parfois appelé « sink » ou puits.

CSP « Communicating Sequential Processes ».

EAS « Electronic Article Surveillance » ou code de surveillance électronique.

ECC « Elliptic Curve Cryptography » ou cryptographie à base de courbes elliptiques.

EPC « Electronic Product Code » ou code de produit électronique.

FDR « Failure Divergence Refinements ».

GPS « Global Positioning System » ou système de positionnement global.

HLPSL « High Level Protocol Specification Language ».

HMAC « Hash-based Message Authentication Codes ».

IdO Internet des Objets, aussi nommé IoT en anglais.

IdOM Internet des Objets Mobile, aussi nommé M-IoT en anglais.

IF « Intermediate Format » ou format intermédiaire.

IFF « Identification Friend or Foe » ou identification ami ou ennemi.

IoT « Internet of Things ».

ISM « Industrial, Scientific and Medical » ou industriel, scientifique et médical.

KDC « Keys Distribution Center » ou centre de distribution des clés.

KGC « Key Generation Center » ou centre de génération de clés.

M-IoT « Mobile-IoT » ou internet des objets mobiles.

MAR « Mobile Active RFID » ou RFID actif mobile.

MDC « Mobile Data Collector » ou collecteur de données mobile.

MiTM « Man-in-The-Middle » ou homme du milieu.

NFC « Near Field Communication » ou communication en champ proche.

NFC-WISP « Near Field Communication-Wireless Identification and Sensing Platform ».

P2P « Peer-to-Peer » ou pair-à-pair.

RF Radio-Fréquence.

RFID « Radio Frequency IDentification » ou identification par radio-fréquence.

RTLS « Real-Time Localisation System » ou système de localication temps réel.

SAR « Static Active RFID » ou RFID actif statique.

STCP « Secure and Trusted Channel Protocol ».

TEM « Trusted Execution Module ».

TPDU « Transmission Protocol Data Unit ».

UAV « Unmanned Aerial Vehicle » ou véhicule volant sans pilote, souvent appelé drone par abus de langage.

UGV « Unmanned Ground Vehicle » ou véhicule terrestre sans pilote, souvent appelé robot par abus de langage.

UMV « Unmanned Marine Vehicle » ou véhicule marin sans pilote.

UUV « Unmanned Underwater Vehicle » ou véhicule sous-marin sans pilote.

WISP « Wireless Identification and Sensing Platform ».

WSN « Wireless Sensor Network » ou réseau de capteurs sans fil.

Introduction

Contexte et motivation

Dans le contexte sans cesse croissant de l'IdO (Internet des Objets, aussi connu sous le terme IoT), des dispositifs informatiques toujours plus nombreux et dotés de diverses fonctionnalités produisent des données (à partir de l'environnement ou d'autres sources). La plupart de ces dispositifs partagent les mêmes caractéristiques à savoir qu'ils :

— communiquent via une technologie sans fil du type RF (Radio-Fréquence) ;

— embarquent une source d'énergie ou un module de récupération d'énergie ;

— offrent une identification automatique unique de l'objet/personne (ou même de l'environnement de ce dernier) ;

— peuvent embarquer d'autres modules proposant des fonctionnalités supplémentaires.

Dans l'IdO, ces dispositifs sont présents sous des formes très hétérogènes : capteurs sans fil, balises intelligentes ou même drones (aussi appelés UAVs pour « Unmanned Aerial Vehicles »). De part leur caractéristiques, tous ces dispositifs, qui mesurent, collectent, stockent, traitent et transmettent des données à l'utilisateur final, peuvent être référencés sous le terme de nœuds RFID actifs ou ARs (« Active RFID »). De nos jours, ils sont

utilisés dans de nombreux domaines et en particulier pour de nouvelles applications telles que la collecte de données mobile, la surveillance d'objet/personne ou la localisation en temps réel.

Dans de nombreux scénarios, pour ces applications, la sécurité des données captées tant lors du stockage que de la communication est une exigence primordiale. Cependant, les nœuds RFID actifs sont souvent déployés dans des environnements hostiles et sans surveillance ce qui les rend potentiellement vulnérables à plusieurs types d'attaques. De plus, ils sont souvent limités en termes de capacité de calcul, de mémoire et d'énergie, ce qui contrarie encore plus l'objectif d'assurer une sécurité élevée.

Pour mieux comprendre l'importance de ces problématiques de sécurité, considérons le scénario militaire où plusieurs nœuds RFID actifs (dotés de capteurs) sont déployés sur le territoire d'un adversaire distant pour capter des données et où des drones sont chargés de collecter les données mesurées par les nœuds RFID actifs afin de les transmettre à la station de base à l'issue de leurs missions. Une attaque physique visant à détruire les nœuds RFID actifs a moins de conséquences pour la personne les ayant déployés qu'une attaque ayant pour objectif la capture de ces nœuds pour en extraire leurs données et clés de chiffrement ; en effet l'adversaire pourrait découvrir qui le surveille et pourquoi. Au delà de ce cas particulier, il est donc essentiel de comprendre quelles sont les menaces et attaques associées à l'exploitation des RFID actifs dans les différents environnements et comment ces menaces et attaques peuvent être contrées.

Afin d'assurer un haut niveau de sécurité, les mécanismes et protocoles à utiliser doivent être efficaces tout en respectant les ressources limitées des nœuds, c'est-à-dire qu'ils doivent être *légers*. De plus, dans de nombreuses applications la station de base est loin de la zone de déploiement des nœuds (et donc pas joignable) ; ces derniers doivent alors fonctionner en mode *sans serveur* (« serverless »). Dans d'autres applications, pour des

raisons d'économie d'énergie, le mode serverless permet d'éviter de dépenser l'énergie pour réaliser des transmissions à longue distance. Quel que soit le contexte, la confidentialité des données de bout en bout (des nœuds à la station de base) doit être assurée.

Si plusieurs protocoles de sécurité pour les systèmes RFID existent dans la littérature, il a souvent été démontré que beaucoup sont sujets à des erreurs. Il est donc primordial de *vérifier* si le protocole utilisé dans une application répond vraiment aux propriétés souhaitées mais aussi son exactitude.

La problématique de cette thèse est donc une fois définies les caractéristiques intrinsèques des nœuds RFID actifs actuels, de proposer une solution de sécurité flexible au travers d'un protocole pouvant s'adapter à un large cadre d'applications. En effet, puisque le protocole aura été conçu, développé et vérifié tant du point de vue sécuritaire, formellement, que du point de vue performance dans un des contextes applicatifs les plus exigeant, il pourra également être utilisé et adapté à des contextes applicatifs moins contraints.

Objectifs de la thèse

Afin de faire face aux problèmes et défis susmentionnés, nous nous sommes fixés les objectifs suivants :

— **Objectif A :** Établir un état de l'art de l'évolution des RFID actifs permettant de définir les caractéristiques d'un nœud RFID actif et de les classifier selon leurs types, les technologies utilisées et les applications actuelles.

— **Objectif B :** (1) Étudier les problématiques de la sécurité des systèmes RFID en général (passif et actif) et les solutions proposées dans la littérature ainsi que la possibilité de les appliquer ou de les adapter sur les systèmes RFID actifs actuels. (2) Étudier l'importance de la vérification formelle des protocoles de sécurité des

systèmes RFID.

— **Objectif C :** Proposer un protocole de sécurité sans serveur pour assurer la confidentialité des données captées par des nœuds RFID actifs et collectées par un MDC (« Mobile Data Collector ») dans un contexte applicatif exigeant un très haut niveau de sécurité. Ainsi, ce protocole pourra être dégradé afin d'être utilisé dans des contextes applicatifs moins exigeants.

— **Objectif D :** Présenter des analyses de sécurité informelles basées sur des jeux de sécurité et une analyse formelle pour prouver que le protocole proposé satisfait réellement les propriétés de sécurité attendues.

— **Objectif E :** Procéder à une évaluation expérimentale des performances du protocole proposé par son implémentation sur les dispositifs de sécurité les plus contraints en ressources.

Organisation de la thèse

Cette thèse est organisée en deux parties : la première traite essentiellement de l'état de l'art alors que la seconde détaille ma contribution majeure à la sécurisation des RFID actifs.

La première est constituée de trois chapitres.

— Dans le chapitre 1, nous retraçons l'évolution des RFID actifs depuis leur apparition jusqu'à ce jour et les défis auxquels ils seront confrontés pour leurs évolutions futures. Tout d'abord, nous présentons l'historique des systèmes RFID en général pour préciser la différence entre les RFID passifs et actifs et comprendre quels sont les facteurs qui, dans le passé, ont freiné l'évolution et le développement des RFID actifs. Ensuite, nous présentons ce qu'est, aujourd'hui, devenue cette technologie

en répondant aux questions suivantes : Qu'est-ce qu'un nœud RFID actif ? Comment fonctionne-t-il en terme d'énergie, de technologie de communication, etc. ? Puis, dans la suite, nous classifions différents types de nœuds RFID actifs selon les critères étudiés. Nous présentons aussi leurs nouvelles applications. Enfin, nous discutons des nouveaux défis qui les attendent à l'avenir. Ce chapitre permet de répondre à l'**objectif A**.

— Le chapitre 2 est consacré à la sécurité des RFID actifs. Pour atteindre l'**objectif B**, nous étudions la sécurité des systèmes RFID passifs et actifs. En se basant sur les travaux de la littérature, nous tirons des conclusions sur l'état actuel de la sécurité des RFID actifs.

— Dans le chapitre 3, nous présentons l'importance de la vérification formelle des protocoles de sécurité RFID afin de finaliser l'**objectif B**. Nous illustrons celle-ci sur un protocole concret dont nous avons participé au développement en appliquant deux outils de vérification formelle.

La seconde partie comporte deux chapitres et détaille ma contribution majeure : la proposition d'une solution de sécurité pour les RFID actifs.

— Dans le chapitre 4, nous présentons notre protocole qui consiste à garantir la confidentialité des données captées par des nœuds RFID actifs et collectées par un nœud MDC même en cas de capture des nœuds tout en respectant les ressources limitées de ces nœuds (le protocole proposé est léger). Nous commençons par la description du scénario utilisé puis nous détaillons les étapes du protocole. Nous analysons ensuite sa sécurité en se basant sur le modèle d'attaquant basé sur des jeux de sécurité et nous procédons à une vérification formelle à l'aide des outils AVISPA et ProVerif. Ce chapitre permet de répondre aux **objectifs C et D** que nous nous étions fixés.

— Le chapitre 5 présente les résultats de l'implémentation du protocole sur des dis-

positifs très limités en ressources (cartes à puce) pour évaluer son adéquation aux exigences de performance. Ce chapitre est la réponse à l'**objectif E**.

Le mémoire se termine par une conclusion et des perspectives de recherche.

Publications

Dans le cadre de mes travaux de thèse, j'ai réalisé plusieurs publications en conférences et en revue.

La publication la plus significative de mes travaux est celle acceptée en 2018 et publiée en 2019 dans *ACM Transactions on Embedded Computing Systems* : **"A Lightweight & Secure Data Collection Serverless Protocol Demonstrated in an Active RFIDs scenario"** dont je suis première auteure et que j'ai co-écrit avec mes directeurs de thèse Malika Belkadi et Damien Sauveron.

Abstract: "In the growing Internet of Things context, thousands of computing devices with various functionalities are producing data (from environmental sensors or other sources). However, they are also collecting, storing, processing and transmitting data to eventually communicate them securely to third parties (e.g. owners of devices or cloud data storage). The deployed devices are often battery-powered mobile or static nodes equipped with sensors and/or actuators and they communicate using wireless technologies. Examples include unmanned aerial vehicles, wireless sensor nodes, smart beacons, and wearable health objects. Such resource-constrained devices include Active RFID (Radio Frequency IDentification) nodes and these are used to illustrate our proposal. In most scenarios, these nodes are unattended in an adverse environment, so data confidentiality must be ensured from the sensing phase through to delivery to authorized entities: in other words, data must be securely stored and transmitted to prevent attack by active adver-

saries even if the nodes are captured. However, due to the scarce resources available to nodes in terms of energy, storage and/or computation, the proposed security solution has to be lightweight. In this paper, we propose a serverless protocol to enable MDCs (Mobile Data Collectors), such as drones, to securely collect data from mobile and static Active RFID nodes and then deliver them later to an authorized third party. The whole solution ensures data confidentiality at each step (from the sensing phase, before data collection by the MDC, once data have been collected by MDC, and during final delivery) while fulfilling the lightweight requirements for the resource-limited entities involved. To assess the suitability of the protocol against the performance requirements, it was implemented on the most resource-constrained devices to get the worst possible results. In addition, to prove the protocol fulfills the security requirements, it was analyzed using security games and also formally verified using the AVISPA and ProVerif tools."

Note : Ces travaux sont repris et détaillés dans les chapitres 4 et 5 de ce mémoire.

Ma première contribution recherche a été la communication **"Overview on Formal Verification Methods for RFID Protocols"** que j'ai présentée à *IWCA'16 (International Workshop on Cryptography and its Applications)* à Oran, que j'avais co-écrit avec mes directeurs de thèse Malika Belkadi et Damien Sauveron et dont j'étais première auteure.

Abstract: "Cryptography plays an important role to ensure security and privacy protection in RFID systems. However, due to economical reasons for market penetration of RFID technologies, tags resources (i.e. computational and memory capabilities) are highly constrained. In this context, cryptographic protocols must rely on lightweight primitives. Since lot of proposed protocols are error-prone, in this paper, we focus on formal verifi-

cation methods and study their use to detect flaws and/or verify the correctness of these RFID protocols."

Note : Ce travail a fortement orienté le déroulement de ma thèse puisque j'ai par la suite souhaité apporter des preuves de sécurité sur les différents protocoles sur lesquels j'ai travaillé. Une partie de ce travail se retrouve dans le chapitre 3.

Forte de cette expertise sur la vérification de protocoles de sécurité, j'ai contribué à plusieurs articles utilisant des RFID actifs pour en particulier écrire ou aider à la réalisation de protocoles en vue de les vérifier formellement avec les outils AVISPA et/ou ProVerif.

Ainsi, j'ai réalisé une partie de la vérification du protocole proposé dans l'article **"A Secure and Trusted Channel Protocol for UAVs Fleets"** accepté à *WISTP2017 (the 11th WISTP International Conference on Information Security Theory and Practice)* qui s'est déroulée Héraklion.

Abstract: "Fleets of UAVs will be deployed in near future in reliability and safety critical applications (*e.g.* for smart cities). To satisfy the stringent level of criticality, each UAV in the fleet must trust the other UAVs with which it communicates to get assurance of the trustworthiness in information received and to be sure not to disclose information to an unauthorized party. In addition, to be protected against an attacker willing to eavesdrop and/or modify the exchanged data, the communication channel needs to be secured, *i.e.* it has to provide confidentiality and integrity of exchanges. The work presented here is based on our previous research which concluded that it is required that each UAV includes a Secure Element (which we called ARFSSD standing for Active Radio Frequency Smart Secure Device) to withstand an adversary with a high attack potential. In this paper, we

propose a secure and trusted channel protocol that satisfies the stated security and operational requirements for a UAV-to-UAV communication protocol. This protocol supports three main objectives: 1) it provides the assurance that all communicating entities can trust each other and can trust their internal (secure) software and hardware states; 2) it establishes a fair key exchange process between all communicating entities so as to provide a secure channel; 3) it is efficient for both the initial start-up of the network and when resuming a session after a cold and/or warm restart of a UAV. The proposed protocol is formally verified using Casper/FDR and AVISPA."

Note : Dans cet article où je suis cinquième auteure sur les sept, j'ai réalisé la vérification AVISPA du protocole pour des drones aériens, qui sont bien une catégorie de RFID actif comme nous le présenterons au chapitre 1. J'ai partiellement repris ce travail dans le chapitre 3 de ce mémoire ainsi que la vérification Casper/FDR afin d'illustrer l'utilisation d'outils de vérification sur un protocole concret.

Ma plus récente participation au domaine de la vérification des propriétés de sécurité de protocoles pour les RFID actifs se trouve au sein de l'article **"New Efficient M2C and M2M Mutual Authentication Protocols for IoT-based Healthcare Applications"** accepté pour publication en juin 2019 dans la revue *Peer-to-Peer Networking and Applications*.

Abstract: "With the rapid advancement of heterogeneous wireless technologies and their proliferation in ambient connected objects, the Internet of Things (IoT) is a paradigm that revolutionizes communication between people/objects. Communication between connected objects is achieved via various communication modes, including Machine-to-Machine (M2M) and Machine-to-Cloud (M2C). In the medical field, monitoring devices help to

collect, exchange and process patient health parameters, and are employed in open and unprotected environments, which expose them to various attacks. For this reason, providing high levels of security and privacy become crucial, and a first requirement to ensure this is authentication. In this paper, we propose three new lightweight, efficient authentication protocols for IoT-based healthcare applications. We formally verify them using AVISPA and ProVerif automated tools. For each protocol, we provide a security analysis and a performance evaluation that we compare to related existing proposals."

Note : Dans cet article où je suis deuxième auteure, derrière mon amie, doctorante comme moi, Fatma Merabet, et co-écrit avec nos directeurs de thèse, son encadrant et Olivier Blazy, j'ai participé avec elle à la vérification de propriétés de sécurité de trois nouveaux protocoles pour l'IoT pour le domaine de la santé. Là aussi, comme nous le présentons au chapitre 1, ces périphériques sont des RFID actifs. Étant donné que la majorité de l'article est le fruit principal du travail de Fatma Merabet, je n'ai pas souhaité reprendre ce travail dans ce mémoire.

Enfin, d'autres travaux, comme par exemple, un article reprenant les chapitres 1 et 2 est en cours de rédaction sous le titre **"Active RFID Systems : Past, Present and Future"** pour une soumission en revue.

Première partie

État de l'art

Chapitre 1

Présentation des systèmes RFID actifs

Introduction

Au cours des trente dernières années, les progrès constants de la technologie en terme de miniaturisation, d'amélioration des performances des communications sans fil et d'efficacité énergétique ont permis l'évolution rapide des applications informatiques utilisant les RFID actifs. Celles-ci reposent souvent sur des systèmes largement distribués, composés d'un grand nombre de dispositifs matériels intelligents et connectés qui mesurent, collectent, transmettent et échangent en permanence des données de manière sécurisée afin d'offrir aux utilisateurs une vision précise et enrichie pour les aider dans leur prise de décisions. Ces dispositifs RFID actifs peuvent être mobiles ou statiques (fixes), dotés d'une source d'énergie, équipés de capteurs et/ou actionneurs et communiquent à l'aide de technologies sans fil. S'il existe plusieurs types de dispositifs correspondant à cette description, tels que les nœuds capteurs sans fil, les drones, les dispositifs médicaux portables, les beacons intelligents, et si chacun d'eux a ses propres caractéristiques (en termes de taille, de nombre de capteurs et/ou d'actionneurs embarqués, technologies de communication sans fil utilisées, de fonctionnalités, etc.), ils ont pour dénominateur commun de

disposer de ressources limitées.

Aujourd'hui, l'ensemble de ces dispositifs RFID actifs participent activement à IdO (ou IoT en anglais). S'il est bien établi que le concept d'IoT est intrinsèquement lié à la technologie RFID (« Radio Frequency IDentification »), initialement dans sa version passive, actuellement les nœuds RFID actifs sont de plus en plus présents. En effet, le terme IoT a été utilisé pour la première fois par le centre Auto-ID en 1999 [4]. L'idée à cette époque était de travailler sur la connexion des dispositifs RFID à Internet pour créer des réseaux RFID. L'identification par radio-fréquence, ou RFID, est certainement la technologie d'identification automatique la plus répandue de l'histoire moderne. Elle a prouvé son efficacité de part ses avantages :

1. elle est sans fil ;
2. elle est sans contact ;
3. elle fournit une identification unique des objets/personnes ;
4. elle permet une lecture simultanée et sans visibilité directe de plusieurs étiquettes RFID.

Historiquement, un système RFID était composé d'un lecteur, une ou plusieurs étiquettes électroniques aussi appelées « tags » [1] et un système de gestion de données (base de données). L'idée initiale était d'incorporer ou d'attacher des tags contenant des informations sur l'objet/personne à identifier (ces informations étant liées au sujet). Ainsi, en utilisant un lecteur pour interroger les tags il était possible de récupérer les informations qu'ils embarquaient comme illustré figure 1.1.

À l'époque, déjà, les tags pouvaient, selon leur mode d'alimentation, être qualifiés

1. Dans ce mémoire, nous utiliserons indifféremment dans la suite le terme anglais de « tag » (sans guillemet) ou celui d'étiquette pour désigner un dispositif RFID. À l'exception de cette section, le terme tag se référera quasi-exclusivement à un périphérique utilisant la technologie RFID passive. Le terme générique utilisé pour un périphérique utilisant la technologie RFID active sera *nœud RFID actif*.

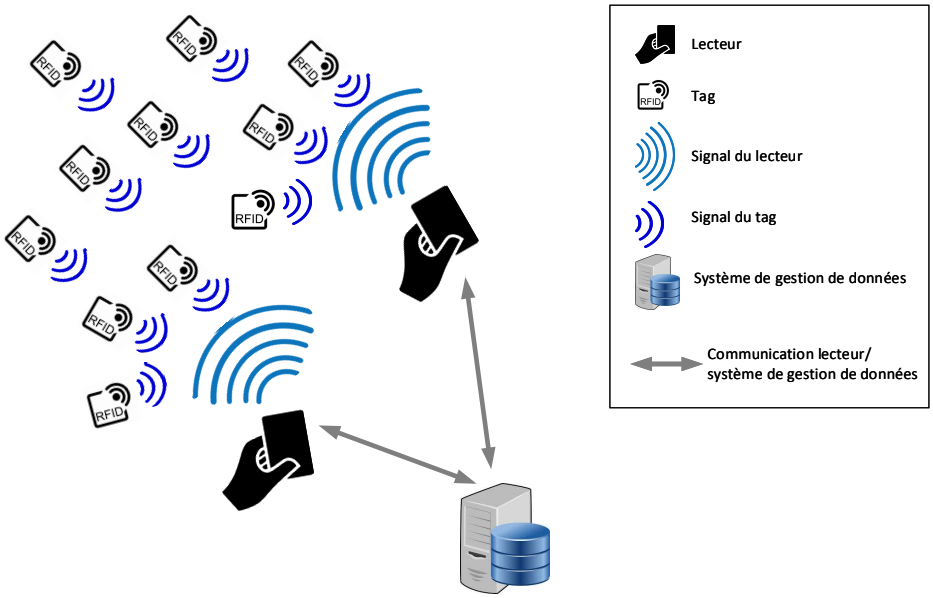


FIGURE 1.1 – Système RFID dans le passé

de passifs ou actifs. Les tags actifs embarquaient une source d'énergie (généralement une batterie) tandis que les tags passifs utilisaient comme source d'énergie une partie du signal radio-fréquence émis par le lecteur. Cette classification prévaut encore aujourd'hui.

Si les premiers tags RFID utilisés pendant la seconde guerre mondiale dans un système de détection d'avions amis/ennemis (IFF : « Identification Friend or Foe ») étaient des dispositifs actifs, ils présentaient des contraintes. En effet, la présence d'une source d'énergie augmentait leur taille et leur poid. Elle limitait aussi la durée de fonctionnement aux périodes pour lesquelles la batterie contenait suffisamment d'énergie. De plus, la présence d'une source de stockage de l'énergie les rendait plus coûteux que les tags passifs. C'est pourquoi historiquement, la technologie RFID a connu son essor essentiellement grâce au développement massif et à la prolifération des tags passifs : ils étaient de taille réduite, avec une durée de vie illimitée et présentaient un coût faible de fabrication.

Au fil du temps, le spectre des applications de la RFID n'a cessé de s'élargir et malgré

les améliorations apportées sur les tags passifs, leurs besoins vont au-delà des capacités de ces derniers. Aujourd'hui, avec les progrès technologiques des systèmes embarqués, la miniaturisation des circuits et les améliorations continues des technologies de batteries et de récupération d'énergie, les simples tags RFID actifs sont devenus des nœuds RFID actifs qui, en plus de l'identification, effectuent d'autres tâches supplémentaires selon les besoins de l'application.

Ainsi, dans ce chapitre, nous passons en revue l'historique des systèmes RFID puis nous décrivons brièvement les composants et les caractéristiques des tags passifs. Ensuite, nous nous focalisons sur les systèmes RFID actifs en répondant aux questions suivantes :

— Qu'est-ce qu'un nœud RFID actif ?

— Comment fonctionne-t-il en terme d'alimentation en énergie, de technologie de communication ?

— etc.

Puis, nous classifions différents types de nœuds RFID actifs (ARs pour « Active RFID ») selon les critères étudiés. Nous présentons aussi leurs nouvelles applications. Enfin, nous présentons leur nouveaux défis mais aussi leur futures évolutions.

1.1 Historique des systèmes RFID

La notion de RFID est apparue la première fois lors de la seconde guerre mondiale [5, 6, 7]. En effet, en 1939, l'armée britannique équipa ses avions d'un système IFF (Identification Friend or Foe) [8], présenté dans la figure 1.2, pour identifier si les avions circulant dans l'espace aérien britannique étaient des avions amis ou ennemis. Les alliés équipaient leurs avions par des transpondeurs (balises) capables de répondre aux interrogations émanant de leurs radars. Cette technologie a été utilisée exclusivement dans

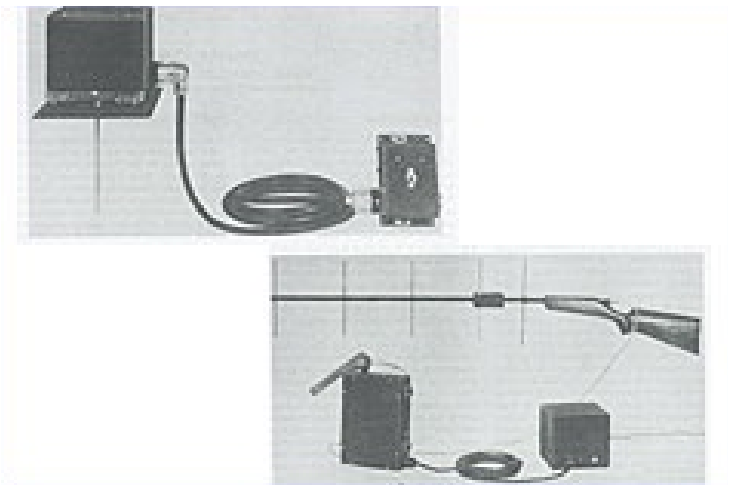


FIGURE 1.2 – Système IFF utilisé dans la 2nde guerre mondiale

le domaine militaire jusqu'à la fin des années 1970. Dans les années 1980, la technologie se répandit dans le secteur privé suite à l'invention des micro-systèmes et à différentes avancées technologiques. Dans ces années, les tags RFID passifs ont en effet commencé à être fabriqués par plusieurs sociétés européennes et américaines. Au cours des années 1990, face à l'accroissement important de la technologie RFID passif, un grand nombre d'initiatives de normalisation ont été initiées. Entre les années 2000-2010, la technologie a continué à se propager. En 2002, Gillette a commandé 500 millions d'étiquettes auprès d'Alien Tech. [9]. Wal-Mart annonçait à la même époque que ses fournisseurs avaient jusqu'à fin 2006 pour intégrer la technologie dans tous les emballages et toutes les palettes [10]. En 2003, une entité à but non lucratif nommée EPCglobal Inc [11] a été créée pour proposer un standard de communication international appelé EPC (« Electronic Product Code ») intégrant les technologies RFID en vue d'améliorer la traçabilité des objets et de faciliter les transactions commerciales.

1.2 Architecture et caractéristiques des systèmes RFID passifs

Si historiquement les tags RFID passifs ont conquis le marché, c'est grâce à plusieurs avantages : taille réduite, durée de vie illimitée, bas coût.

Cette section permet d'illustrer leur architecture et leur principales caractéristiques afin de mieux les comparer avec les systèmes RFID actifs.

1.2.1 Architecture et composants d'un tag RFID passif

Les tags RFID passifs sont constitués d'une puce électronique (unité de contrôle et mémoire) et d'une antenne pour transmettre ou recevoir des données. Ces deux éléments sont assemblés sur un support nommé en français, label ou étiquette (d'où leur nom). Comme illustré figure 1.3, les composants d'un tag sont :

— l'interface RF (antenne) dont le rôle est d'assurer la réception des signaux RF émis par le lecteur et de réagir à ces signaux pour lui fournir une réponse ;

— l'unité de contrôle qui permet d'intégrer des mécanismes d'authentification et de sécurisation de l'accès aux données ;

— une mémoire pour stocker les informations.

L'unité de contrôle et la mémoire sont alimentées par l'énergie récupérée par l'antenne.

1.2.2 Caractéristiques d'un tag RFID passif

Cette section présente quelques caractéristiques importantes des tags RFID passifs.

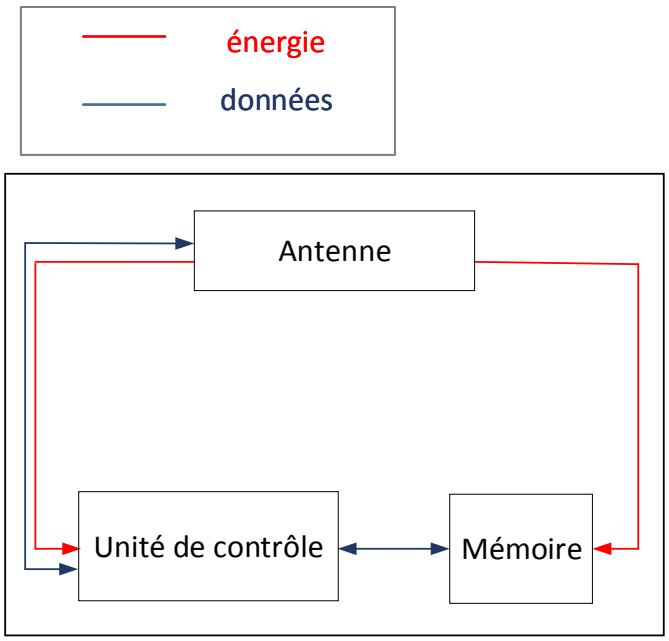


FIGURE 1.3 – Composants d'un tag RFID passif

1.2.2.1 Énergie et méthode de communication

Comme illustré figure 1.3, un tag passif ne possède pas de source d'énergie interne. Par conséquent, le lecteur doit lui fournir l'énergie nécessaire à son fonctionnement. Il existe deux types de couplage pour transférer cette énergie [12]. Le type utilisé dépend essentiellement de la fréquence, de la distance [13] :

— Couplage inductif : Cette méthode fonctionne en champ proche ; deux bobines sont utilisées comme antennes du tag et du lecteur. L'énergie est transmise de la bobine du lecteur à la bobine du tag par induction. Ainsi, une tension sera produite dans la bobine du tag, elle sera redressée et utilisée par l'unité de contrôle et la mémoire. Pour communiquer, le tag modifie la charge dans sa bobine afin de moduler. Cette variation de charge est ensuite détectée par le lecteur comme résultat du couplage mutuel.

— Couplage électromagnétique : Contrairement au couplage inductif, cette méthode fonctionne en champ lointain. Le lecteur émet un signal radio vers le tag. Ensuite, le tag utilise une partie du signal reçu comme source d'alimentation et reflète le reste sous forme de réponse (données) au lecteur.

On notera que nous avons traité dans cette section de l'aspect énergie et méthode de communication car les deux sont fortement intriqués : de la technique de couplage fournissant l'énergie dépend la méthode de communication.

1.2.2.2 Fréquences

Les systèmes RFID passifs utilisent principalement trois types de fréquences dont les caractéristiques diffèrent selon les paramètres de communication (portée, débit de transmission) et aussi l'environnement dans lequel ils fonctionnent (présence de métal, liquide et activités électromagnétiques, etc.) [14] ainsi les tags RFID passifs sont classés selon leurs fréquences :

— Basse fréquence, LF (« Low Frequency ») : les deux fréquences utilisées par les RFID sont 125 kHz et 134 kHz. Ces derniers fonctionnent sur une distance de quelques centimètres. Les tags LF fonctionnent même en présence de métal ou liquide mais sont plus coûteux que ceux utilisant les autres fréquences.

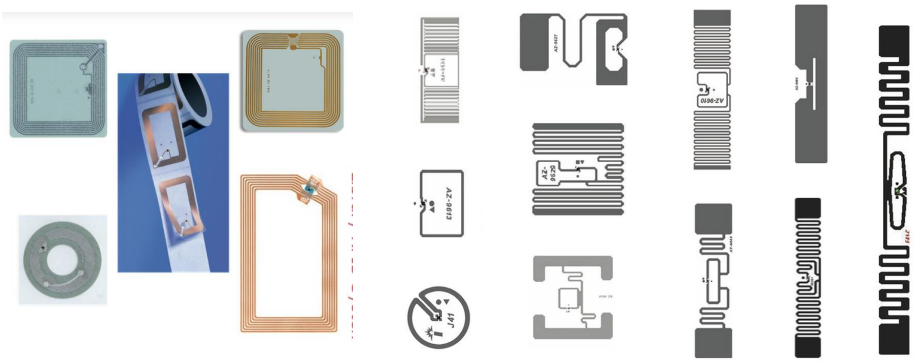
— Haute fréquence, HF (« High Frequency ») : une seule fréquence est utilisée, 13,56 MHz, sur une distance qui ne dépasse pas 1 m. Les tags NFC (« Near Field Communication ») [15], sont un type de tags RFID passifs utilisant cette fréquence. On peut noter que la technologie NFC propose aussi un mode de communication dit P2P (« Peer-to-Peer ») qui sera utilisé pour les systèmes RFID actifs.

— Ultra haute fréquence, UHF (« Ultra High Frequency ») : fréquences dans l'intervalle [860 MHz – 960 MHz], sur une distance allant de 1 m jusqu'à 9 m. Ce type de tags

est largement utilisé pour inventorier des produits.

La figure 1.4 présente des exemples de tags RFID passifs utilisant ces différentes fréquences [16, 17, 18, 19].

Tags LF passifs [10, 11]



Tags HF passifs [12] Tags UHF passifs [13]

FIGURE 1.4 – Exemples de tags passifs

1.2.2.3 Types de communication

Le mode de communication dans un système RFID passif est similaire à celui d'un système client-serveur, le lecteur envoie la requête pour interroger le(s) tag(s). Le tag la reçoit, la traite et *envoie* la réponse au lecteur. Comme illustré figure 1.5, la communication entre le lecteur et le tag est du type maître-esclave, un tag ne répond que lorsqu'il est interrogé.

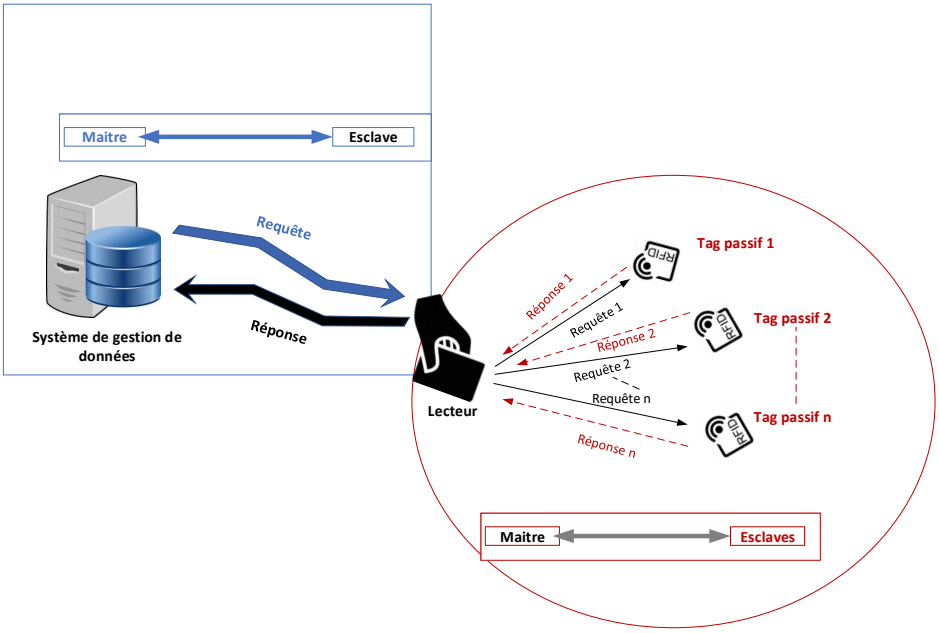


FIGURE 1.5 – Types de communication d'un système RFID passif

1.2.3 Applications

La vente au détail et l'inventaire [5] sont considérés comme les deux premiers domaines d'utilisation des systèmes RFID passifs. Ensuite, ils se sont répandus dans d'autres domaines tels que la santé, les transports, l'identification des animaux, les bibliothèques, le militaire et le bancaire. Le tableau 1.1 présente quelques exemples d 'applications des systèmes RFID passifs.

1.2.4 Défis, évolutions et limites

Cette section présente les défis auxquels les RFIDs passifs ont eu à faire face mais aussi leur évolutions récentes et leurs limites.

Tableau 1.1 – Exemples de quelques domaines d'application des systèmes RFID passifs

Domaine	**Exemples**	**Références**
Vente au detail	Wal-Mart, Tesco	[20]
Fabrication	Toyota, Harley Davidson, HP, Viper Motorcycles	[21, 22, 23]
Médical	Chang-Gung Hospital, Hamilton Health	[24, 25]
Transport et étiquetage	Parcelforce, Zonar systems, Spectra Inc	[26]
Militaire	US Departement of Defense	[27]
Bibliothèques	Farmington country library, Dutch research institute TNO	[26]
Agriculture	Identification et contrôle des animaux d'élevage	[28, 14]
Bancaire	Paiement sans contact EMV	[29]

1.2.4.1 Défis

Les premiers tags RFID passifs, comme ceux des systèmes EAS (« Electronic Article Surveillance ») [30], étaient fabriqués sans aucun mécanisme de sécurité intégré et ils assuraient donc seulement la tâche d'identification des produits. Les tags pouvant être interrogés par n'importe quel lecteur, étaient vulnérables à de multiples attaques comme l'écoute passive, le clonage, etc. Dans certains domaines critiques, par exemple le domaine militaire, la divulgation d'informations pouvait avoir des conséquences graves. Aussi, pour garantir un minimum de sécurité aux tags, ceux-ci devaient pouvoir s'assurer de communiquer avec un lecteur légitime et vice-versa. Les RFID ont alors glissé de l'identification à l'authentification.

Plusieurs solutions ont alors été proposées dans la littérature. Certains travaux se sont intéressés à la sécurisation physique des tags [31, 32]. D'autres ont choisi d'utiliser des protocoles cryptographiques. Comme l'utilisation des mécanismes traditionnels de sécurité était impossible en raison des ressources limitées des tags [33], plusieurs protocoles

d'authentification ultra-légers ont été proposés dans la littérature [34, 35, 36, 37, 38, 39, 40, 41].

Il sera intéressant de voir au chapitre 2 comment les nœuds RFID actifs font et vont faire face à ces défis.

1.2.4.2 Évolutions

Les systèmes RFIDs passifs ont subi depuis quelques années deux grandes évolutions. La première concerne leurs fonctionnalités et la seconde les protocoles d'interrogation des tags dans le système global.

Fonctionnalités Avec les progrès récents de la technologie et l'évolution des systèmes embarqués, comme illustré figure 1.6, il est devenu possible de réaliser des extensions sur les tags passifs afin de les utiliser dans des applications plus complexes qui nécessitent d'accomplir des tâches supplémentaires (au-delà de la simple identification) : comme la surveillance de l'environnement ou de l'objet/personne via des capteurs.

Si l'idée d'utiliser des tags capteurs passifs n'était pas nouvelle, ce n'est que les récents progrès technologiques qui ont permis ces développements. Ainsi, la figure 1.7 présente le tag plateforme de prototypage WISP (« Wireless Identification and Sensing Platform ») [1] qui contient un capteur de température et un accéléromètre. La figure 1.8 présente quant à elle une étiquette du commerce de PHASE IV Engineering [2] pour détecter la température, la déformation et la pression.

Un autre exemple d'étiquette intelligente intégrant un capteur de température est celle commercialisée par ThinFilm présentée figure 1.9 [3] et qui utilise le NFC.

Protocoles sans serveur Initialement, comme illustré figure 1.1, les dispositifs d'un système RFID passif (lecteur et tags) fonctionnaient seulement en mode connecté, ce qui

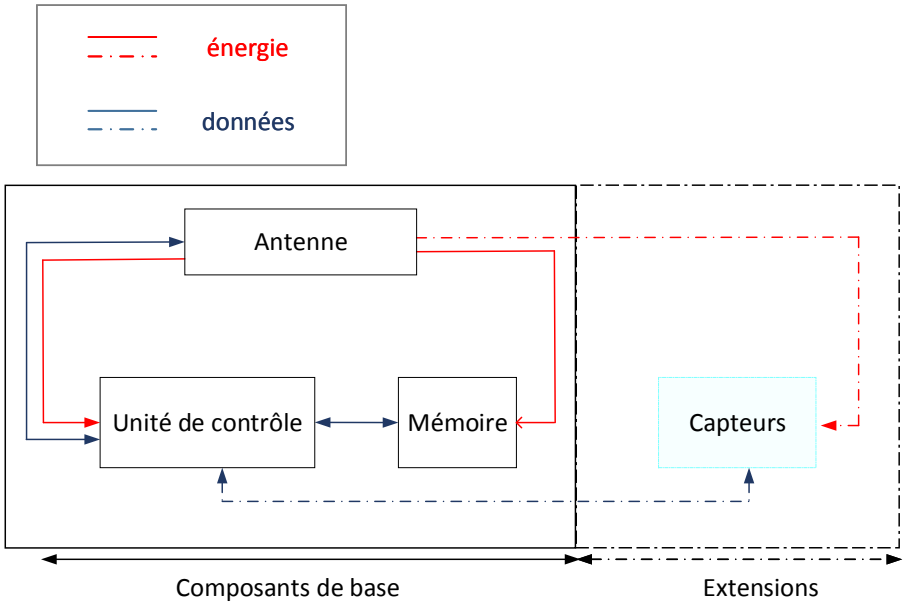


FIGURE 1.6 – Composants d'un tag RFID passif équipé de capteurs

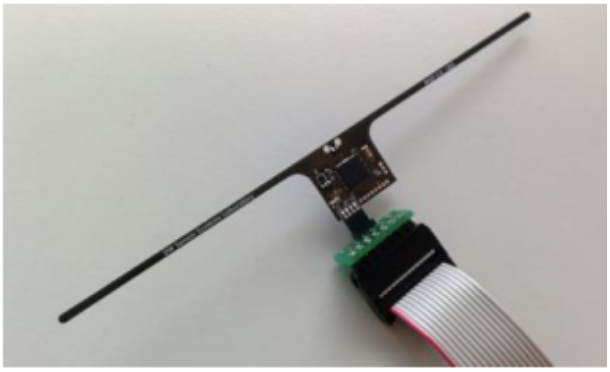


FIGURE 1.7 – Plateforme de prototypage RFID UHF : WISP [1]

nécessitait une communication constante avec le serveur pour s'authentifier et échanger des données en toute sécurité. Cependant, comme pour certaines applications les tags passifs ont été déployés dans des zones isolées où la communication avec le serveur, via le lecteur, n'est pas toujours possible, l'authentification et les échanges sécurisés de données doivent pouvoir se faire directement entre le lecteur et les tags sans l'intervention du

FIGURE 1.8 – Exemple de tag UHF de PHASE IV Engineering [2]

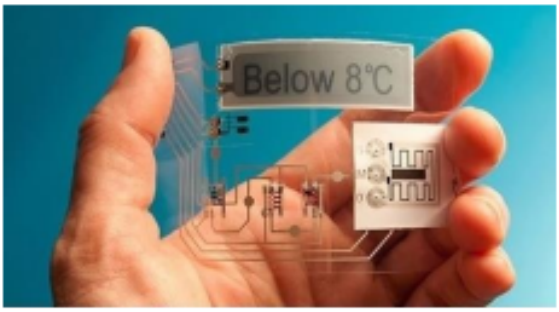


FIGURE 1.9 – Étiquette NFC avec capteur de température de ThinFilm [3]

serveur. Ce mode est appelé *mode sans serveur* (« serverless mode ») [42]. Cela consiste à donner des accréditations (c'est-à-dire le matériel cryptographique nécessaire) au lecteur afin qu'il puisse dialoguer et en particulier s'authentifier de façon autonome avec les tags. Contrairement aux protocoles d'authentification nécessitant un serveur, dans le contexte des RFID passifs, les protocoles serverless sont particulièrement efficaces concernant le temps de réponse puisqu'ils réduisent le nombre d'intermédiaires. Ils permettent aussi d'assurer une meilleure disponibilité et donc une fiabilité globale du système. Le système est plus résilient face à une perte de connexion entre le lecteur et le serveur ou une panne

de ce dernier puisqu'il n'est pas nécessaire pour le fonctionnement.

Ce type de fonctionnement n'est finalement qu'une décentralisation du système, qui comme nous le verrons en sections 1.3.2.4 et 1.4.3, est particulièrement adapté aux nœuds RFID actifs.

1.2.4.3 Limites

Certaines applications nécessitent aujourd'hui le déploiement des systèmes RFID dans des environnements qui représentent un défi pour les transmissions RF (par exemple, les mines profondes, les conteneurs en métal, l'intérieur des bâtiments). Dans de telles situations, les systèmes RFID passifs ne peuvent pas accomplir leurs tâches correctement car :

— Les tags RFID passifs embarquant des capteurs n'étant alimentés que si ils se trouvent à proximité d'un lecteur, les capteurs ne sont pas capable de faire des mesures en permanence.

— Les systèmes RFID passifs n'opèrent que sur de courtes distances (inférieures à 10 m).

— Certaines applications nécessitent, en plus de l'identification, d'accomplir d'autres tâches ou d'échanger de l'information sur de grands réseaux sans fil. Cela est évidemment impossible avec les tags passifs qui ne peuvent pas communiquer entre eux.

— Dans ces environnements pouvant être fortement hostiles, les mécanismes de sécurité bas coût utilisés dans les tags passifs ne sont pas suffisants pour protéger les tags eux-même, leurs données et leur communications avec les lecteurs.

Afin de passer outre ces limites, des dispositifs RFID actifs autonomes (possédant leurs propres sources d'énergie) et pouvant accomplir à la fois le rôle d'un interrogateur (lecteur)

et celui d'un transpondeur (tag) ont été développés. Actuellement, avec les progrès des technologies des batteries et des techniques de récupération d'énergie, il est possible de développer ce genre de dispositifs qui constituent un système RFID actif et rassemblent sur le même support une puce (micro-contrôleur), une antenne RF et une source d'énergie avec possibilité d'intégrer d'autres éléments tels que des capteurs, des actionneurs, des modules de sécurité, etc. selon le besoin de l'application. Cette combinaison constitue ce que l'on nomme un *nœud RFID actif*.

1.3 Qu'est-ce qu'un nœud RFID actif ?

Bien qu'initialement les systèmes RFID passifs et actifs partageaient le même objectif d'identification automatique des objets/personnes. Aujourd'hui ces deux derniers sont complètement différents que ce soit dans l'architecture ou dans les applications qui les utilisent. Les systèmes RFID actifs sont caractérisés par l'utilisation des nœuds actifs (ARs pour « Active RFID »).

Cette section est le pendant pour les nœuds RFID actifs de la section 1.2 pour les systèmes RFID passifs. Elle présente donc leur architecture et leur principales caractéristiques.

1.3.1 Architecture et composants d'un nœud RFID actif

Comme illustré figure 1.10, un nœud actif est composé des deux éléments de base : la puce (unité de contrôle et la mémoire) et l'antenne. Cet ensemble est alimenté par une source d'énergie embarquée sur le nœud, ce qui lui permet d'agir d'une manière autonome. Des éléments supplémentaires peuvent être intégrés au nœud actif tels que des mémoires, des capteurs/actionneurs, des modules de sécurité ou même des interfaces de

communication secondaires.

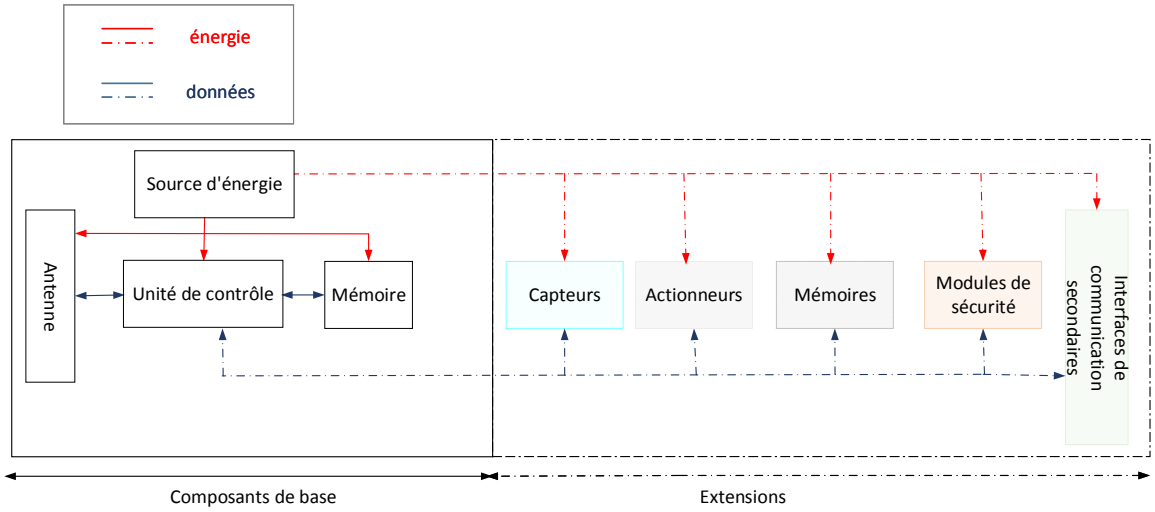


FIGURE 1.10 – Composants d'un nœud RFID actif

À noter

L'autonomie énergétique du nœud RFID actif est au cœur du changement de paradigme. Les nœud RFID actifs peuvent fonctionner de façon autonome (au contraire des tags passifs), élargissant ainsi le panel des domaines d'applications.

1.3.2 Caractéristiques d'un nœud RFID actif

Cette section présente quelques caractéristiques importantes des nœuds RFID actifs.

1.3.2.1 Énergie

La source d'énergie nécessaire au fonctionnement du nœud RFID actif diffère selon son type et sa taille. L'énergie électrique peut provenir d'une unité de stockage, c'est-à-dire une pile ou une batterie, elle peut être produite par l'utilisation de techniques de récupération

d'énergie (« energy harvesting techniques ») à partir des différentes sources externes telles que : l'énergie lumineuse (soleil, lumière ambiante, etc.), l'énergie mécanique (vibrations, mouvements, etc.), l'énergie thermique (température, etc.) [43, 44]. Si ces techniques de récupération d'énergie sont intégrés au nœud, il dispose, comme illustré figure 1.11, d'une unité de récupération d'énergie qui comprend un module de conversion d'énergie, un module de stockage d'énergie et un circuit régulateur de puissance [45, 46].

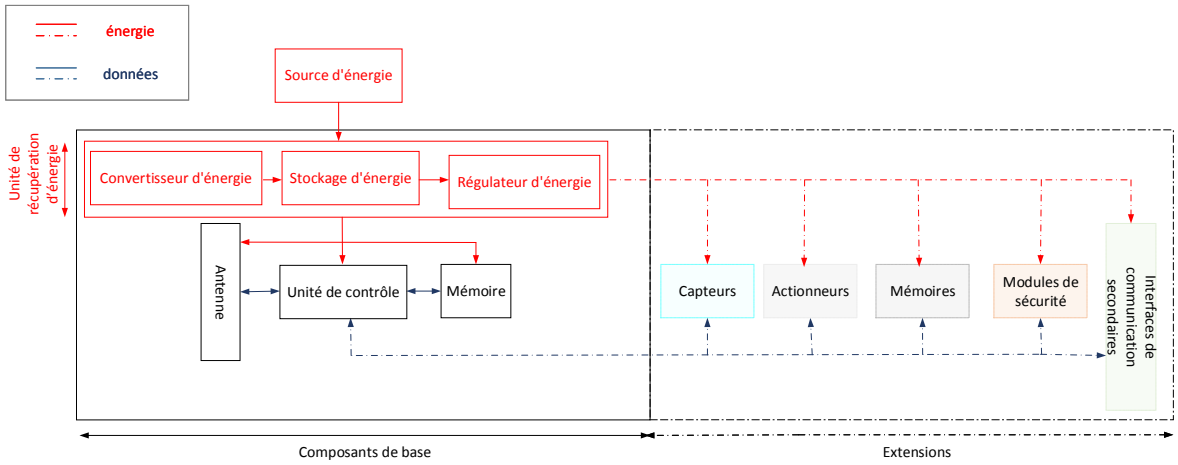


FIGURE 1.11 – Composants d'un nœud RFID actif embarquant une unité de récupération d'énergie

On notera que contrairement à la section 1.2.2.1, pour les nœud RFID actifs, les méthodes de communication sont totalement indépendantes de la source d'énergie. Au moment où je finalise la rédaction de ce mémoire de thèse, je trouve assez fascinante l'annonce récente d'Atmosic [47] Technologies selon laquelle le futur des nœud RFID actifs (périphériques qui utiliseront Bluetooth 5.0) se trouvent peut-être dans les techniques de récupération d'énergie RF (provenant soit des équipements présents dans l'environnement, soit de techniques contrôlées de communication RF ou d'orientation de faisceaux) [48]. C'est finalement là une belle ironie de l'histoire : les techniques ancestrales d'*alimentation* des RFID passifs au service des RFID actifs.

1.3.2.2 Fréquences

Les systèmes RFID actifs traditionnels opèrent sur des fréquences entre 433 MHz (UHF) et 2,4 GHz (SHF, « Super High Frequency ») ce qui leur permet de communiquer selon les besoins applicatifs sur une distance pouvant être supérieure à 100 m. En effet, les récentes avancées dans le domaine des communications ont conduit à l'intégration sur les nœuds RFID actifs d'un ensemble de nouvelles technologies de communication dans les bandes ISM (« Industrial, Scientific and Medical ») telles que, dans la bande des 2,4 GHz, le Bluetooth, ZigBee et Wi-Fi [49]. Ainsi, aujourd'hui, les nouveaux nœuds RFID actifs peuvent communiquer entre eux et même à longue distance ou dans des réseaux.

1.3.2.3 Types de communication

Selon les technologies de communication présentes dans le système, essentiellement, deux grands modes de communication sont possibles.

Dans certains systèmes RFID actifs (traditionnels) la présence des lecteurs est possible. Ces systèmes suivent le même mode de communication client/serveur (ou maître/esclave) que les systèmes RFID passifs avec toutefois deux possibilités de communication :

— Les lecteurs initient la communication en interrogeant les nœuds RFID actifs, comme dans le cas des Trax-Boxes interrogés par les M-Trax [42] ou dans celui de transpondeurs actifs interrogés par des lecteurs ; c'est une configuration similaire aux systèmes RFID passifs.

— Les nœuds RFID actifs initient la communication comme émetteurs et les lecteurs accomplissent le rôle des récepteurs comme dans le cas des clés de voitures pour l'ouverture et le verrouillage à distance.

Dans d'autres systèmes actifs, le mode de communication adopté est le P2P (« Peer-to-Peer »). Les nœuds actifs sont à la fois des émetteurs et des récepteurs comme dans le

cas des nœuds d'un réseau de capteurs sans fil.

Les communications peuvent aussi être de type broadcast ou unicast selon les besoins de l'application :

— Unicast : comme dans certains protocoles de routage dans les réseaux de capteurs sans fil (la destination est la station de base, le « sink »).

— Broadcast : comme dans le cas des balises « beacons » qui diffusent l'information sans spécifier le destinataire.

1.3.2.4 Modes de fonctionnement applicatif

Au delà des modes de communication des nœuds, c'est-à-dire client/serveur et P2P présentés, en fonction des besoins de l'application, les nœuds RFID actifs peuvent soit fonctionner en mode connecté, soit en mode sans serveur (ou « serverless »), même si ce dernier est souvent privilégié.

Mode connecté Dans ce mode, les nœuds RFID actifs maintiennent une communication permanente entre eux et des serveurs centralisés distants en particulier pour effectuer la phase d'authentification. Si ce mode est viable pour certaines applications nécessitant d'être « always online », il est consommateur d'énergie et de bande passante puisqu'une connexion permanente est requise. De plus, comme la phase d'authentification entre les nœuds se réalise via un serveur distant, ce mode n'est utilisable que lorsqu'une certaine latence peut être tolérée ou lorsque le réseau à une faible dynamique (c'est-à-dire peu de connexions/déconnexions liées à l'environnement ou à la mobilité par exemple). Son principal défaut est que si la connexion au serveur n'est plus possible alors les nœuds se retrouvent isolés et ne peuvent plus communiquer entre eux.

En raison, des caractéristiques exigeantes du mode connecté, pour le fonctionnement

des systèmes RFID actifs, c'est souvent le mode sans serveur qui est privilégié.

Mode sans serveur Ce terme de « sans serveur », déjà introduit en section 1.2.4.2, fait référence aux mécanismes permettant aux entités (ici, les nœuds RFID actifs) contrôlées de manière centralisée de s'authentifier mutuellement de manière autonome, c'est-à-dire sans l'intervention active des serveurs. Les protocoles serverless visent à sécuriser la communication de proximité entre les nœuds actifs tout en optimisant le peu de ressources disponibles [42] (c'est-à-dire en ne les gaspillant pas pour établir une communication avec le serveur).

Les avantages du mode sans serveur sont :

- De minimiser la consommation d'énergie et donc d'augmenter l'autonomie, c'est-à-dire la durée de vie (fonctionnement), des nœuds RFID actifs :
  - en évitant de maintenir une connexion permanente au travers du réseau entre les nœuds et un serveur (cette économie d'énergie est particulièrement visible sur les réseaux longues distances qui nécessitent soit des nœuds relais soit beaucoup de puissance pour établir une connexion point à point) ;
  - en minimisant les nombres d'échanges entre les nœuds dans le protocole.
- D'offrir aussi une plus grande résilience. La fiabilité et la disponibilité sont meilleures puisque l'initiation des communications applicatives est locale et qu'elle ne repose pas sur des nœuds relais ou sur un serveur qui peuvent être l'objet de défaillance ou d'attaques.
- De consommer moins de bande passante (puisque les communications sont essentiellement locales) et d'obtenir de meilleurs temps de réponse (il y a moins d'intermédiaires et de distance) que le mode connecté.

— De ne pas nécessiter de connexion permanente avec un serveur en privilégiant les communications locales (ce qui est conforme aux concepts imaginés par exemple dans la future technologie 5G ou encore dans le paradigme émergent du « edge computing »).

De part leurs nombreux avantages, les protocoles sans serveur sont donc particulièrement adaptés aux nœuds RFID actifs qui peuvent être connectés seulement par intermittence (comme ceux déployés dans des zones géographiques isolées où ils peuvent rester hors ligne lors de longues périodes). Comme ces protocoles réduisent la consommation de ressources tout en optimisant la sécurité et la résilience des communications des nœuds RFID actifs situés à proximité géographique et qu'en plus ils sont adaptables à de nombreux contextes applicatifs utilisant les nœuds RFID actifs (voir par exemple la section 1.4.3), nous pensons que les protocoles sans serveur sont clairement les candidats idéaux pour s'imposer dans le futur de la technologie RFID active et nous avons fait le choix de proposer en partie II, une solution de sécurité utilisant ce mode de fonctionnement.

1.3.3 Taxonomie de nœuds RFID actifs

Avec le développement de l'IdO, de nombreux nœuds RFID actifs sont apparus. Dans cette section, nous présentons les grandes catégories les plus connues et les plus utilisées actuellement en décrivant quelques unes de leur caractéristiques importantes.

1.3.3.1 Transpondeurs RFID actifs

Dans cette catégorie, on trouve les tags RFID actifs traditionnels dont les composants sont présentés figure 1.12, qui sont principalement serveurs et attendent d'être interrogés par un lecteur pour répondre, comme celui présenté figure 1.13, mais aussi ceux qui initient

la communication comme ceux présents dans les clés de voiture comme illustré figure 1.14.

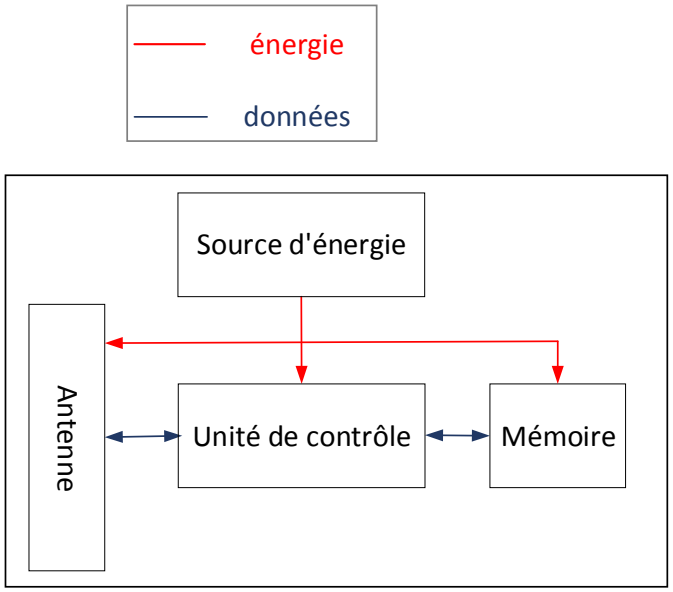


FIGURE 1.12 – Composants d'un exemple de transpondeur RFID actif

FIGURE 1.13 – Transpondeur RFID actif classique dans son boîtier

Ces systèmes RFID sont composés de lecteurs, de transpondeurs RFID actifs et parfois de serveurs hébergeant des bases de données comme dans le cas d'une application

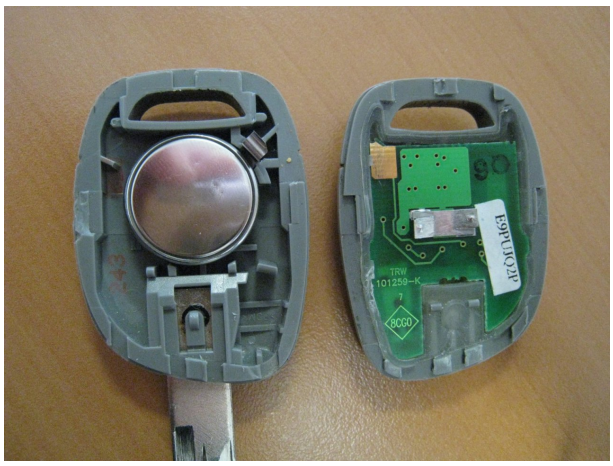


FIGURE 1.14 – Transpondeur RFID actif d'une clé de voiture

d'inventaire d'entrepôts.

Les transpondeurs RFID actifs les plus basiques ont un seul rôle qui est l'identification (c'est-à-dire qu'ils transmettent leur identifiant unique aux lecteurs).

Toutefois, il existe aussi des transpondeurs RFID actifs plus complexes qui peuvent intégrer des capteurs, comme ceux utilisés dans la surveillance de conteneurs [42]. Dans ce cas, les transpondeurs peuvent transmettre des informations collectées par les capteurs intégrés, en plus de leur identifiant. Ce type de transpondeur a généralement besoin d'une capacité de stockage supérieure à celle des transpondeurs basiques [50].

Équipés de capteurs, les transpondeurs RFID actifs se rapprochent fortement des nœuds RFID capteurs présentés ci-après. Il est toutefois important de noter que les transpondeurs RFID actifs de cette catégorie ne peuvent pas communiquer entre eux.

1.3.3.2 Nœuds RFID capteurs

Les réseaux de capteurs sans fil (WSNs pour « Wireless Sensor Networks ») consistent en un grand nombre de nœuds RFID capteurs qui coopèrent (c'est-à-dire qui communiquent entre eux pour router les messages) afin de fournir à la destination finale (c'est-à-

dire la station de base) [51] les informations collectées sur la zone de déploiement ou sur l'environnement des objets/personnes sur lesquels ils sont attachés.

Comme illustré figure 1.15, un nœud d'un WSN étant composé bien évidemment d'un module capteur, d'un microprocesseur, d'une mémoire, d'une interface de communication RF (par exemple ZigBee comme sur le nœud TelosB présenté figure 1.16) et d'une batterie permettant d'alimenter tous ces composants, du point de vue matériel, son architecture est identique à celle d'un nœud RFID actif présenté figure 1.10.

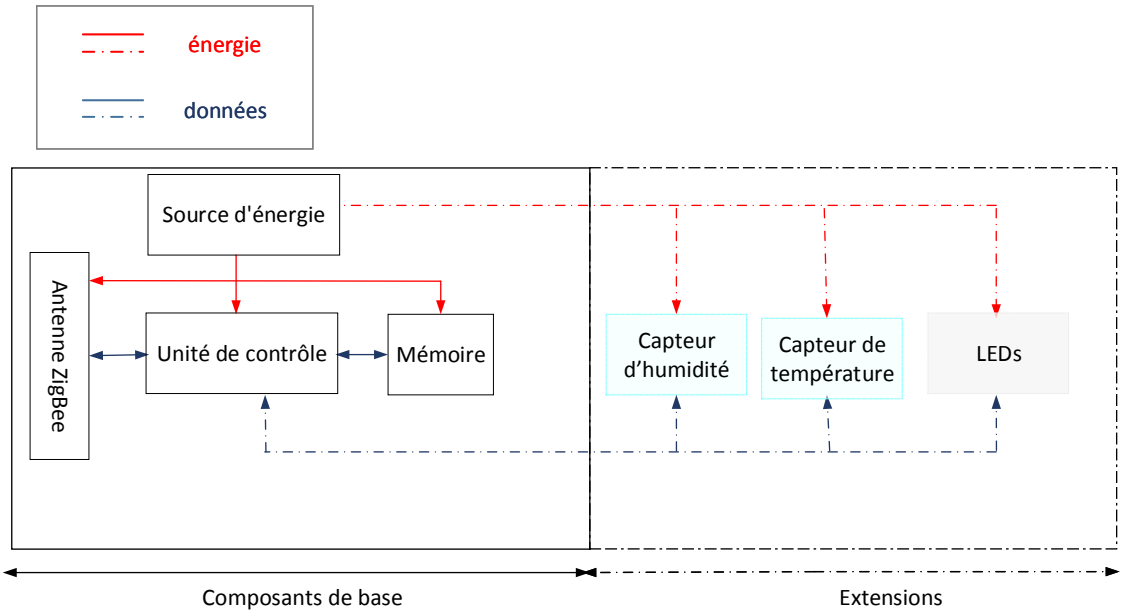


FIGURE 1.15 – Composants d'un exemple de nœud RFID capteur

La seule petite différence entre un nœud RFID capteur d'aujourd'hui et les premiers nœuds RFID actifs concerne la notion d'identifiant unique [2]. En effet, les premiers nœuds RFID actifs ayant eu pour seul but l'identification, on devait pouvoir les différentier de façon unique alors que les nœuds des WSN peuvent posséder le même identifiant.

2. Les règles et normes de nommage (format, longueur, etc.) peuvent varier mais le principe est le même.

FIGURE 1.16 – Nœud RFID TelosB

Toutefois en pratique, il est nécessaire que chacun d'eux possèdent un identifiant unique afin de permettre les communications unicast [52] et assurer le routage.

Ainsi, dans cette catégorie, les nœuds des WSN sont des nœuds RFID actifs et les WSNs sont des réseaux de nœuds RFID actifs.

1.3.3.3 Balises et balises intelligentes

Dans cette catégorie, se trouvent les balises classiques et celles dites intelligentes.

Les balises classiques ou « beacons » sont des nœuds RFID actifs équipés d'une technologie de communication (en général Bluetooth Low Energy) et dont l'objectif est de diffuser un identifiant de manière périodique sous forme d'un signal RF. Ce signal peut être reçu et traité par tout appareil prenant en charge la technologie de communication utilisée (encore une fois, c'est souvent Bluetooth). Ainsi, les « smartphones » (téléphones intelligents) peuvent utiliser les informations émises par les beacons pour réaliser de la géolocalisation en intérieur (« indoor »); ce qui est par exemple utile partout où les signaux des satellites de type GPS (Global Positioning System), ne passent pas.

Les balises intelligentes ou « smart beacons » diffusent pour leur part, en plus de

leur identifiant, les données mesurées par les capteurs qu'ils intègrent comme illustré figure 1.17.

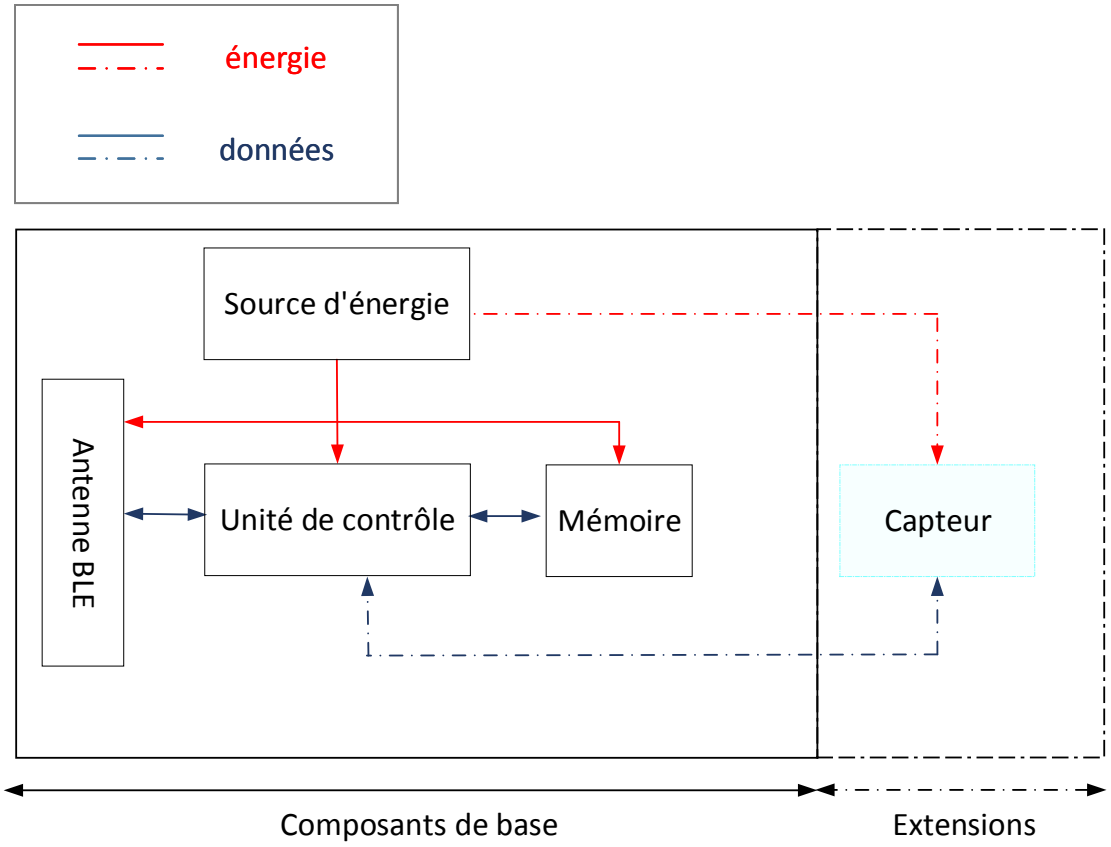


FIGURE 1.17 – Composants d'un exemple de smart beacon

La figure 1.18 illustre deux types de smart beacons : la plateforme de prototypage à base de Bluetooth nRF51822 [53] de Nordic Semiconductor dont on peut choisir les capteurs et l'ASensor, un produit commercialisé par la société April Beacon [54], qui contient comme capteur un accéléromètre 3 axes sur 12 bits.

Il est important de noter que les beacons ou smart beacons ne peuvent pas être assimilés à des transpondeurs RFID actifs puisqu'ils sont initiateurs de la communication, que celle-ci à lieu en permanence et qu'elle est à destination de tous les récepteurs et non d'un

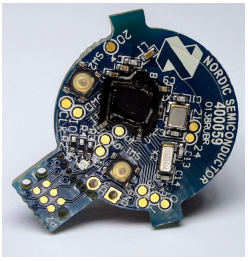



FIGURE 1.18 – Smart beacons

seul comme pour le transpondeur RFID actif de la clé de voiture (dont la destination est le récepteur dans la voiture). Il faut aussi noter qu'au contraire des nœuds RFID capteurs, et tout comme les transpondeurs RFID actifs, les beacons ou smart beacons ne peuvent pas communiquer entre eux.

1.3.3.4 Drones

Cette catégorie se différencie de celle des nœuds RFID capteurs par l'aspect mobilité. En effet, un drone (terrestre, volant, marin ou sous-marin) n'est finalement qu'un capteur (ou une plateforme de plusieurs capteurs) mobile. Quand des drones opèrent en flotte homogène ou hétérogène, ils constituent finalement un WSN dont les nœuds sont mobiles. Il est à noter qu'ils peuvent aussi être combinés avec un ou des WSNs pour les étendre ou pour réaliser des missions de collecte de données depuis des réseaux ou des nœuds isolés afin de les rapatrier vers une station de base. Il peut aussi être intéressant de les utiliser dans des environnements hostiles, car ils peuvent être moins facilement détectables par un adversaire en raison de leur mouvement.

Il est à noter, comme illustré figure 1.19, que les drones (comme par exemple le drone volant présenté figure 1.20) comportent des actionneurs (c'est-à-dire des moteurs) pour leurs permettre de se déplacer. Si par leur présences, ces derniers apportent la mobilité, ils consomment une grande partie de l'énergie de ces nœuds RFID actifs ; c'est la principale raison qui nous a poussé à ne pas les intégrer en sous-catégorie des nœuds RFID capteurs

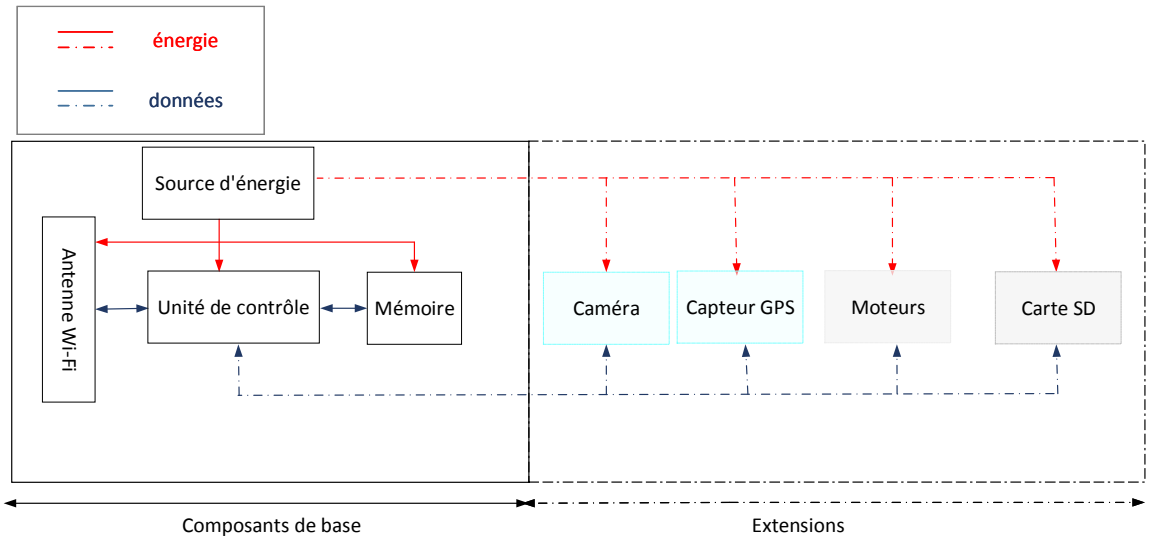


FIGURE 1.19 – Composants d'un exemple de drone

et à en faire une catégorie spécifique.

FIGURE 1.20 – Drone volant

Les drones constituent donc une catégorie particulière de nœuds RFID actifs en apportant la mobilité.

1.3.3.5 Téléphones intelligents

Si bien évidemment les téléphones intelligents (« smartphones ») servent toujours à passer des appels comme autrefois, ils constituent aujourd'hui, comme illustré figure 1.21, des nœuds RFID actifs bien spécifiques, car, de part toutes les technologies de communications qu'ils supportent (2G, 3G, 4G, LTE, 5G, Wi-Fi, Bluetooth, NFC, etc.) et de part tous les capteurs qu'ils embarquent (capteur GPS, accéléromètre, magnétomètre, gyroscope, etc.), ils sont des passerelles multi-technologies qui peuvent assurer l'interopérabilité avec les différentes catégories de nœuds RFID actifs présentés.

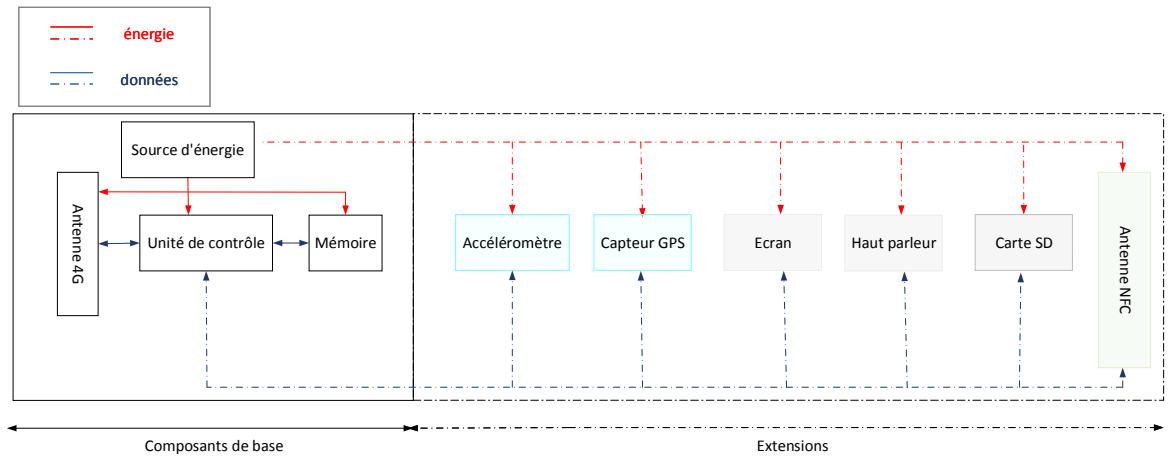


FIGURE 1.21 – Composant d'un exemple de smartphone

Par exemple, si la technologie NFC peut leur permettre d'effectuer des paiements (ici en mode RFID passif – le smartphone émulant un tag RFID passif et le terminal étant le lecteur RFID passif – ce qui permet donc de faire des paiements même lorsque le téléphone n'a plus d'énergie dans sa batterie), comme illustré figure 1.22 [55], elle peut aussi permettre d'échanger avec les nœuds RFID actifs que nous présentons en section 1.3.3.6. La technologie Bluetooth peut aussi permettre au téléphone de recevoir des informations des balises de la section 1.3.3.3.

FIGURE 1.22 – Smartphone équipé de la technologie NFC

Il est donc tout à fait envisageable d'interconnecter des smartphones au travers de différentes technologies, en utilisant, ou non, une infrastructure (c'est-à-dire potentiellement même en P2P), afin de constituer un réseau de capteurs.

1.3.3.6 Plateforme NFC-WISP

La dernière catégorie que nous proposons est très particulière car elle ne contient à notre connaissance que la plateforme NFC-WISP (« Near Field Communication-Wireless Identification and Sensing Platform ») qui est en quelque sorte un pont entre la technologie des tags RFID passifs (NFC en mode lecteur-tag) et la technologie des nœuds RFID actifs.

En effet, le périphérique [56] présenté figure 1.23 est une plate-forme programmable composée d'un ensemble de capteurs et d'actionneurs (écran d'affichage), d'un micro-contrôleur programmable et d'une mémoire. Ce nœud pouvant être lu par des périphériques capables de se comporter comme des lecteurs NFC (c'est-à-dire des lecteurs classiques pour carte à puce sans contact ou des smartphones équipés) et lui agissant comme un tag passif, on aurait pu le considérer comme n'étant pas un nœud RFID actif. Toutefois, comme le NFC-WISP peut réagir de manière autonome grâce à sa batterie, c'est bien un nœud RFID actif.

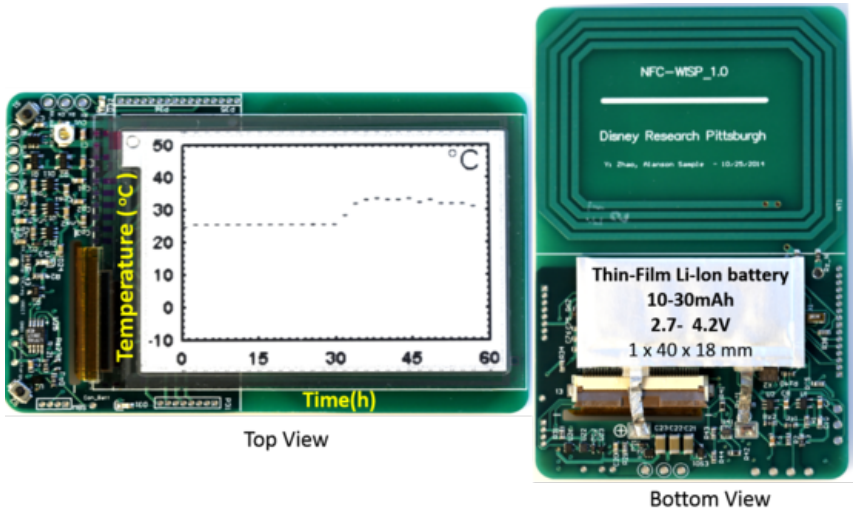


FIGURE 1.23 – NFC-WISP

Zhao et al. [57] ont discuté deux exemples d'utilisation de cette plateforme illustrant bien la dualité du NFC-WISP :

1. comme dispositif passif utilisé pour afficher une image envoyée par un lecteur NFC (tag RFID passif disposant d'un écran) ;
2. comme un dispositif autonome pour surveiller et afficher les variations de température des aliments (nœud RFID actif utilisant ses capteurs et réalisant des actions).

On pourra noter que pour recharger sa batterie, le NFC-WISP utilise des techniques de récupération d'énergie, ici via le couplage inductif NFC. Si aujourd'hui deux NFC-WISP ne sont pas capables de communiquer entre eux, il semble qu'il suffirait de programmer le mode P2P de NFC dans le projet pour qu'ils puissent le faire.

1.3.3.7 Exemples de nœuds RFID actifs existants

Dans le tableau 1.2, nous donnons quelques caractéristiques de nœuds RFID actifs existants.

Tableau 1.2 – Exemples de nœuds RFID actifs et leurs caractéristiques

Nœud RFID actif	**Source d'énergie**	**Communication**			
		Fréquence ou Technologie	**Distance** m	**Débit** kbps	**Mode**
Clé de voiture [58]	Pile	UHF	10-100	4-19	I-U
iBeacon [59]	Pile	Bluetooth 4.2	50-150	1000	I-B
Eddyston [60]	Pile	Bluetooth 4.2	50-150	1000	I-B
TelosB [61]	Pile	ZigBee 3.0	10-100	250	P2P-U,B
T5 sensor [62]	Pile	Wi-Fi	>50	200000	C/S-U
Téléphone	Batterie	2G, 3G, 4G, etc.	10000	10-300000	C/S-U

I : Initiateur – C/S : Client/Serveur – P2P : Peer-to-Peer
U : Unicast – B : Broadcast

1.4 Applications récentes

Comme nous l'avons évoqué à plusieurs reprises depuis le début de ce chapitre, les applications utilisant les nœuds RFID actifs vont au-delà de la simple identification. En effet, plusieurs applications nécessitent l'utilisation de systèmes RFID à grande échelle pour couvrir des zones très vastes. Compte tenue de leur caractéristiques, seuls les nœuds RFID actifs sont adaptés. Nous détaillons quelques applications représentatives de leur utilisation ci-après.

1.4.1 Surveillance de l'environnement des objets/personnes

Les nœuds RFID actifs sont utilisés dans de nombreuses applications nécessitant la mesure d'informations relatives à l'environnement de l'objet/personne [63] (telle que la surveillance de la température d'un objet/personne) grâce à la présence d'un ou de plusieurs capteurs sur la plateforme du nœud actif [64, 65]. Par exemple, un nœud RFID actif possédant un capteur de lumière placé dans un conteneur peut permettre de détecter et enregistrer si la porte du conteneur est ouverte pendant le transport [46]. Dans le domaine

médical, les nœuds RFID actifs peuvent surveiller les constantes de santé d'un patient, puis les transmettre directement au serveur central si le patient se trouve dans la salle de soins ou via Internet s'il est chez lui [46]. Comme mentionné dans [66], il est par exemple possible d'utiliser les nœuds RFID actifs pour l'identification et le diagnostic de produits en béton ou pour la surveillance du transport de marchandises dangereuses.

1.4.2 Suivi et localisation en temps réel

La technologie RTLS (Real-Time Localisation System) [52] est utilisée pour l'identification et la localisation automatique d'objets ou de personnes en temps réel, souvent dans un environnement intérieur. Si certains systèmes de localisation en temps réel utilisent uniquement des nœuds RFID actifs basiques, c'est-à-dire sans aucune extension, d'autres supportent des technologies de communication sans fil supplémentaire afin de bénéficier d'une efficacité et d'une précision accrues. Voici quelques exemples de nœuds RFID actifs permettant le suivi et la localisation en temps réel : LANDMARC [67], les Beacons, SpotOn [68], VIRE [69], ZigBee-RFID [70, 71], WIFI-RFID [72, 73], UWB-RFID [74, 69]. On peut noter que les beacons sont le type de nœuds actifs les plus utilisés dans ces systèmes.

1.4.3 Collecte de données mobile

Au cours des années, les réseaux de capteurs sans fil ont prouvé leur efficacité dans la collecte de données. Cette opération est accomplie après la phase de mesure (par les capteurs embarqués sur le nœud). Chaque nœud RFID actif envoie les données captées à un collecteur de données appelé nœud puits (« sink »), qui les transmettra à son tour à la BS (« Base Station » ou station de base). La transmission entre les nœuds du réseau et le nœud sink peut parfois être réalisée directement en un seul saut. Toutefois, si les nœuds sont éloignés du sink, leurs envois consomment beaucoup d'énergie pour trans-

mettre ces données. Une des solutions envisagées est l'ajout de nœuds intermédiaires pour acheminer les données jusqu'au sink en multi-sauts (généralement à l'aide d'algorithmes de routage). Malheureusement, le déploiement de nœuds intermédiaires présente aussi des inconvénients :

— l'utilisation de la communication multi-sauts entraîne une consommation importante d'énergie ;

— la défaillance d'un nœud peut rendre le système inutilisable.

Ainsi, l'introduction de MDC (Mobile Data Collector) permet de surmonter ces limites [75, 76, 77, 78, 79, 80, 81]. Selon les caractéristiques (topologie, géographie, distance ou taille) de la région surveillée, les MDC peuvent être des véhicules mobiles sans pilote, dont on rappelle qu'ils sont des nœuds RFID actifs en accord avec notre taxonomie présenté en section 1.3.3.4 :

— au sol, UGV (« Unmanned Ground Vehicle ») ;

— aériens, UAV (« Unmanned Aerial Vehicle ») ;

— marins, UMV (« Unmanned Marine Vehicle ») ;

— sous marins, UUV (« Unmanned Underwater Vehicle »).

En résumé, les collecteurs de données peuvent être classés en deux grandes catégories comme illustré figure 1.24.

Collecteur de données mobile au sol (UGV) : La mission de ce collecteur de données mobile consiste à visiter un réseau via un trajet terrestre et à collecter les données de tous les nœuds RFID actifs. La collecte de données au sol pourrait résoudre le problème du déséquilibre de la consommation énergétique et pourrait permettre de collecter les données des nœuds capteurs même lorsque des parties du réseau sont déconnectées. Cependant, ce type de collecte est fortement limité par la nature terrestre du trajet que

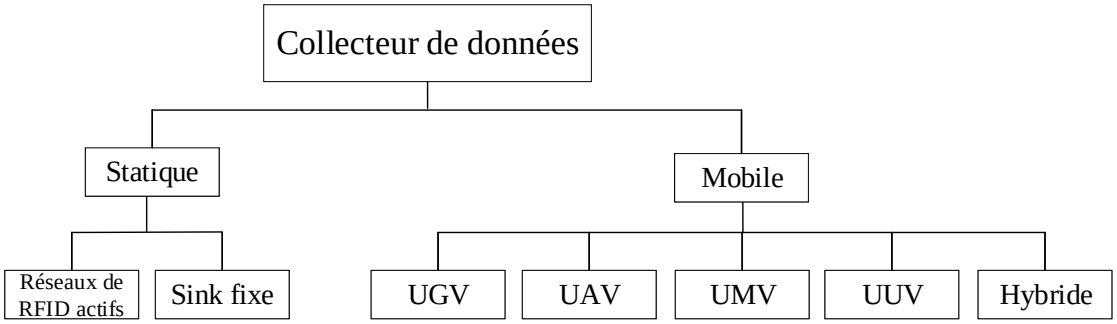


FIGURE 1.24 – Types de collecteurs de données

l'UGV doit suivre car son trajet peut contenir des obstacles comme illustré figure 1.25.

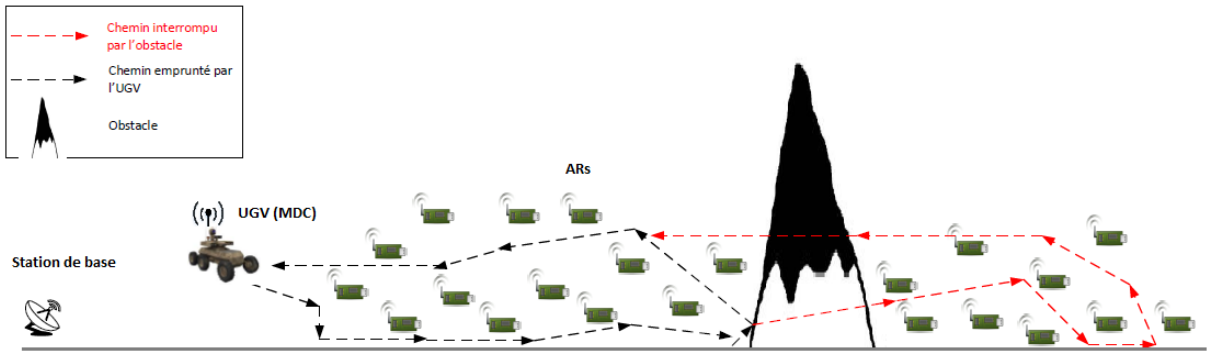


FIGURE 1.25 – Schéma d'un réseau de nœuds RFID actifs avec un collecteur de données UGV en présence d'un obstacle

Collecteur de données mobile aérien (UAV) : Initialement les véhicules aériens sans pilote (UAVs) ont été utilisés dans des applications militaires. Aujourd'hui, leur utilisation s'est démocratisée et les véhicules aériens sans pilote, peuvent être utilisés dans de nombreux domaines d'applications civiles, y compris la surveillance en temps réel, la couverture téléphonique, la télédétection, la recherche et le sauvetage, la livraison de marchandises, la sécurité et la surveillance, l'agriculture de précision, etc. Un UAV collecte les données des nœuds RFID actifs du réseaux et cette fois même en présence

d'obstacles au sol puisqu'il peut les contourner facilement dans les airs comme illustré figure 1.26. De plus, le temps de collecte de données en utilisant un UAV est réduit par rapport à celui en utilisant un UGV car la vitesse de recherche et de visite de nœuds est supérieure, cela est particulièrement vrai dans le cas de réseau à grande échelle.

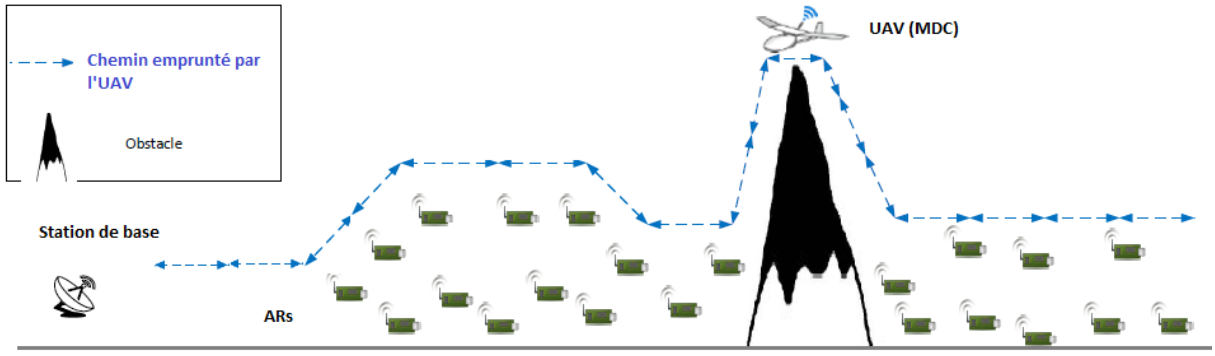


FIGURE 1.26 – Schéma d'un réseau de nœuds RFID actifs avec un collecteur de données UAV en présence d'un obstacle

Les UAVs peuvent être utilisés dans deux types de scénarios différents pour la collecte des données :

— Plusieurs UAVs, dont on rappelle qu'ils sont des nœuds RFID actifs, embarquent des capteurs afin de capter des données directement sur la zone à étudier et ils les transmettent plus tard à la station de base. Si la zone à étudier est étendue, les différents UAVs peuvent collaborer et former un réseau pour acheminer en multi-sauts les données vers la station de base. Un exemple d'application de ce scénario est la surveillance d'incendies de forêts.

— Un ou plusieurs UAVs sont utilisés pour collecter des données captées par des nœuds RFID actifs mobiles ou statiques embarquant des capteurs déployés dans la zone à étudier/surveiller ou une combinaison des deux (nœuds RFID actifs mobiles et statiques) : par exemple des UGVs pour les nœuds RFID actifs mobiles et des nœuds capteurs statiques pour les nœuds RFID actifs statiques. Ainsi, dans le domaine de

l'agriculture, des nœuds capteurs statiques peuvent être dispersés dans des champs immenses afin de surveiller la température et l'humidité du sol en vue d'optimiser les opérations sur les cultures qui permettront d'obtenir une meilleure récolte. Les données mesurées peuvent être ensuite collectées par un drone pour être remises à l'exploitant agricole.

1.5 Défis et futur des systèmes RFID actifs

L'interopérabilité est un des principaux défis pour les nœuds RFID actifs. En effet, au vu du grand nombre de nœuds RFID actifs existants dont beaucoup sont dotés de technologies de communication différentes, des normes et des protocoles de communication doivent être développés pour permettre l'intéropérabilité.

Si les nœuds actifs se présentent sous toutes les formes, types et capacités en termes de stockage, de calcul et de sources d'énergie, comme nous l'avons vu dans ce chapitre, les niveaux de sécurité et de confidentialité varient également d'un nœud à l'autre. Aussi, afin de satisfaire la nature hétérogène des nœuds, un second défi consiste à mettre en place des mécanismes de sécurité flexibles et efficaces pour protéger les données sensibles et critiques lors de leur stockage mais aussi lors de leur transits au travers des différents nœuds dans les réseaux (même à grande échelle).

Si aujourd'hui la plupart des nœuds RFID actifs existants sont statiques (attachés à un objet/personne), demain la mobilité introduite via le développement de nœuds RFID actifs mobiles (drones, robots, véhicules, etc.) devrait contribuer à l'évolution de l'IoT en M-IoT (« Mobile-IoT » ou IoT mobile).

Conclusion

Dans ce chapitre nous avons présenté la technologie RFID en général, en commençant par ses origines, le système RFID actif IFF. Puis, assez logiquement, nous avons montré comment les tags RFID passifs ont autrefois été la source de l'essor de la technologie RFID. Ensuite, nous nous sommes focalisés sur les nœuds RFID actifs qui sont l'objet de cette thèse. Nous avons commencé par présenter leur architecture et les différents composants, leur source d'énergie et leur technologie de communication. Puis, par la suite, nous avons classé les grands types de nœuds RFID actifs les plus utilisés selon les critères étudiés précédemment en donnant des exemples pour chaque type. Enfin, nous avons présenté les nouvelles applications basées sur ces nœuds.

Le chapitre suivant, présentera un état de l'art de la sécurité des systèmes RFID actifs en dégageant les spécificités de ceux-ci par rapport aux systèmes RFID en général.

Chapitre 2

Sécurité des systèmes RFID actifs

Introduction

Les systèmes RFID étant utilisés dans de nombreux domaines, ils sont souvent déployés dans des environnements hostiles dans lesquels ils sont susceptibles d'être la cible d'attaques. Pour certaines applications, par exemple dans les domaines militaires ou médicaux, assurer un haut niveau de sécurité est primordial. La sécurité d'un système RFID consiste souvent à garantir un ensemble de propriétés (telles que la confidentialité, la disponibilité, l'authentification, l'intégrité). Pour cela, ces systèmes doivent utiliser des primitives cryptographiques et des mécanismes de sécurité adaptés à leurs contraintes intrinsèques. C'est pourquoi l'un des challenges majeurs étudiés dans la littérature consiste à mettre en œuvre des mécanismes de sécurité efficaces.

Comme nous l'avons déjà souligné dans le chapitre précédent, par le passé, l'attention des chercheurs s'était essentiellement focalisée sur les systèmes RFID passifs, y compris du point de vue de la sécurité pour laquelle de très nombreux protocoles ont été proposés. Toutefois, avec l'évolution des systèmes RFID actifs et l'émergence de nouvelles applications à grande échelle (généralement en mode serverless comme nous l'avons vu en

sections 1.2.4.2 et 1.3.2.4) les utilisant, les contraintes deviennent plus strictes et la sécurité devient plus critique. Par conséquent, tout en visant l'efficacité, de nouvelles solutions et de nouveaux mécanismes plus sophistiqués doivent être mis en place pour garantir le haut niveau de sécurité attendu.

Dans ce chapitre, nous souhaitons présenter un état de l'art de la sécurité des RFID actifs. Toutefois, comme celle-ci est assez proche de celles des systèmes RFID passifs, nous présenterons dans les prochaines sections, à chaque fois le contexte générique (passifs et actifs) et nous préciserons ce qui est propre au contexte des RFID actifs chaque fois qu'il y aura des différences. Ainsi, nous commençons par détailler et classer les exigences de sécurité et de protection de la vie privée des systèmes RFID. Puis, nous étudions les menaces et leurs classifications dans la littérature. Les différentes attaques les plus connues seront aussi détaillées. Nous détaillons ensuite les limites inhérentes des systèmes RFID pour leur sécurisation et les mécanismes de sécurité de la littérature pour faire face aux attaques présentées. Ensuite, nous analysons la sécurité des protocoles sans serveur les plus cités dans la littérature. Nous concluons ce chapitre par un résumé des sections précédentes et sur notre vision pour proposer une solution efficace offrant un haut niveau de sécurité pour les systèmes RFID actifs.

2.1 Exigences de sécurité et de protection de la vie privée

Dans cette section, nous présentons les différentes exigences de sécurité et de protection de la vie privée que l'on peut trouver dans un système RFID passif ou actif. Comme ces exigences sont génériques, nous n'avons pas dégagé de spécificités dans le contexte des RFID actifs.

2.1.1 Exigences de sécurité

La sécurité peut se définir par un ensemble de propriétés (dites propriétés de sécurité) [82, 83, 84] :

— Disponibilité : La disponibilité est la propriété d'un système d'être accessible à la demande par un utilisateur autorisé à y accéder. Dans le cadre des systèmes embarqués, comme les systèmes RFID, il s'agit essentiellement d'éviter les attaques par déni de service d'utilisateurs non autorisés. Assurer la disponibilité c'est donc offrir la garantie pour les utilisateurs autorisés, de pouvoir accéder à des informations, des ressources ou des services donnés lorsqu'ils le souhaitent.

— Intégrité : L'intégrité consiste à empêcher des utilisateurs non autorisés d'un système d'écrire des informations pour lesquelles ils n'ont pas le droit. En particulier pour le contexte des communications, assurer l'intégrité se résume souvent à détecter les modifications des informations transmises.

— Confidentialité : La confidentialité est le pendant de l'intégrité et elle consiste à empêcher des utilisateurs non autorisés de lire des informations pour lesquelles ils n'ont pas le droit. C'est la propriété de « *secret* » attachée aux informations. Pour assurer la confidentialité dans le contexte des communications ou lors du stockage [1], l'outil principal est la cryptographie.

— Authentification : L'authentification consiste à prouver la validité de l'information fournie. On distingue deux types d'authentification :

  — *Authentification d'un tiers :* dans le cas de deux entités communicantes, l'information à prouver est l'*identité*. L'authentification peut être simple (c'est-à-dire une seule partie prouve son identité) ou mutuelle (c'est-à-dire les deux entités

1. Sur un système multi-utilisateurs, l'utilisation de mécanismes de contrôle d'accès peut être mis en œuvre pour assurer la confidentialité.

doivent prouver leur identité respective).

— *Authentification de la source des données :* cela consiste à prouver que les données proviennent réellement de l'entité communicante déclarée.

— Non-répudiation : La non-répudiation, dans le cadre des communications, est la garantie qu'un émetteur ne peut nier avoir transmis des messages.

Tout système s'appuyant sur la technologie radio-fréquence pouvant faire l'objet d'interception, d'injection ou de brouillage du signal entre l'émetteur et le récepteur, les exigences de confidentialité, d'intégrité et, selon le contexte de disponibilité, sont très prégnantes.

2.1.2 Exigences de protection de la vie privée

De part son histoire, la protection de la vie privée est intrinsèquement liée aux RFIDs et les systèmes doivent classiquement assurer deux propriétés : anonymat et non-traçabilité.

— Anonymat : L'anonymat peut être défini comme un sous-ensemble de la confidentialité restreint à une donnée particulière : l'*identité*. En particulier, dans le contexte des communications, assurer l'anonymat consiste à empêcher un utilisateur non autorisé d'identifier l'expéditeur et/ou le destinataire d'un message. L'anonymat et la confidentialité sont souvent des pré-requis pour assurer la protection du secret des communications.

— Non-traçabilité : La non-traçabilité vise à garantir qu'un attaquant ne peut pas identifier et/ou localiser une entité en utilisant des observations des communications.

2.2 Menaces et attaques

Afin de bien appréhender les exigences de sécurité requises pour sécuriser les systèmes RFID, il est indispensable d'étudier et de classifier les menaces et attaques les concernant.

2.2.1 Classification des menaces et attaques

Avoine et al. [85] se sont focalisés sur les menaces concernant la vie privée. Ils ont divisé ces menaces en deux catégories : la fuite d'information et la traçabilité. Ils ont en particulier identifié des relations entre les couches protocolaires (c'est-à-dire couche physique, couche communication et couche application) et la traçabilité.

Garfinkel et al. [86] ont également mis l'accent sur la vie privée et ont divisé les menaces en deux catégories : les menaces contre la sécurité des données et celles contre la vie privée.

Karygiannis et al. [87] ont proposé un modèle de taxonomie détaillée des risques RFID. Ils ont divisé les risques en trois catégories : les risques liés aux réseaux, les risques liés aux processus métiers et les risques liés à l'intelligence économique.

Ding et al. [88] ont proposé un modèle de taxonomie des menaces à deux niveaux. Le premier niveau classe les menaces selon les trois couches du modèle de communication : menaces de la couche application, menaces de la couche de communication et menaces de la couche physique. Le second niveau s'intéresse quant à lui aux types d'attaques spécifiques aux systèmes RFIDs.

Mirowski et al. [89] ont proposé une taxonomie qui s'intéresse au comportement de l'attaquant représenté sous la forme d'un arbre d'attaques, mais cette taxonomie est axée principalement sur la couche matérielle.

Mitrokotsa et al. [90] ont divisé les menaces sur les systèmes RFID en quatre couches : celles contre la couche physique, celles contre la couche réseau-transport, celles contre la

couche application et celles contre la couche qu'ils ont appelé « *stratégique* » qui comprend tout ce qui est lié à des facteurs logistiques, des contraintes du monde réel, les compromis entre le coût et l'usage. En outre, ils ont créé une catégorie distincte dite multi-couches qui exploite les vulnérabilités de plusieurs couches.

Mitrokotsa et al. [91] ont proposé une taxonomie qui se divise en trois principales catégories d'attaques selon la partie ciblée du système RFID (c'est-à-dire attaques contre la couche physique et communication mais aussi attaques contre le système de gestion des tags RFIDs). Ensuite chaque catégorie est subdivisée en trois groupes principaux selon les propriétés de la sécurité suivantes : confidentialité, intégrité et disponibilité.

Hong et al. [92] ont classé les menaces en trois catégories. La première catégorie concerne les menaces physiques. La deuxième catégorie concerne les menaces sur le canal de communication. La troisième catégorie concerne quant à elle les menaces sur les systèmes.

Si, en examinant la littérature, on constate facilement que la quasi-totalité des travaux, pour ne pas dire tous, ne concernent que les systèmes RFID passifs, il nous a semblé important de les présenter sommairement pour proposer des solutions viables dans le cadre des systèmes RFID actifs. On notera que les différents travaux présentés sont assez similaires, en nombre de couches ou en terme de taxonomie et qu'ils peuvent au final s'appliquer assez bien aux systèmes RFID actifs.

2.2.2 Principales attaques

Contrairement à la section précédente, nous allons dans la suite présenter non seulement les principales attaques génériques (c'est-à-dire pour les systèmes RFID passifs et actifs) mais aussi des attaques spécifiques aux RFID actifs.

2.2.2.1 Attaques génériques

Les attaques sur les systèmes RFID (passifs et actifs) sont nombreuses et chacune a ses propres caractéristiques. Nous citons ci-dessous les plus connues :

— **Écoute passive (eavesdropping) :** De part la nature même du support de communication sans fil des systèmes RFID, cette attaque est incontournable et est très largement déployée. L'adversaire place une antenne entre les deux entités communicantes pour espionner passivement leurs communications.

— **Clonage :** Cette méthode consiste à produire une copie non autorisée d'une entité du système RFID (tag, lecteur ou nœud actif). La copie contient l'identifiant et les données de l'entité légitime ; elle peut donc communiquer avec les autres entités légitimes et échanger des données. En théorie, il est impossible de demander au fabriquant une copie d'un tag ou d'un nœud actif mais en pratique ; ce n'est ni difficile ni coûteux [90, 93].

— **Attaque de l'homme du milieu (MiTM, « Man-in-The-Middle attack ») :** Cette attaque est une forme d'attaque active dans laquelle l'attaquant peut modifier, injecter ou supprimer les messages entre les entités communicantes du système RFID (tag et lecteur ou nœuds actifs). En résumé, le canal de communication est entièrement contrôlé par l'attaquant.

— **Attaque relai :** L'attaque par relai est une attaque de type MiTM dans laquelle l'attaquant ne modifie pas les messages. L'attaquant se place simplement entre deux entités légitimes du système RFID (tag et lecteur ou nœuds actifs) se trouvant hors de portée de communication l'une de l'autre et il relaie tous les messages de la couche physique entre les entités légitimes, les amenant à considérer qu'ils sont à portée de communication respective. Cette attaque est très puissante car elle passe outre les

mécanismes cryptographiques utilisés. Elle est très utilisée dans les systèmes RFID passifs pour *casser* les systèmes de contrôle d'accès à des bâtiments ou pour effectuer des paiements sans contact frauduleux. Mais elle est aussi utilisée dans les systèmes RFID actifs, comme pour l'ouverture à distance des portes d'une voiture.

— **Dénis de service :** L'attaque par déni de service (DoS pour Deni of Service) peut prendre différentes formes en attaquant les différentes entités du système RFID (tags, lecteurs, nœuds actifs, la station de base ou la communication entre ces entités). Le but n'est pas de voler ou de modifier des informations, mais de désactiver le système RFID afin qu'il ne puisse pas être utilisé. L'attaque de désynchronisation est un type d'attaque DoS dans laquelle les valeurs secrètes partagées entre le tag et le lecteur ou le nœud actif et la station de base ou encore entre des nœuds actifs sont rendues incohérentes par un attaquant. La conséquence est un dysfonctionnement du système RFID puisque les deux entités ne peuvent plus se reconnaître.

— **Attaque par rejeu :** Dans un système RFID, les entités (tags/lecteur ou nœuds actifs) partagent des informations en utilisant différents protocoles. Un attaquant peut intercepter des transactions passées entre les entités du système, et les rejouer plu-tard pour tromper le système RFID et obtenir l'accès (il sera considéré comme une entité légitime).

— **Capture de nœud :** L'attaquant capture le tag passif ou le nœud actif, statique ou mobile, dans le but de récupérer l'information qu'il contient (pour éventuellement ensuite le cloner) ou pour tout simplement créer un déni de service du système (par exemple dans le cas d'un collecteur mobile de données pour l'empêcher d'atteindre sa destination, c'est-à-dire la station de base).

— **Traçabilité, suivi et désanonymisation :** Si, sur la base de messages observés/capturés, ou de réponses à des sollicitations qu'il a émises lors de une ou diffé-

rentes sessions, l'attaquant est capable :

— de retrouver l'identité d'un tag ou d'un nœud actif alors il a réussi une attaque de désanonymisation ;

— de suivre ou de tracer un tag ou un nœud alors il a réussi une attaque de traçabilité ou suivi.

2.2.2.2 Attaques spécifiques aux RFID actifs

— **Attaque interne :** Les nœuds RFID actifs fonctionnant parfois dans des réseaux, ils peuvent être vulnérables aux attaques internes. Par exemple, un nœud égoïste (c'est-à-dire souhaitant minimiser sa consommation d'énergie pour préserver sa batterie et ainsi prolonger sa durée de vie) peut refuser de participer aux tâches collectives du réseau, comme le routage, ce qui engendre une sur-consommation d'énergie des autres participants, un partitionnement du réseau et une perte de liens de communication.

2.3 Limites intrinsèques des systèmes RFID pour leur sécurisation

Malgré l'évolution vers des systèmes RFID actifs, les systèmes RFID passifs et actifs présentent un certains nombre de contraintes, souvent communes, que nous classons en trois niveaux comme illustrés figure 2.1.

Niveau dispositifs : Les dispositifs RFID (tags passifs ou nœuds actifs) à faible coût sont caractérisés par une capacité de calcul restreinte et une mémoire limitée. De plus, l'énergie de certains types de nœuds RFID actifs est aussi limitée (la batterie ne peut pas être remplacée facilement car les nœuds sont souvent déployés dans des environnements

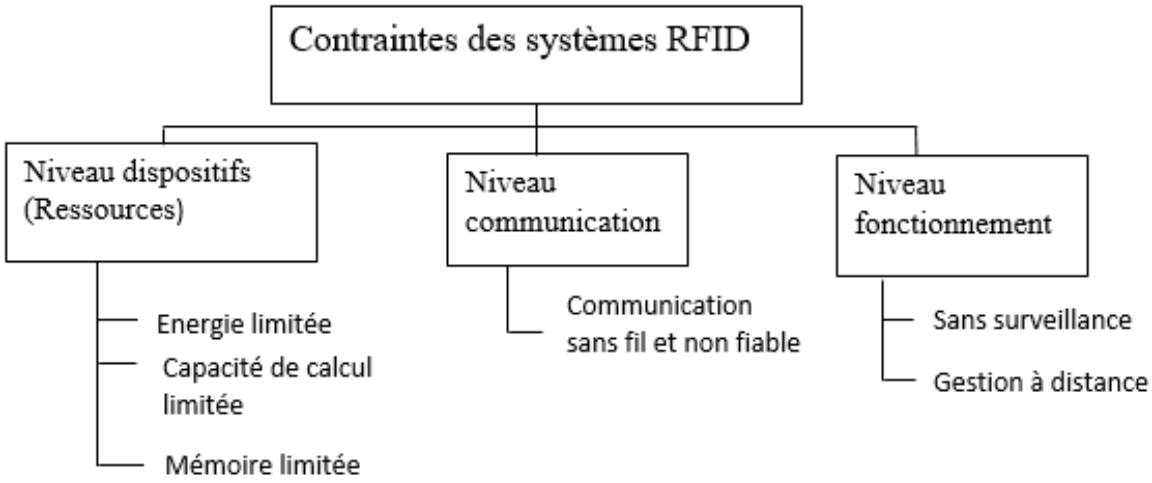


FIGURE 2.1 – Classification des contraintes des systèmes RFID

hostiles). L'ensemble de ces contraintes rendent l'utilisation des mécanismes de sécurité traditionnels difficile et nécessitent le développement de protocoles dédiés.

Niveau communication : Le canal de communication entre les dispositifs d'un système RFID étant par nature sans fil, l'information transmise est diffusée (broadcast) et toutes les transmissions peuvent être interceptées, modifiées ou rejouées par un adversaire.

Niveau fonctionnement : Les systèmes RFID fonctionnent dans des environnements ouverts et souvent hostiles. L'accès physique aux dispositifs est donc facile pour l'adversaire. Il peut par exemple, détacher un tag passif ou un nœud actif de son objet/personne et le déplacer ou le capturer. Par nature, les nœuds RFID actifs sont autonomes et peuvent communiquer et former un réseau pour transférer des données à l'utilisateur final. Toutefois, si un niveau minimum de sécurité n'est pas assuré, un adversaire peut s'infiltrer dans le réseau en utilisant les techniques décrites en section 2.2.2. Par exemple, dans les réseaux de capteurs sans fil, les nœuds sont déployés d'une manière dense et à grande échelle dans des environnements souvent hostiles et pas forcément accessibles facilement

pour les propriétaires des équipements.

Au final, les contraintes des RFID actifs sont presque identiques à celles des RFID passifs.

2.4 Mécanismes de sécurité

Dans cette section, nous pouvons distinguer les mécanismes de sécurité génériques et ceux propres aux RFID actifs.

2.4.1 Mécanismes de sécurité génériques

Les mécanismes de sécurité des systèmes RFID passifs et actifs proposés dans la littérature se divisent en deux principales catégories :

— **Les mécanismes physiques :** Les mécanismes physiques (ou matériels) ont été proposés pour protéger les tags passifs contre l'accès non autorisé par l'adversaire. Ces techniques sont basées sur l'isolation ou la désactivation des tags. Celles-ci incluent : KILL command [94], Blocker tag [31], RFID Gardian [95], Farady Cage [96], Vibrate-to-Unlock [97]. L'inconvénient de ces méthodes est qu'elles requièrent l'intervention d'un humain pour accomplir la protection à l'aide de dispositifs auxiliaires, ce qui élimine l'un des principaux avantages des systèmes RFID, à savoir l'automatisation.

— **Les mécanismes ou protocoles cryptographiques :** Par définition, la cryptographie est l'étude des techniques liées à la sécurité de l'information [98]. Un protocole cryptographique utilise une ou plusieurs primitives cryptographiques pour assurer certaines propriétés telles que celles vues en section 2.1 : la confidentialité, l'intégrité, la disponibilité, l'authentification, la non-répudiation, l'anonymat.

Dans le contexte des systèmes RFID, à la lumière des contraintes de mémoire et de puissance de calcul, la cryptographie conventionnelle s'applique mal. Il a donc été nécessaire de concevoir des protocoles cryptographiques légers utilisant des primitives légères [83] : voici plusieurs exemples, pour le chiffrement léger (symétrique ou asymétrique) [99, 100, 101, 102, 92, 103, 104, 105, 106, 107, 108, 109, 110], pour les fonctions de hachage [111, 112, 113, 114] ou encore pour les générateurs de nombres pseudo-aléatoires [115, 116, 117, 118].

Ces dernières années, les chercheurs ont beaucoup travaillé sur les protocoles limiteurs de distance (surnommé « distance-bounding » en anglais) afin de contrer les attaques par relai pour améliorer la sécurité des systèmes RFID. L'idée fondamentale de ces protocoles est de mesurer précisément le temps de transit aller-retour entre deux messages (la requête et la réponse) afin que le vérificateur s'assure que le prouveur se trouve à proximité. Si ce temps de réponse est supérieure à une limite définie par le vérificateur, celui-ci considère que le prouveur se trouve au delà de la distance souhaitée [119, 120, 121, 122]. Pour le domaine du RFID passif, NXP a mis en place ce type de mécanismes dans ses cartes Mifare plus. Dans le domaine des RFID actifs, ces mécanismes sont mis en œuvre dans les mécanismes d'ouverture des voitures récentes.

2.4.2 Mécanismes spécifiques aux RFID actifs

En raison des nouveaux champs d'application des RFID actifs, leur sécurité devient critique et nécessite des solutions plus résistantes que celles utilisées pour les RFID passifs. Les nœuds RFID actifs disposant de plus de ressources que les tags RFID passifs, les protocoles cryptographiques peuvent reposer sur des solutions plus complexes comme les courbes elliptiques, les signatures numériques, les certificats [123, 124, 125, 126, 127, 128].

Pour traiter les attaques internes, les mécanismes de gestion de la confiance semblent

les meilleurs candidats [129]. Le modèle de confiance sert à détecter les nœuds malveillants et à les différencier des nœuds légitimes d'un réseau. Plusieurs mécanismes de gestion de la confiance ont été proposés dans la littérature [130, 131, 132, 133, 134, 135].

2.5 Analyses de la sécurité des protocoles RFID sans serveur

Comme nous l'avons expliqué en section 1.3.2.4, les protocoles sans serveur étant parfaitement adaptés aux RFID actifs, il nous a semblé pertinent d'étudier la sécurité des protocoles proposés avant de développer notre proposition au chapitre 4. La plupart des solutions étudiées concernent les RFID passifs.

Tan et al. [136] ont été les premiers à proposer un protocole d'authentification serverless. Avant de commencer l'authentification, le lecteur récupère au niveau du serveur la liste des tags avec lesquels il est autorisé à communiquer. Toutefois, bien que ce protocole ait introduit la notion serverless dans les protocoles de sécurité RFID, il a été montré à plusieurs reprises dans la littérature qu'il contient plusieurs vulnérabilités :

— Il permet seulement l'authentification unilatérale d'un tag auprès du lecteur et un tag ne peut pas prouver l'authenticité du lecteur [103, 137].

— La divulgation de l'identité du lecteur et des tags ; il est possible de tracer les tags [138].

Pour résoudre ces problèmes, Lin et al. [139] ont proposé une amélioration de ce protocole. Cependant, cette amélioration n'a résolu aucun des problèmes cités. Pour améliorer les protocoles proposés par Tan et al. et Lin et al., Lee et al. [140] ont, pour leur part, proposé un protocole qui assure une authentification mutuelle. Toutefois, ce protocole

reste vulnérable à plusieurs attaques telles que celles concernant la traçabilité des tags et la divulgation d'identité [141].

Si Hoq et al. [142] ont présenté un protocole d'authentification serverless basé sur les générateurs de nombre aléatoires, il a été démontré par Dang et al. [143] que ce protocole souffrait d'une attaque par désynchronisation pour laquelle ils ont proposé une amélioration. Pourpouneh et al. [144] ont à leur tour démontré que le problème de la désynchronisation n'a pas été résolu alors que Abdolmaleky et al. [145] ont montré que le protocole est aussi vulnérable à l'usurpation d'identité du tag.

ERAP (« ECC-based RFID Authentication Protocol ») est un protocole d'authentification proposé par Ahamed et al. [146] qui utilise la cryptographie à base de courbes elliptiques (ECC pour « Elliptic Curve Cryptography ») pour générer les clés publiques afin de garantir une authentification mutuelle entre le lecteur et le tag. Cela nécessite que le lecteur et le tag supportent ECC, ce qui est une contrainte importante car même si ECC est moins coûteux que la cryptographie asymétrique classique, cela reste coûteux. Par ailleurs, Mahalle et al. [147] ont prouvé que ce protocole est vulnérable à deux attaques : le déni de service et le MiTM.

Ya-TRAP (« Yet another Trivial RFID Authentication Protocol ») est le premier protocole serverless qui utilise le temps comme un paramètre d'authentification. Proposé par Tsudik et al. [148], dans ce protocole, le lecteur doit diffuser périodiquement son temps actuel et les tags situés à proximité du lecteur stockent cette valeur de manière statique comme leur temps actuel. En cas d'émissions répétées, le tag compare la valeur du lecteur au temps mémorisé. Si le temps de diffusion est supérieur au temps stocké, le tag met à jour son temps et répond avec le calcul d'un « hash » à clé de sa clé permanente et du nouveau temps. Sinon, le tag envoie un nombre pseudo-aléatoire pour tromper un adversaire. Cependant, Tsudik lui-même a noté que son protocole est vulnérable au déni

de service.

Won et al. [149] ont proposé un système de communication sécurisée entre des UAVs et des objets intelligents, nommé eCLSC-TKEM (« efficient Certificateless Signcryption Tag Encapsulation Mechanism »), qui utilise ECC. Dans eCLSC-TKEM, le centre de génération de clés (KGC, « Key Generation Center ») insère une durée en tant que paramètre lorsqu'il génère une clé privée partielle pour un utilisateur. Ceci a pour effet que la clé privée partielle n'est valide que pour cette durée. Si le délai expire, une nouvelle clé privée doit être générée. En insérant cette durée, ils limitent l'utilisation malveillante de la clé même en cas de compromission. Pour révoquer un drone compromis suite à une capture par exemple, le KGC cesse de lui générer une clé privée partielle. Même si ce protocole prend en compte la capture du drone, il ne traite pas de la capture des objets intelligents (qui sont des nœuds RFID actifs) comme nous le ferons dans notre proposition du chapitre 4. En plus, le coût en calcul de ces opérations cryptographiques choisies est bien plus élevé que celui de notre proposition.

Akram et al. [150] ont proposé un protocole sécurisé et fiable pour renforcer la sécurité des avions au sol. Ce travail vise à permettre à un UAV d'établir une communication sécurisée avec les nœuds d'un réseau avionique de capteurs sans fil (là encore des nœuds RFID actifs) et avec d'autres systèmes de mesure de données. Cependant, les opérations cryptographiques utilisées sont là aussi bien plus complexes que celles de notre proposition.

Zhou et al. [151] ont introduit un protocole serverless pour assurer la collecte sécurisée des données à l'aide de collecteurs de données mobiles (MDC) dans les clusters d'un réseau de capteurs sans fil. Les nœuds de ce schéma sont divisés en deux types : les nœuds statiques, déployés en clusters pour capter des données ; tandis que les nœuds mobiles agissent en tant que MDC. Ce protocole est résilient contre les fausses données générées par des nœuds malveillants. Cependant, il ne protège pas contre les attaques de capture

de MDC et/ou des nœuds contrairement à notre proposition.

Mtita et al. [152] ont proposé un protocole serverless léger, sécurisé et respectant la vie privée, adapté de [153]. Ils ont utilisé des UAVs pour effectuer des opérations d'inventaire et de recherche efficace sur des objets étiquetés avec des tags RFID dans les aéroports, tout en garantissant l'anonymat de ces tags et de leurs secrets lorsqu'un UAV est capturé et compromis. Bien que le protocole proposé nécessite peu de ressources de calcul puisqu'il est conçu pour les tags RFID passifs, il ne se concentre que sur la phase d'authentification contrairement à notre proposition qui permet à la fois la collecte sécurisée des données captés mais qui supporte aussi la capture des nœuds.

Poornima et al. [154] ont aussi proposé un protocole serverless pour assurer la collecte de données sécurisée dans les réseaux de capteurs sans fil en cluster. Ce travail a porté sur l'identification des MDC malveillants, l'attaque par rejeu sur les messages et la compromission du MDC et des nœuds. Cependant, en raison de la topologie mais aussi en raison de la gestion des clés, ce protocole devrait consommer beaucoup plus d'énergie que celui proposé au chapitre 4. De plus, le cas des nœuds isolés n'est pas pris en compte (ce qui peut se produire dans les réseau de capteurs sans fil en cluster lorsque les batteries de certains nœuds relai sont épuisées).

Conclusion

À l'issue de ce chapitre, nous pouvons conclure que :

— Peu de travaux ont été proposés pour assurer la sécurité des systèmes RFID actifs.

— Si, les systèmes RFID actifs sont caractérisés par des capacités mémoire et de calcul supérieures à celles des systèmes RFID passifs qui utilisent essentiellement des protocoles de sécurité dit ultraléger (opérations simples comme le xor, des fonctions

de hachages, des générateurs de nombres aléatoires ou une combinaison de ces dernières), ils sont aussi sujets à de nouveaux types d'attaques plus complexes. Par conséquent, leurs mécanismes de sécurité doivent être à la fois plus puissants tout en restant légers et efficaces : par exemple en utilisant des mécanismes robustes comme les algorithmes cryptographiques comme ECC ou en adaptant les solutions issues du monde du RFID passif pour les rendre plus résistantes (par exemple en les combinant avec d'autres solutions telles que les protocoles limiteur de distance).

— Si l'utilisation des mécanismes cryptographiques est efficace pour résister aux attaques externes (celles présentées en section 2.2.2.1), comme les systèmes RFID actifs peuvent fonctionner en réseaux, ils sont vulnérables aux attaques internes telles que l'égoïsme. Pour faire face à ce type d'attaques, les mécanismes de gestion de confiance nous semblent une bonne solution.

— Si la plupart des protocoles proposés dans la littérature sont des protocoles d'authentification, peu de travaux s'intéressent à la confidentialité des données, spécialement dans le contexte de la capture éventuelle des nœuds.

Dans le cadre de la proposition que nous présenterons dans la seconde partie de ce mémoire, c'est ce dernier point qui a retenu toute notre attention, en particulier dans le très prometteur mode sans serveur. Toutefois, comme nous l'avons vu dans la section précédente, la conception des protocoles de sécurité étant sujette à des erreurs et des failles ayant été découvertes dans la plupart d'entre eux bien des années plus tard, nous pensons que, pour toute nouvelle proposition de protocole, il est absolument nécessaire de procéder à la vérification de son exactitude et de sa conformité vis-à-vis des propriétés qu'il revendique si l'on veut pouvoir l'utiliser en toute confiance. C'est pourquoi le prochain chapitre traitera de la vérification formelle de protocoles.

Chapitre 3

Vérification formelle des protocoles de sécurité RFID

Introduction

La cryptographie joue un rôle essentiel dans les protocoles pour garantir la sécurité et la protection de la vie privée dans les systèmes RFID. Cependant, afin de permettre l'essor des technologies RFID sur le marché, pour des raisons économiques, les ressources embarquées (mémoires et calculatoires) sont très réduites. Dans ce contexte, les protocoles de sécurité doivent utiliser des primitives cryptographiques dites « légères ». Ainsi, beaucoup de protocoles de sécurité spécifiques aux technologies RFID ont été proposés et se sont révélés contenir des erreurs. Par conséquent, aujourd'hui, pour toute proposition de nouveau protocole de sécurité, il est indispensable de vérifier à la fois qu'il satisfait vraiment les propriétés exigées ainsi que son exactitude. Il existe deux principales méthodes de vérification, la vérification formelle (symbolique) et la vérification calculatoire [155, 156]. Dans ce chapitre, nous nous intéressons aux méthodes de vérification formelle et nous étudions leur utilisation pour détecter des failles et/ou vérifier l'exactitude des protocoles

RFID.

3.1 Vérification formelle

La vérification formelle est une combinaison de modèles mathématiques d'un protocole cryptographique et de ses propriétés de sécurité, avec une procédure efficace pour déterminer si le protocole satisfait ses propriétés [157]. Pour vérifier formellement un protocole de sécurité, un modèle formel doit être adopté. Un tel modèle comprend [158, 159] :

— Un modèle du protocole (ou une spécification du protocole)

— Un modèle de l'adversaire

— Un modèle d'exécution

— La spécification des propriétés de sécurité

Le modèle de Dolev-Yao [160] a été le premier modèle formel et c'est le plus utilisé dans la littérature. Dans ce modèle, l'intrus a une connaissance complète des messages échangés sur le système de communication et il peut supprimer, modifier et envoyer des messages arbitraires pendant l'exécution du protocole.

3.2 Classification des méthodes et outils de vérification formelle

Il est possible de distinguer trois principales catégories de méthodes de vérification formelle [161, 162] :

— les méthodes basées sur la logique ;

— les méthodes de vérification de modèles (« model checking ») ;

— les méthodes de preuve de théorèmes (« theorem proving »).

3.2.1 Méthodes basées sur la logique

Des logiques ont été construites pour raisonner sur les protocoles. Les logiques de croyances raisonnent sur ce que croient les participants du protocole [163]. La BAN-logique (Burrows, Abidi et Needham) proposée dans [164, 165] est le modèle basé logique le plus connu. Il se fonde sur des déclarations sur les messages envoyés et reçus tout au long du processus d'exécution du protocole [166]. Plusieurs extensions du modèle de la logique BAN ont été réalisées telles que : GNY-logique (Gong, Needham et Yahalom) [167] ; Brackin a lui étendu GNY à BGNY [168]. Cependant, l'inconvénient majeur de ces logiques est qu'elles ne reposent pas directement sur la sémantique opérationnelle du protocole.

— **BAN-logique :** En 1989, Burrows, Abadi et Needham [164, 165] ont développé une logique d'analyse des protocoles d'authentification, la logique BAN. Avec cette logique, toutes les primitives reposant sur les crypto-systèmes à clés publiques et à clés secrètes sont formalisées, ainsi que la notion de « *nouveau message* ». Cela permet de formaliser un protocole défi-réponse.

  La BAN-logique peut être utilisée pour répondre aux questions suivantes [169] :

  — Quelles sont les conclusions de ce protocole ?

  — Quelles hypothèses sont nécessaires pour ce protocole ?

  — Le protocole utilise-t-il des actions inutiles qui peuvent être omises ?

  — Est-ce que le protocole chiffre tout ce qui pourrait être envoyé en clair, sans affaiblir la sécurité ?

  Les étapes à suivre pour utiliser la logique BAN [170] sont :

  — Idéaliser le protocole dans le langage de la logique formelle.

- Identifier les hypothèses initiales de sécurité dans le langage de la logique BAN.
- Utiliser les productions et les règles de la logique pour en déduire de nouveaux prédicats.
- Interpréter les déclarations que vous avez prouvées par ce processus. Avez-vous atteint vos objectifs ?
- (facultatif) Éliminer les parties inutiles du protocole et recommencer.

— **GNY-logique :** La logique GNY a été générée par Gong, Needham et Yahalom en 1990 [167]. Cette logique est une amélioration de la logique BAN et a donc une nouvelle approche de la méthode. Elle présente certains avantages par rapport à la logique BAN. Les notations utilisées dans cette méthode sont les mêmes que celles utilisées dans la logique BAN. Mais ici, la méthode utilise des formules supplémentaires et introduit également de nouvelles instructions. De plus, la logique construit des postulats logiques qui diffèrent également de la logique BAN.

Contrairement à l'analyse de protocole de la logique BAN, la logique GNY ne commence pas son analyse en idéalisant le protocole d'origine. Il suffit de simples transformations des messages pour obtenir une forme permettant une manipulation directe dans la logique [170].

— **BGNY-logique :** La logique BGNY, introduit en 1996 par Brackin [168] est une version étendue de la GNY-logique. De la même manière que la logique GNY, la logique BGNY ne traite que l'authentification. Cependant, la BGNY-logique étend la logique GNY en incluant la possibilité de spécifier des propriétés de protocole à des étapes intermédiaires et de spécifier des protocoles qui utilisent plusieurs opérations de chiffrement et de hachage, des codes d'authentification des messages (MAC), des fonctions de hachage à clés et des algorithmes d'échange de clés.

3.2.2 Méthodes de vérification de modèles (model checking)

Ces méthodes de vérification de modèles (« model checking »), aussi appelées méthodes d'exploration d'états, consistent à construire un modèle d'un protocole et à vérifier que chaque état atteignable satisfait certaines propriétés [171, 160].

Les méthodes de vérification de modèles peuvent être divisées en deux catégories.

— ***Vérification bornée :*** Cela consiste à explorer l'ensemble de l'espace des attaques possibles, pour un nombre borné de sessions et pour des messages de taille bornée et à rechercher un état particulier où les propriétés de sécurité sont violées et générer un contre-exemple en exploitant une trace depuis cet état jusqu'à un état initial. Les exemples d'outils de cette catégorie incluent : Casper/FDR [172] (qui a été utilisé pour découvrir l'attaque contre le protocole à clé publique Needham-Schroeder), Murphi [173], Brutus [174], CL-ATSE [175], OFMC [176], AVISPA [177], Scyther [178], Athéna [179].

— ***Vérification non bornée :*** La deuxième catégorie est la vérification de modèles utilisant le « strand space » proposé par Fabrega, Herzog et Guttman [180]. Ils définissent un « strand » comme une séquence d'événements qu'un agent (un agent honnête ou un intrus) peut exécuter dans un protocole et un strand space comme un ensemble de strands définissant l'activité des agents. Les outils de cette catégorie incluent : Scyther [178], Athéna [179], ProVerif [181] et Tamarin [182].

Nous décrivons ci-dessous quelques outils de ces deux catégories.

— **ProVerif :** ProVerif est basé sur une représentation abstraite du protocole par un ensemble de clauses de Horn et sur un algorithme de résolution de ces clauses. Il utilise comme langage de spécification une variante du pi calculus appliqués (« applied pi calculus »), une extension du pi calculus avec prise

en charge de types et de la cryptographie. ProVerif traduit les spécifications décrivant le protocole (selon le modèle présenté en figure 3.1) en clauses de Horn. Cet outil présente les caractéristiques suivantes :

- Le processus de vérification est entièrement automatique. L'utilisateur ne donne que la spécification du protocole et les propriétés à vérifier.
- Il peut gérer une grande variété de primitives cryptographiques (telles que chiffrement symétrique et asymétrique, signatures, fonctions de hachage) définies par des règles de réécriture mais il supporte aussi celles définies par des équations.
- À la différence des outils de la méthode bornée, il peut vérifier les protocoles sans limiter arbitrairement le nombre de sessions exécutées (même en parallèle) du protocole ou la taille des messages. Ceci permet d'obtenir des preuves réelles des propriétés de sécurité.
- Il peut vérifier la « secrecy » et la correspondance. Il prend également en charge l'équivalence (un adversaire ne peut pas distinguer deux processus du protocole) même si, en raison d'approximation excessive, il peut signaler de fausses attaques.

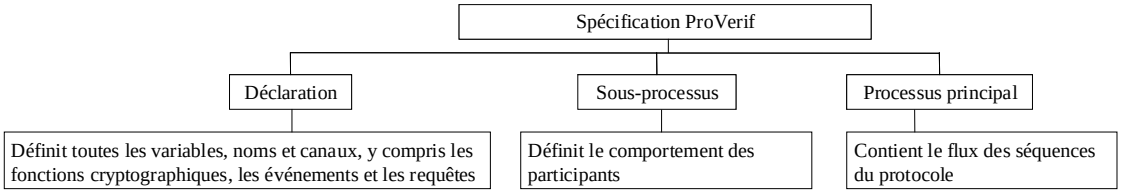


FIGURE 3.1 – Spécification avec l'outil ProVerif

- **AVISPA :** AVISPA prend en entrée une spécification des protocoles et de leurs propriétés écrites à l'aide d'un langage formel de haut niveau appelé HLPSL

(« High Level Protocol Specification Language »). Comme illustré figure 3.2, HLPSL est un langage basé rôles. La spécification est ensuite traduite en IF (« Intermediate Format » ou format intermédiaire). Le résultat de cette traduction est ensuite utilisé comme entrée pour les différents back-ends.

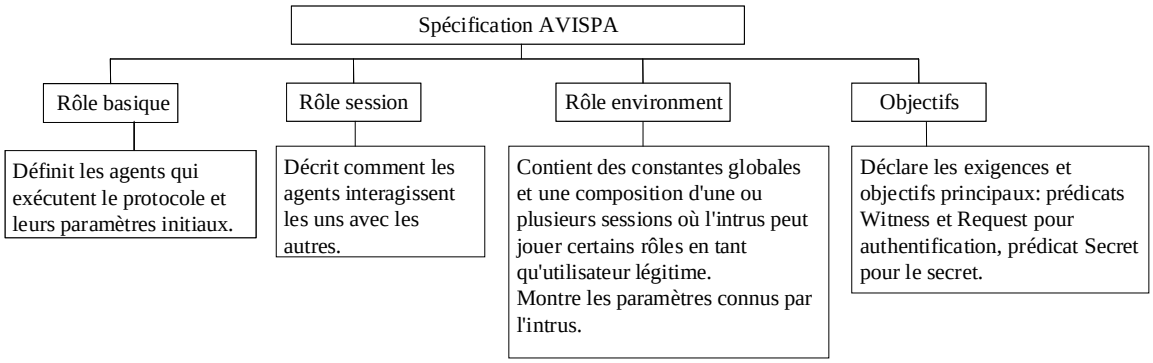


FIGURE 3.2 – Spécification avec l'outil AVISPA

AVISPA intègre les back-ends suivants :

— OFMC (« On-the-Fly Model-Checker ») : effectue une vérification bornée en explorant le système de transitions décrit par une spécification IF. Il supporte la spécification des opérateurs à propriétés algébriques tels que le OU exclusif ou encore l'exponentielle.

— CL-AtSe (« CL-based Model-Checker ») : c'est un outil basé sur des techniques de résolution de contraintes et implémentant une procédure de décision pour traiter le problème d'insécurité pour un nombre borné de sessions en implémentant un algorithme d'unification spécifique et en considérant les opérateurs à propriétés algébriques comme le OU exclusif.

— SATMC (« SAT-based Model-Checker ») : il permet de construire une formule propositionnelle codant un déploiement borné du système de transi-

tion IF, l'état initial et l'ensemble des états représentant la violation des propriétés de sûreté spécifiées en IF.

— TA4SP (« Tree Automata based on Automatic Approximations for the Analysis of Security Protocols ») : son but est de permettre la vérification de protocoles avec un nombre non borné de sessions. C'est une approche pour la validation de protocoles complémentaires aux autres approches, plutôt destinées à la capture d'attaque(s).

AVISPA est particulièrement bien adapté pour l'analyse des propriétés de sécurité d'un nombre borné de sessions (à l'aide d'OFMC, de CL-Atse et de SATMC, mais TA4SP ne fournit qu'une prise en charge limitée dans le cas non borné).

— **Scyther :** Scyther utilise une approche de vérification de modèles sans limite et applique un algorithme de recherche en arrière sur les modèles (trace). Ces modèles décrivent un ensemble d'événements ordonnés (les « strands ») séparément qui doivent se produire dans les traces du protocole pour que ces modèles puissent être vérifiés. La présence d'événements dans les traces du protocole est vérifiée en les faisant correspondre à des critères spécifiques définis dans la sémantique du protocole.

Scyther offre aussi la possibilité d'introduire une limite et d'appliquer une vérification bornée du modèle si la recherche non bornée ne termine pas. Lors de l'utilisation d'une borne, le résultat n'est valide que pour la limite spécifique du nombre de sessions. Cependant, Scyther ne prend pas en charge les théories équationnelles définies par l'utilisateur et le flux de contrôle ne peut pas être modélisé. Ces facteurs limitent la classe de protocoles pouvant être modélisés. Cet outil prend en entrée un fichier `.spdl` (langage de description de proto-

cole de sécurité), qui inclut une spécification du protocole et les propriétés de sécurité revendiquées.

— **Casper/FDR :** est un compilateur qui prend une description de haut niveau du protocole et ses exigences de sécurité et produit un code CSP (« Communicating Sequential Processes ») vérifié par FDR (« Failure Divergence Refinements »). Cet outil est utilisé pour vérifier les exigences d'authentification et de confidentialité du protocole. Pour cela, Casper/FDR vérifie les spécifications d'authentification et de confidentialité en examinant les événements associés. En d'autres termes, les événements `Running` et `Commit` sont associés aux spécifications d'authentification. Il vérifie les spécifications de confidentialité via un événement appelé `Claim_Secret`, qui est exécuté par les deux parties du système.

3.2.3 Méthodes de preuve de théorèmes

Utilisées pour prouver l'exactitude d'un protocole de sécurité, les méthodes de preuve de théorèmes (« theorem proving ») utilisent les axiomes et les règles [183]. On distingue deux catégories de méthodes de démonstration de théorèmes : les méthodes de logique du premier ordre comme NPA [184] et Coral [185], et les méthodes utilisant une logique d'ordre supérieur, comme exemple Isabelle/HOL [186]. Isabelle/HOL est l'outil le plus cité dans la littérature.

— **Isabelle/HOL :** Isabelle est un outil d'assistance de preuve générique. Il permet aux formules mathématiques d'être exprimées dans un langage formel et fournit des outils pour prouver ces formules dans un calcul logique. L'application principale est la formalisation de preuves mathématiques, et en particulier de vérifications formelles, qui consistent à prouver l'exactitude du matériel informatique ou des

logiciels et à prouver les propriétés des langages et des protocoles informatiques. La variante la plus répandue d'Isabelle à l'heure actuelle est Isabelle/HOL, qui fournit un environnement de démonstration de théorèmes logiques d'ordre supérieur.

3.3 Exemple de vérification formelle du protocole STCP pour des RFID actifs

Maintenant que nous avons décrit l'importance de la vérification formelle des protocoles de sécurité et que nous avons présenté les grandes méthodes et les outils, nous allons présenter l'application de deux outils à la vérification du protocole STCP (« Secure and Trusted Channel Protocol ») [187] pour des RFIDs actifs, ici mobiles puisqu'il s'agit d'une flotte de drones. Cet exemple permettra d'expliquer la mise en œuvre de deux outils de vérification formelle : Casper/FDR et AVISPA. La réalisation de la vérification formelle AVISPA a été ma contribution à l'article [187] alors que la réalisation de la vérification formelle Casper/FDR est une de celles des autres auteurs.

Les objectifs du protocole STCP sont :

— de fournir l'assurance que toutes les entités en communication peuvent se faire confiance et peuvent faire confiance à leurs états internes (sécurisés) matériels et logiciels ;

— d'établir un processus équitable d'échange de clés entre toutes les entités communicantes afin de leur fournir un canal sécurisé pour communiquer.

Le tableau 3.1 présente les notations utilisées dans le protocole alors que le tableau 3.2 illustre les messages du protocole.

Les RFID actifs considérés dans ce protocole sont équipés de TEM (« Trusted Execution Module ») pour réaliser les opérations d'attestation matérielle et logicielle. Préa-

lablement à leur déploiement les différentes entités sont configurées avec la clé publique du propriétaire des différents nœuds afin de pouvoir vérifier la validité de la clé publique contenue dans C_{SE}. Les différentes entités sont également pré-configurées avec la clé publique de l'autorité qui a réalisé la certification sécuritaire du TEM des ARs afin de pouvoir vérifier la validité de la clé publique contenue dans $C_{TEM_{SE}}$. Enfin, chaque entité est également pré-configurée avec la valeur d'assurance de sécurité de chacun de ses partenaires, ce qui leur permettra d'attester qu'ils sont dans des états sécurisés et de confiance.

3.3.1 Vérification avec l'outil Casper/FDR

Le script de la spécification est présenté dans l'annexe B.1 :

- Les variables et leurs types sont déclarés dans la partie `#Free variables` y compris les agents (`SE1,SE2`).
- Chaque agent du système est représenté par un processus CSP dans la partie `#Processes` en utilisant les mots clés `INITIATOR` et `RESPONDER`.
- Les clés secrètes et publiques et toute autre fonction sont déclarées dans la partie `#Functions`.
- Les agents et leurs paramètres instanciés devant figurer dans le système à contrôler doivent être listés dans la partie `#System`.
- La partie `#Intruder Information` contient les opérations que l'intrus peut effectuer sur le système. Cela est possible en listant son identifiant et les variables auxquelles il a accès
- Les propriétés de sécurité à vérifier sont exprimées dans la partie `#Specification` :

```
#Specification
Aliveness(SE2, SE1)
```

Tableau 3.1 – Notations utilisées dans la description du protocole STCP

$SE1$	:	Représente le nœud RFID actif 1.
$SE2$	:	Représente le nœud RFID actif 2.
$A \rightarrow B$	:	Message envoyé par l'entité A à l'entité B.
TEM_X	:	Représente le TEM de l'entité X
X_i	:	Représente l'identité de l'entité X.
g^{rx}	:	L'exponentiation Diffie-Hellman generée par l'entité X.
C_X	:	Certificat de l'entité X.
N_X	:	Nombre aléatoire généré par l'entité X.
$X \| Y$	:	Représente la concaténation des données X et Y dans cet ordre.
$[M]_{K_a}^{K_e}$	:	Le message M est chiffré par la clé de chiffrement de session K_e et alors le MAC est calculé en utilisant la clé de session MAC K_a. Les clés K_e et K_a sont générées lors de l'exécution du protocole.
$Sign_X(Z)$	:	Signature générée sur les données Z par l'entité X en utilisant un algorithme de signature [188].
$H(Z)$	:	Résultat de la génération du hachage de la donnée Z.
$H_k(Z)$	:	Résultat de la génération du hachage à clé de la donnée Z en utilisant la clé k.
S_{Cookie}	:	Cookie de session généré par une des entités communicantes. Il contient des informations sur la session et facilite la protection contre les attaques de types DoS tout en (éventuellement) permettant la reprise de session interrompue.
VR_{A-B}	:	Requête de validation envoyée par l'entité A à l'entité B. En réponse l'entité B fournit son assurance de sécurité et de confiance à l'entité A.
SAS_{A-B}	:	L'assurance de sécurité générée par l'entité A pour fournir à l'entité demandeuse, B, une preuve de confiance de son état.

Tableau 3.2 – Le protocole STCP (« Secure and Trusted Channel Protocol »)

1.	$SE1 \rightarrow SE2$	:	$SE1_i \| SE2_i \| N_{SE1} \| g^{r_{SE1}} \| VR_{SE1-SE2} \| S_{Cookie}$
2.	$SE2 \rightarrow SE1$	:	$SE2_i \| SE1_i \| N_{SE2} \| g^{r_{SE2}} \| [Sign_{SE2}(SE2-Data) \| Sign_{TEM_{SE2}}(SE2-Validation) \|$
		:	$C_{SE2} \| C_{TEM_{SE2}}]_{K_a}^{K_e} \| VR_{SE2-SE1} \| S_{Cookie}$
		:	$SE2-Data = H(SE2_i \| SE1_i \| g^{r_{SE1}} \| g^{r_{SE2}} \| N_{SE1} \| N_{SE2})$
		:	$SE2-Validation = SAS_{SE2-SE1} \| N_{SE1} \| N_{SE2}$
3.	$SE1 \rightarrow SE2$	:	$[Sign_{SE1}(SE1-Data) \| Sign_{TEM_{SE1}}(SE1-Validation) \| C_{SE1} \| C_{TEM_{SE1}}]_{K_a}^{K_e} \| S_{Cookie}$
		:	$SE1-Data = H(SE1_i \| SE2_i \| g^{r_{SE2}} \| g^{r_{SE1}} \| N_{SE2} \| N_{SE1})$
		:	$SE1-Validation = SAS_{SE1-SE2} \| N_{SE2} \| N_{SE1}$

```
Aliveness(SE1, SE2)

Agreement(SE2, SE1, [EnMaKey])

Secret(SE2, EnMaKey, [SE1])

Secret(SE1, U, [SE2])
```

1. Les secrets sont modélisés à l'aide du prédicat `Secret`.

   Par exemple : `Secret(SE2, EnMaKey, [SE1])` signifie que `SE2` croit que `EnMaKey` est un secret qui ne devrait être connu que par `SE1`.

2. L'authentification est modélisée par le prédicat `Agreement`.

   Par exemple, `Agreement(SE2, SE1, [EnMaKey])` signifie que `SE2` est authentifié auprès de `SE1` avec `EnMaKey`

3. La spécification `Aliveness(SE1, SE2)` signifie que si `SE2` pense avoir terminé avec succès une exécution du protocole avec `SE1`, alors `SE1` a déjà exécuté le protocole.

3.3.2 Vérification avec l'outil AVISPA

Le script de la spécification est présenté dans l'annexe B.2 :

— Il existe deux agents `Se_1`, `Se_2` et ils utilisent les trois fonctions de hachage `H, Hk, MAC` et une signature `SIGN`.

— Chaque agent possède deux clés publiques respectivement `PK1, PKTM1` et `PK2, PKTM2`.

— Les secrets calculés lors des échanges sont modélisés à l'aide du prédicat `secret`.

`secret (Kdh,sec_kdh1, {A,B})`, `secret(Ke, sec_ke1, {A,B})`, `secret(Ka, se_ka1, {A,B})` et `secret (Kdh,sec_kdh2, {B,A})`, `secret(Ke, sec_ke2, {B,A})`, `secret(Ka, sec_ka2, {B,A})`, et ils sont gérés par les identifiants de protocole `sec_kdh1`, `sec_ke1`, `sec_ka1`, `sec_kdh2`, `sec_ke2`, `sec_ka2`. Les paramètres `Kdh, Ke, Ka` sont gardés secrets pour `Se_1` et `Se_2`.

— L'authentification mutuelle est réalisée via `witness` et `request` : c'est-à-dire `witness(A, B, ns1, NS1)` et `witness(B, A, ns2, NS2')`, `request(A, B, ns2, NS2)` et `request(B, A, ns1, NS1)`.

`witness(A, B, ns1, NS1)` déclare que l'agent `A` prétend être l'homologue de l'agent `B` et lui envoie la valeur `NS1`. `ns1` est l'identifiant de l'authentification en `NS1` indiquée dans la section `goals` alors que `request(B, A, ns1, NS1)` déclare que l'agent `B` prétend être en communication avec l'agent `A` et accepte la valeur `NS1`.

— L'environnement contient les constantes globales et la composition d'une ou plusieurs sessions. L'intrus participe en tant que session concrète au protocole d'exécution.

Conclusion

Comme la conception de protocoles de sécurité est une activité particulièrement sujette à l'introduction d'erreur ou de faille ainsi que le prouve le grand nombre de défauts ayant

été détectés des années plus tard sur la plupart des protocoles proposés, il est impératif de vérifier leur exactitude.

Après avoir exposées, différentes méthodes de vérification formelle, la méthode de vérification de modèles nous semble être la meilleure candidate pour réaliser une vérification formelle des protocoles de sécurité RFID grâce à ses avantages [189] :

— Elle ne nécessite pas de preuves.

— Elle est totalement automatique et fournit une couverture complète.

— Elle traite non seulement l'exactitude mais également l'inexactitude des protocoles : c'est-à-dire qu'elle génère un contre-exemple dans le cas où le programme ne répond pas aux spécifications.

— Elle fonctionne également avec une spécification partielle. En d'autres termes, l'utilisateur n'a pas à attendre que la phase de conception soit terminée pour vérifier son protocole. Elle peut donc être utilisée lors de la conception de systèmes complexes.

— Elle est plus rapide que les autres méthodes.

C'est pour cette raison que nous avons montré l'application de deux outils de vérification de modèles (Casper/FDR et AVISPA) pour vérifier le protocole STCP pour des RFID actifs dans un article accepté en 2017 à la conférence WISTP.

Dans le prochain chapitre, nous allons proposer un protocole de sécurité serverless qui vise à assurer la confidentialité des données captées par des nœuds RFID actifs et collectées par un drone (un autre RFID actif). La description de ce protocole sera détaillée et elle contiendra à la fois une analyse de sécurité informelle et la vérification du protocole à l'aide des outils AVISPA et ProVerif (ce dernier est encore un outil de vérification de modèles comme nous l'avons présenté dans ce chapitre).

Deuxième partie

Proposition d'une solution sécurisée pour les RFID actifs

Chapitre 4

Description du scénario, du protocole et validation sécuritaire

Introduction

Comme nous l'avons vu au cours du chapitre 1, les nœuds ARs englobent une large variété de périphériques qui partagent des caractéristiques communes : technologies de communication sans fil, supports de différents capteurs, ressources contraintes (énergétiques, mémoires et calculatoires). En fait, la plupart des objets communiquant sans fil dans les scénarios de l'IdO (IoT) peuvent être considérés comme des nœuds ARs (UAV, nœuds d'un réseau de capteurs, téléphones mobiles, balises intelligentes, etc.). Dans la plupart des scénarios, les nœuds étant sans surveillance dans un environnement adverse et sans connexion permanente (c'est-à-dire sans serveur), la confidentialité des données doit être assurée de l'étape de mesure à celle de la délivrance à l'autorité en charge de la traiter. En d'autres mots, les données doivent être stockées et transmises de façon sécurisée afin de contrer les attaques d'un adversaire, et même en cas de capture des nœuds. Toutefois en raison des faibles ressources des nœuds ARs, les solutions proposées doivent

rester légères pour être efficaces.

Dans ce chapitre, nous présentons notre solution de sécurité pour des nœuds ARs dans ce type de scénario sans serveur. Ainsi, nous décrirons tout d'abord une instance de ces scénarios, avant de décrire le modèle (entités, système, etc.) et les exigences de sécurité et de performance attendues. Puis nous détaillerons notre solution et nous la validerons d'un point de vue sécuritaire d'abord de manière informelle via une analyse et l'utilisation de jeux de sécurité et ensuite de façon formelle avec les outils AVISPA et ProVerif. La validation de son efficacité sera l'objet du chapitre 5.

4.1 Description du scénario

Le scénario de notre solution [190] est illustré par la figure 4.1. Les MDCs collectent des données captées par les nœuds ARs qui sont déployés dans une zone de mission. Les nœuds ARs ne peuvent pas communiquer directement avec la station de base et doivent donc confier leurs données captées à des MDCs qui peuvent transmettre ces données à leur retour à la station de base. Comme illustré figure 4.1, les nœuds ARs peuvent être équipés d'un ou de plusieurs capteurs et peuvent être statiques, c'est-à-dire des SARs, ou mobiles, c'est-à-dire des MARs. La zone de mission n'étant pas de confiance, un attaquant pourrait essayer de compromettre les MDCs ou les nœuds ARs pour en extraire des informations. Un attaquant peut aussi tenter d'espionner les communications sans fil entre les MDCs et les nœuds ARs ou de réaliser d'autres attaques, notamment la modification de messages échangés ou l'injection de fausses données.

Il existe plusieurs applications concrètes pour lesquelles le scénario proposé est plus que pertinent et spécialement dans le domaine militaire. En effet, des nœuds ARs peuvent être déployés sur un territoire ennemi distant pour capter différentes données qui seront

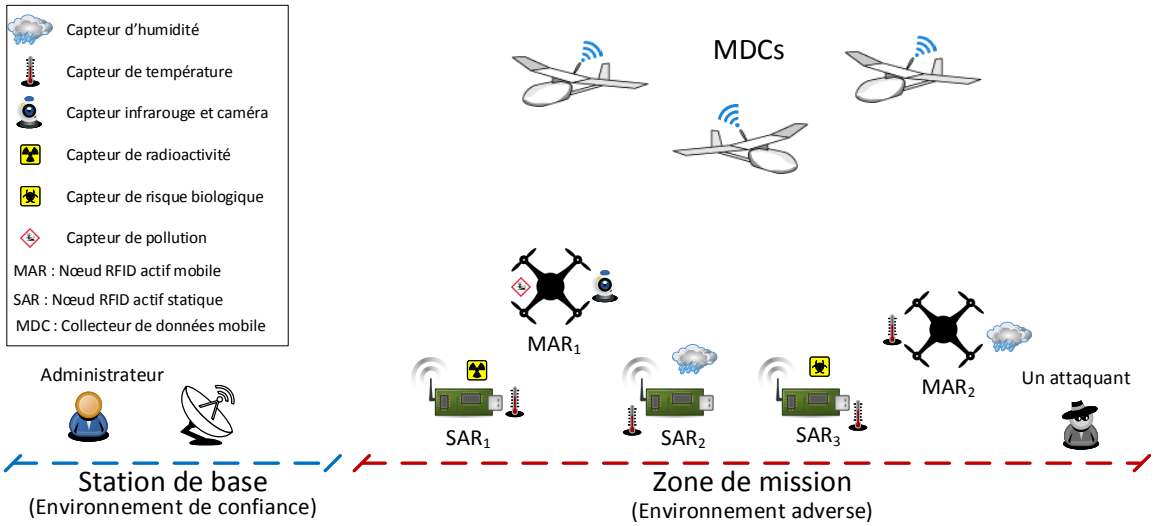


FIGURE 4.1 – Scénario auquel notre proposition s'applique

ensuite collectées par les MDCs. Le propriétaire des nœuds ARs ne souhaitant pas que l'ennemi ait accès aux données captées, leur confidentialité doit être assurée pendant les échanges et même en cas de capture de nœuds ARs ou de MDCs.

4.2 Modèle du système de collecte de données

Dans cette section, nous spécifions d'abord chaque entité impliquée dans le système et le protocole avant de présenter l'architecture générale du modèle du système décrit figure 4.2. Ensuite, nous décrivons les exigences de sécurité et de performance, les hypothèses, ainsi que les modèles de menaces et d'attaques.

4.2.1 Entités

Le modèle du système contient trois types d'entités communicantes, présentées ci-dessous avec leurs caractéristiques respectives, qui utilisent le protocole pour interagir. La

définition de chaque paramètre utilisé dans le protocole est fournie dans le tableau 4.1.

— **Le nœud RFID actif (Active RFID node) :** noté AR_j. Un nœud AR comporte des capteurs intégrés pour capter les données de la zone de surveillance. Il se caractérise par des ressources limitées en termes de stockage, de calcul, d'énergie et de portée de communication. Le nœud peut être statique (SAR) ou mobile (MAR). Chaque nœud AR_j, possède une clé secrète K_i^j qui évolue à chaque chiffrement des données captées.

— **Le collecteur mobile de données (Mobile Data Collector) :** noté MDC_l. Un MDC dispose de ressources limitées pour le stockage, le calcul et une portée de communication limitée. Il a plus d'énergie que les nœuds ARs mais il en consomme l'essentiel pour se déplacer. Un MDC possède des accréditations (« credentials ») lui autorisant la collecte de données captées par les nœuds ARs.

— **La station de base :** notée BS. Une BS est une entité de confiance et puissante, c'est-à-dire avec des capacités mémoire et de calcul illimitées. Comme présentée figure 4.2, une station de base est composée de deux parties. La première est le KDC (« Keys Distribution Center ») qui fournit les accréditations initiales aux MDCs et aux ARs à chaque nouvelle phase de pré-mission. La seconde est le « sink », qui déchiffre les données collectées renvoyées par les MDCs au cours de la phase de post-mission.

4.2.2 Modèle du système

Pour mettre en œuvre le modèle de collecte de données présenté figure 4.2, les activités de la mission peuvent être regroupées en trois phases principales : *pré-mission, mission et post-mission*. Les première et dernière phases se déroulent essentiellement dans un environnement de confiance, tandis que la phase de mission se déroule dans un environnement

hostile.

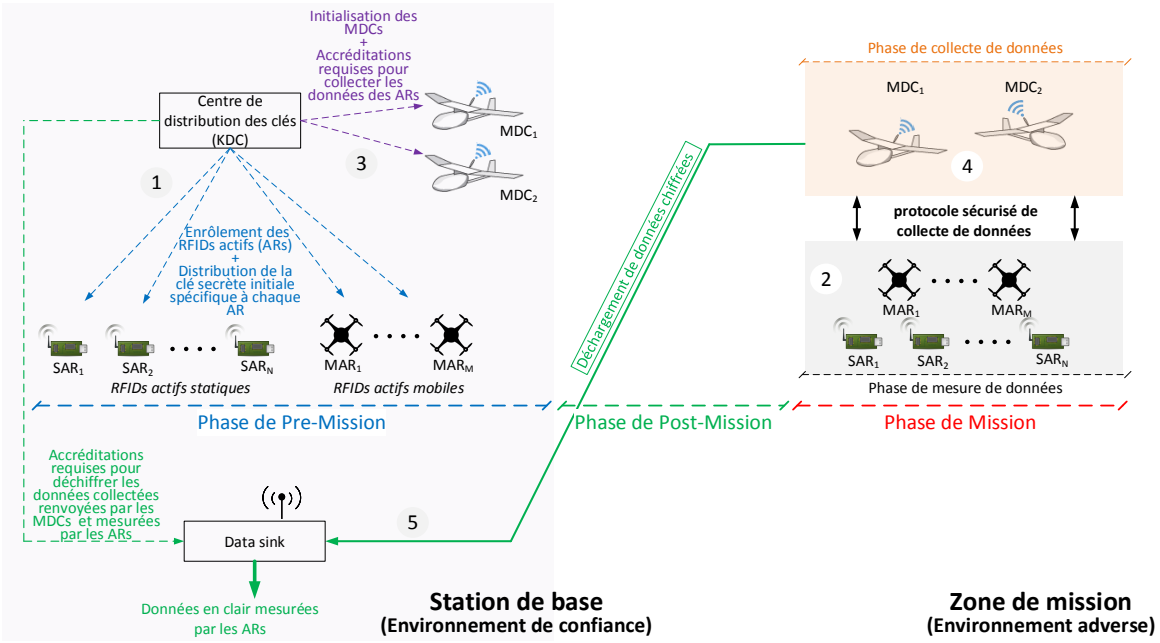


FIGURE 4.2 – Architecture globale du système de collection des données

— Dans la phase de pré-mission, le KDC fournit aux nœuds ARs leur clé secrète initiale personnelle (❶). Ensuite, ces nœuds ARs sont déployés dans la zone de mission pour commencer la phase de mesure des données.

— La phase de mission à proprement parler comprend deux sous-phases : la phase de mesure des données (❷) et la phase de collecte de données (❹). Pendant la phase de mesure des données, ces dernières sont immédiatement chiffrées et stockées dans la mémoire du nœud AR. Avant le début de la phase de collecte des données, les MDCs se connectent de manière sécurisée au KDC et demandent les informations (❸) nécessaires pour la collecte des données. Le KDC fournit aux MDCs les accréditations requises pour communiquer avec les nœuds ARs afin de permettre la collecte de données.

— Une fois la phase de mission terminée, à leur retour à la station de base, les MDCs communiquent leurs données chiffrées collectées au sink (❺). Le KDC fournit ensuite au sink les paramètres nécessaires pour déchiffrer les données collectées.

4.2.3 Exigences de sécurité

Les exigences de sécurité souhaitées pour notre protocole sont répertoriées ci-dessous :

(ES1) Les données captées doivent être stockées sous une forme chiffrée dans les mémoires des nœuds ARs pour être protégées contre leur divulgation à un attaquant. Donc, les nœuds ARs ont besoin de fonctionnalités cryptographiques telles que des algorithmes de chiffrement.

(ES2) Les nœuds ARs et les MDCs doivent supporter les fonctions de hachage et de HMAC (« Hash-based Message Authentication Codes ») qui seront utilisées dans le schéma de gestion des clés pour protéger les canaux de communication et éviter les attaques par rejeu.

(ES3) Les clés secrètes utilisées pour chiffrer les données captées doivent être protégées de toute divulgation à un adversaire. Par exemple, elles peuvent être stockées dans une mémoire inviolable et/ou peuvent être supprimées après utilisation.

(ES4) Pour assurer la confidentialité des échanges passés (« forward secrecy »), les clés secrètes doivent évoluer de manière dynamique pendant les opérations de mesure des données et être propres aux différents nœuds ARs.

(ES5) Les nœuds MDCs et ARs doivent pouvoir s'authentifier mutuellement pour éviter l'usurpation d'identité par un adversaire.

4.2.4 Exigences de performance

Pour prolonger la durée de vie à la fois des MDCs et des nœuds ARs, compte tenu de leurs ressources limitées, notamment en termes d'énergie, les opérations cryptographiques doivent être légères et les calculs doivent être minimisés.

4.2.5 Hypothèses

Dans notre proposition, les hypothèses sont les suivantes :

(H1) La BS est une entité de confiance avec des ressources illimitées et qui ne peut être compromise.

(H2) Les fonctions de hachage et les fonctions HMAC sont robustes (c'est-à-dire, à sens unique, résistantes aux collisions, etc.).

(H3) Les algorithmes de chiffrement utilisés par les ARs et la BS sont sécurisés et légers.

(H4) Les données mesurées par les capteurs des ARs sont fiables.

4.2.6 Modèles de menaces et d'attaques

Pour modéliser la sécurité de notre protocole qui sera décrit en section 4.3.3.2, nous considérons un adversaire en temps polynomial α qui l'attaque suivant les jeux de sécurité décrits ci-dessous dans le but d'obtenir un accès à des informations secrètes ou de perturber une exécution normale du protocole. Les jeux de sécurité sont conçus pour montrer les capacités, les limites et les options de l'adversaire lorsqu'il tente d'attaquer le protocole.

Jeu 1 : *α se fait passer pour un MDC*

— **étape 1.1 :** α écoute et capture plusieurs échanges entre un MDC légitime et AR_j.

— **étape 1.2 :** α envoie les messages légitimes 1 et 3 (du tableau 4.2) à AR_j.

α gagne s'il peut envoyer un message 3 valide.

Jeu 2 : *α crée une contrefaçon du nœud AR_x*

— **étape 2.1 :** α attaque physiquement AR_j pour accéder à ses données.

— **étape 2.2 :** α utilise les données d'AR_j valide pour créer une contrefaçon AR_x, où $x \neq j$.

α gagne s'il peut créer une contrefaçon AR_x et tromper le MDC légitime.

4.3 Description du protocole sécurisé de mesure et de collecte des données

Dans cette section, nous détaillons notre protocole. Afin d'assurer la confidentialité des données captées lors des échanges entre les nœuds ARs et les MDCs, le protocole utilise un mécanisme défi-réponse utilisant un algorithme de chiffrement léger et un système de gestion de clés qui reposent sur des fonctions de hachage et HMAC.

4.3.1 Notations utilisées

Le tableau 4.1 présente les notations utilisées dans notre protocole.

4.3.2 Phase de pré-mission

Le protocole proposé nécessite certaines opérations préalables à la mission, comme illustré en figure 4.3.

❶ Chaque nœud AR_j reçoit du KDC avant le déploiement (c'est-à-dire sur le site de la BS, dans un environnement sécurisé) son identifiant unique Id^j et sa propre K_0^j, l'origine de la chaîne de clés. L'origine de la chaîne de clés est utilisée par chaque nœud AR pour calculer ses propres clés secrètes suivantes afin de chiffrer

Tableau 4.1 – Notations utilisées dans la description du protocole

MDC_l	:	Un collecteur de données mobile l.
AR_j	:	Le nœud RFID actif j.
Id^j	:	L'identifiant du $j^{\text{ème}}$ nœud.
SD_i^j	:	La $i^{\text{ème}}$ donnée détectée par AR_j.
$[Z]^K$	:	Le chiffrement de la donnée Z avec la clé K.
$H(Z)$	:	Le résultat du hachage de la donnée Z.
$HMAC_K(Z)$	:	Le résultat du HMAC de la donnée Z avec la clé K.
$A\|\|B$	:	La concaténation des données respectivement A et B.
K_i^j	:	La clé secrète unique de AR_j utilisée pour chiffrer SD_i^j. Elle évolue à chaque opération de chiffrement.
ACK_i^j	:	La clé secrète unique utilisée par AR_j pour calculer $HMAC$ de chaque SD_i^j chiffrée et utilisée par MDC pour vérifier l'intégrité.
HM_i^j	:	Le résultat du calcul de $HMAC_{ACK_i^j}([SD_i^j]^{K_i^j})$.
Request	:	Le message envoyé par un MDC pour initier la communication avec les nœuds ARs en les réveillant du mode d'économie d'énergie au travers de l'interface radio.
T_i^j	:	Un ticket pour attester que le MDC a les accréditations de collecter les données reçues et pour confirmer que ces données ont été correctement reçues.
$List_{Id}$	:	La liste de tous les identifiants des AR pour lesquels le MDC est accrédité à collecter des données ; avec $List_{Id}$=(Id^1, ..., Id^M).
$List_T^j$	:	La liste de v tickets relatifs à AR_j, c'est-à-dire $List_T^j$=(T_0^j, ..., T_v^j).
$List_{ACK}^j$	:	La liste de v ACKs relatifs à AR_j, c'est-à-dire $List_{ACK}^j$=(ACK_0^j, ..., ACK_v^j).
$List_T$	:	La liste de toutes les listes des tickets relatives à chaque AR de $List_{Id}$ avec $List_T$=($List_T^1$, ..., $List_T^M$).
$List_{ACK}$	:	La liste de toutes les listes des ACK relatives à chaque AR de $List_{Id}$ avec $List_{ACK}$=($List_{ACK}^1$, ..., $List_{ACK}^M$).

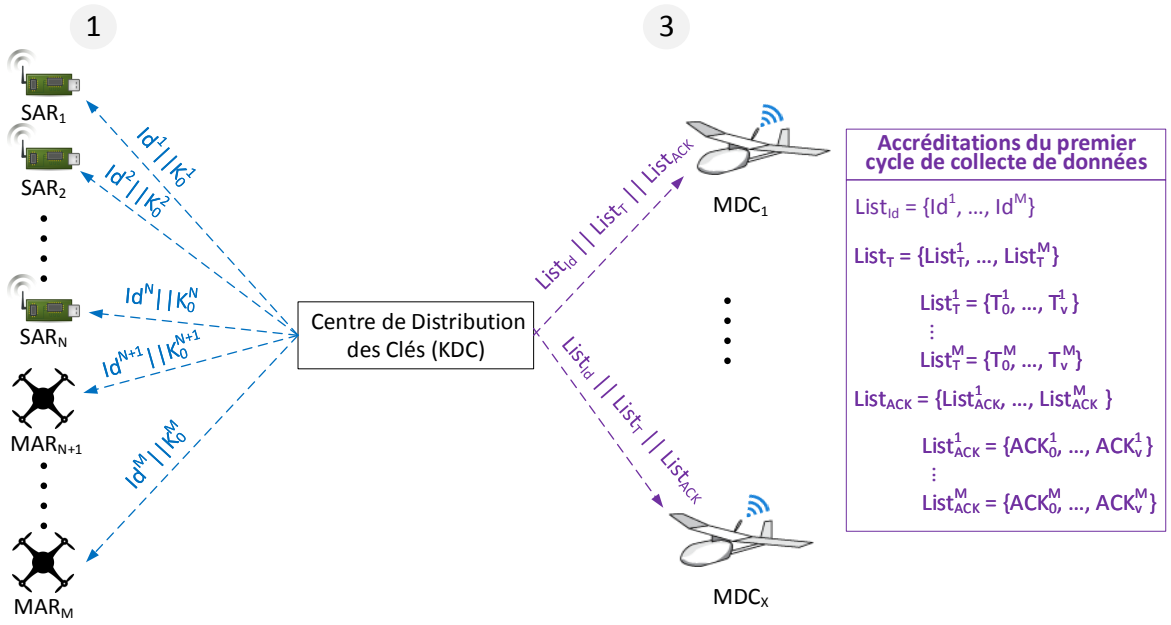


FIGURE 4.3 – Phase de pré-mission : enrôlement, distribution des clés et des accréditations

les données captées. Il est important de noter que dans la phase de mesure décrite dans la section 4.3.3.1, chaque clé secrète unique est effacée après le calcul de la clé suivante dans la chaîne. En outre, comme indiqué dans (ES3), une mémoire inviolable peut être utilisée pour une sécurité accrue concernant son stockage.

❸ Le MDC reçoit du KDC $List_{Id}$, la liste des identifiants des ARs de AR_1 jusqu'à AR_M pour lesquels il est autorisé à collecter des données. Il reçoit également les deux listes associées : $List_T$ et $List_{ACK}$ de ces ARs. Les paramètres M (nombre de nœuds ARs) et v (nombre de données captées par nœud AR qu'un MDC peut collecter) sont définis en fonction des exigences de la mission.

4.3.3 Phase de mission

Comme illustrée figure 4.2, la phase mission est divisée en deux sous phases : la phase mesure des données et la phase collecte des données. La phase de mesure a lieu avant, pendant ou après la phase de collecte des données.

4.3.3.1 Phase de mesure de données

La phase de mesure est divisée en trois étapes, comme nous le montrons en figure 4.4 :

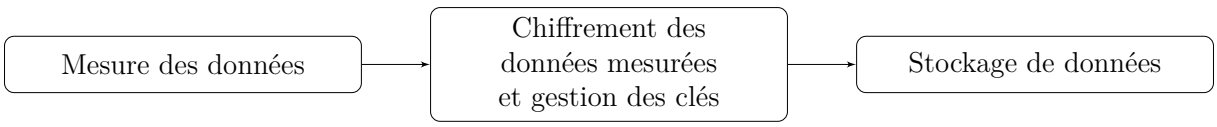


FIGURE 4.4 – Étapes lors des mesures de données

— ***Mesure des données :*** Les capteurs intégrés aux nœuds ARs effectuent les mesures requises.

— ***Chiffrement des données mesurées et gestion des clés :***

— AR_j chiffre immédiatement chaque élément des données mesurées SD_i^j à l'aide de la clé actuelle K_i^j : $[SD_i^j]^{K_i^j}$.

— Il calcule T_i^j, ACK_i^j comme suit :

$$T_i^j = H(K_i^j || \text{`1'})$$
$$ACK_i^j = H(K_i^j || \text{`2'})$$

— En utilisant ACK_i^j, AR_j calcule HM_i^j, le HMAC de la donnée chiffrée : $HM_i^j = HMAC_{ACK_i^j}([SD_i^j]^{K_i^j})$ et efface ACK_i^j.

— Ensuite, il calcule la prochaine clé $K_{i+1}^j = HMAC_{Id^j}(K_i^j)$ et efface la clé précédente K_i^j.

— ***Stockage de données :*** Ainsi, pour chaque élément de données mesurées SD_i^j, AR_j stocke dans sa mémoire les données chiffrées $[SD_i^j]^{K_i^j}$, T_i^j et HM_i^j comme illustré figure 5.1.

4.3.3.2 Phase de collecte de données

Les messages du protocole sont listés dans le tableau 4.2 et décrit ci-après.

Tableau 4.2 – Protocole de collecte de données sécurisée

1. MDC_l –> AR_j	:	$Request$
2. AR_j –> MDC_l	:	$Id^j \mid\mid [SD_i^j]^{K_i^j} \mid\mid HM_i^j$
3. MDC_l –> AR_j	:	T_i^j

— **Message 1 :** Lorsque les MDCs se déplacent vers la zone de mission, ils émettent un message *Request* pour réveiller les nœuds ARs déployés dans cette zone et initier la communication nécessaire à la collecte de données.

— **Message 2 :** Dès réception du *Request* de MDC_l, AR_j envoie son identité Id^j, les données chiffrées $[SD_i^j]^{K_i^j}$ et le $HMAC_{ACK_i^j}([SD_i^j]^{K_i^j})$ correspondant.

— **Message 3 :** Dès réception, MDC_l vérifie d'abord que l'Id^j reçu existe dans sa $List_{Id}$. Ensuite, il recherche l'ACK_i^j correspondant à l'Id^j reçu dans sa mémoire. Il calcule $HMAC_{ACK_i^j}$ sur les données chiffrées $[SD_i^j]^{K_i^j}$ reçues et il le compare à celui reçu, HM_i^j, pour s'assurer qu'elles proviennent réellement de AR_j. Si la comparaison aboutit, MDC_l envoie à AR_j le ticket T_i^j correspondant à son Id^j afin d'en accuser bonne réception. À ce stade, MDC_l est synchronisé avec AR_j et peut supprimer T_{i-1}^j et ACK_{i-1}^j s'ils existent. Toutefois, si la comparaison échoue, MDC_l tente de

valider le message 2 avec la clé précédente ACK_{i-1}^{j}. Si cela réussit, MDC_l envoie à AR_j le ticket T_{i-1}^{j}, et, s'il échoue à nouveau, le message reçu indique des données incorrectes (potentiellement envoyées par un attaquant). MDC_l redémarre alors avec le message 1 (*Request*) et/ou se déplace ailleurs. AR_j attend le ticket T_i^j qu'il compare ensuite à son T_i^j des dernières données chiffrées envoyées. Si la comparaison réussit, il supprime les données chiffrées de sa mémoire, ainsi que T_i^j et HM_i^j, et, de la même manière dans les messages précédents, il envoie les prochaines données chiffrées $[SD_{i+1}^{j}]^{K_{i+1}^{j}}$ jusqu'à ce qu'il ne reçoive plus de MDC_l de ticket associé ou qu'il ne lui reste plus de données à envoyer.

4.3.4 Phase de post-mission

La mission prend fin lorsque les MDCs ont collecté toutes les données captées pour lesquels ils ont les accréditations ou quand ils doivent rentrer à la station de base à cause d'un faible niveau d'énergie. À cette étape, les MDCs retournent à la BS et communiquent les données chiffrées collectées au sink. Le sink récupère alors auprès de KDC les paramètres nécessaires, c'est-à-dire les clés associées, pour déchiffrer les données collectées reçues.

4.4 Analyses de sécurité

Pour notre analyse de sécurité, nous fournissons à la fois une analyse informelle, en utilisant les modèles de menaces et d'attaques pertinents proposés dans la section 4.2.6, et une analyse formelle en utilisant les outils AVISPA et ProVerif.

4.4.1 Analyse informelle de sécurité

Nous nous référons aux exigences de sécurité par leurs numéros respectifs, comme décrit à la section 4.2.3.

— **Confidentialité (ES1), (ES3) :** La confidentialité est violée lorsqu'un adversaire peut accéder à des données secrètes. Dans notre protocole, la confidentialité des données captées SD_i^j est assurée par le chiffrement utilisant une clé secrète K_i^j. L'algorithme de chiffrement utilisé étant par définition sécurisé (voir l'hypothèse (H3) à la section 4.2.5), l'adversaire doit essayer de récupérer la clé. Cependant, un adversaire ne peut pas retrouver K_i^j car celle-ci est supprimée lors des opérations de mesures des données, et, seule la station de base dispose de la clé K_0^j initiale qui permet de calculer K_i^j. Même en capturant un nœud AR ou un MDC, l'attaquant ne peut pas calculer le K_i^j précédent utilisé à partir des données présentes (c'est-à-dire pour un i choisi, T_i^j, ACK_i^j, HM_i^j ou $[SD_i^j]^{K_i^j}$) puisque les fonctions de hachage et HMAC utilisées sont supposées être robustes (voir l'hypothèse (H2) à la section 4.2.5). Comme indiqué ci-dessous concernant la « Perfect Forward Secrecy », l'attaquant ne peut pas utiliser K_i^j dans cet objectif. En d'autres termes, casser la confidentialité des données captées est similaire à casser l'algorithme de chiffrement.

— **Intégrité et fraîcheur des données (ES2) :** Le protocole proposé peut résister aux menaces pesant sur l'intégrité. Le MDC peut garantir que le message 2 n'a pas été modifié par un adversaire en calculant $HMAC_{ACK_i^j}([SD_i^j]^{K_i^j})$ et en vérifiant qu'il est égal au HM_i^j reçu.

— **« Perfect Forward Secrecy » (ES4) :** La « Perfect Forward Secrecy » signifie que si une clé secrète K_i^j est compromise, cela ne permettra pas à l'adversaire de compromettre les clés générées précédemment. Le protocole proposé utilise la fonction HMAC pour calculer la clé de chiffrement suivante $K_i^j = HMAC_{Id^j}(K_{i-1}^j)$

et, sous l'hypothèse (H2) de la section 4.2.5, la fonction HMAC est considérée comme robuste, c'est-à-dire qu'un adversaire ne peut pas calculer K_{i-1}^j à partir de K_i^j. Par conséquent, notre protocole fournit la protection des transmissions passées.

— **Authentification mutuelle (ES5) :** Le protocole proposé permet une authentification mutuelle, les nœuds MDCs et ARs s'authentifiant mutuellement. Plus spécifiquement, lorsqu'un MDC reçoit le message 2, il calcule le HMAC sur $[SD_i^j]^{K_i^j}$ en utilisant l'ACK_i^j correspondant à l'Id^j reçu et fourni par la station de base lors de la phase de pré-mission et il le compare au HM_i^j qu'il a reçu. En cas d'égalité, le MDC authentifie le nœud AR et envoie un ticket T_i^j pour accuser réception de $[SD_i^j]^{K_i^j}$ et demander la transmission des données chiffrées suivantes $[SD_{i+1}^j]^{K_{i+1}^j}$. Lorsque le nœud AR reçoit le message 3, il compare le T_i^j reçu au T_i^j stocké dans sa mémoire pour les dernières données chiffrées envoyées $[SD_i^j]^{K_i^j}$. Si la comparaison réussit, le nœud AR authentifie le MDC et envoie les données cryptées suivantes $[SD_{i+1}^j]^{K_{i+1}^j}$.

Jeu 1 - α ***se fait passer pour un MDC*** : α tente d'envoyer les messages légitimes 1 et 3. Autrement dit, α peut soit casser la clé K_i^j, soit générer directement un message 3 valide basé sur les messages 1, 2 et 3 interceptés des sessions précédentes.

— Pour casser K_i^j, α essaiera d'extraire du message 2 la valeur de K_i^j en utilisant des valeurs publiques. Cela n'est pas possible en raison de l'hypothèse (H2) de la section 4.2.5 indiquant que la fonction HMAC est robuste et non réversible.

— α peut combiner les messages 1, 2 et 3 afin de déduire des informations précieuses qu'il pourrait utiliser pour casser la clé K_i^j. Cependant, les messages 2 et 3 évoluent indépendamment l'un de l'autre car leurs entrées sont différentes. Ainsi, il est impossible d'extraire une information intéressante et ce quel que soit le nombre de messages 1, 2 et 3 interceptés ; le jeu ne peut donc pas réussir.

Jeu 2 - α ***crée une contrefaçon*** AR_x : En se référant au *Jeu 2* dans la section 4.2.6, si α compromet physiquement un AR_j, il pourra accéder à tout ce qu'il contient, y compris les informations secrètes actuelles et les informations échangées avec le MDC. Pour créer un nœud AR_x contrefait et tromper un MDC, α doit connaître l'identité Id^x de AR_x. Hors comme l'identité de chaque nœud AR est différente et unique, α ne peut pas deviner l'identité de AR_x légitime en connaissant l'identité Id^j de AR_j. Ainsi, il est impossible pour α de dériver d'autres nœuds ARs de $List_{Id}$ en compromettant un AR_j. Par conséquent, α ne peut pas gagner le jeu. De plus, s'il réussissait à trouver un Id^x existant, il ne trouverait pas le K_i^x associé requis pour calculer avec succès T_i^x, ACK_i^x, HM_i^x, etc.

Le protocole proposé résiste également aux attaques suivantes :

— *Attaque de l'homme du milieu (MiTM)* : Cette attaque est possible contre notre protocole, mais elle ne peut être utile qu'aux parties en communication légitimes, car l'attaquant ne peut agir qu'en tant que répéteur en élargissant ses plages de communication. En effet, l'attaquant ne peut transférer des messages entre les MDCs et les ARs que : (a) s'il transmet le *Request* (message 1) reçu d'un MDC pour obtenir le message 2 d'un AR et envoie un message modifié au MDC ; ce dernier le rejettera puisque le HMAC sera faux ; et (b) si il laisse le message de l'AR atteindre le MDC et intercepte le ticket (message 3) pour le modifier, alors l'AR le rejettera car il ne correspond pas au ticket attendu. Ainsi, notre protocole résiste à une attaque de type MiTM.

— *Attaque de désynchronisation* : Notre protocole résiste à une telle attaque.

  — Du côté AR, un AR_j envoie toujours le même message 2 (Id^j || $[SD_i^j]^{K_i^j}$ || HM_i^j) à la réception de chaque *Request* (ou T_{i-1}^j si des échanges réussis ont déjà eu lieu) jusqu'à ce qu'il reçoive le bon ticket (T_i^j), qui est la preuve de la réception des données par le MDC.

— Du côté du MDC, avant de supprimer le ticket précédent T_{i-1}^j et le ACK_{i-1}^j, un MDC attend toujours de recevoir et de valider le message 2 suivant de AR_j (c'est-à-dire $Id^j \mid\mid [SD_i^j]^{K_i^j} \mid\mid HM_i^j$, qui est une preuve implicite de la réception du ticket précédent T_{i-1}^j envoyé par MDC à AR_j). Si le MDC le valide avec ACK_i^j, il supprime T_{i-1}^j et ACK_{i-1}^j. Si le MDC ne peut pas valider le message 2, il tente de le valider avec ACK_{i-1}^j car une désynchronisation est peut-être survenue. S'il réussit à le valider avec ACK_{i-1}^j, il envoie au AR_j le T_{i-1}^j et attend à nouveau de recevoir le prochain message 2 valide de AR_j comme expliqué avant. Si la validation échoue même avec ACK_{i-1}^j, cela indique un faux message (de la part d'un attaquant) et, comme cela a été expliqué, il attend à nouveau de recevoir le prochain message 2 valide de AR_j.

Un attaquant pourrait essayer de désynchroniser MDC et AR_j en supprimant le ticket précédent T_{i-1}^j envoyé par le MDC, puis en envoyant un *Request* à AR_j, un MDC légitime pourrait recevoir à nouveau le message 2 reçu précédemment (c'est-à-dire $Id^j \mid\mid [SD_{i-1}^j]^{K_{i-1}^j} \mid\mid HM_{i-1}^j$) puisque le AR_j n'a pas reçu le T_{i-1}^j supprimé. Ainsi, si le MDC légitime avait supprimé (T_{i-1}^j et ACK_{i-1}^j), il ne pourrait pas se resynchroniser.

Cependant, grâce aux deux mécanismes mentionnés précédemment côté AR et coté MDC, notre protocole résiste aux attaques de désynchronisation.

— *Attaque par déni de service* : Comme tout protocole dans un réseau sans fil, notre protocole est sensible aux attaques par brouillage. Cependant, la vérification HMAC effectuée dans le MDC à la réception du message 2 a été ajoutée afin de se protéger contre une attaque par déni de service d'un attaquant tentant de saturer la mémoire de fausses données (son objectif étant d'éviter que le MDC mène à bien sa mission de collecte de données). Sur l'AR, la vérification du ticket (ce qui est très simple et

efficace à réaliser) permet également de contrer une attaque de type DoS d'un attaquant essayant d'effacer les données captées de sa mémoire. Ainsi, notre proposition comprend des protections contre plusieurs attaques par déni de service.

4.4.2 Vérification formelle du protocole

Pour vérifier l'exactitude de notre protocole, nous avons sélectionné deux outils AVISPA et ProVerif présentés lors du chapitre 3. Ces outils prennent en charge le modèle d'attaque Dolev-Yao [160] dans lequel l'adversaire a le contrôle total sur le canal de communication : c'est-à-dire qu'il peut entendre, intercepter, supprimer et injecter n'importe quel message.

4.4.2.1 Vérification avec l'outil AVISPA

Comme les spécifications et les résultats sont disponibles dans l'annexe A.1, nous ne présentons dans cette section que les informations les plus importantes.

Spécification HLPSL du protocole

Le script de notre spécification est présenté dans l'annexe A.1.1.

— Il y a deux agents `MDC` et `ARJ`, et ils utilisent tous les deux une fonction de hachage `Hk` et une clé secrète `Acki` pour calculer le `Hmac` des données captées `SDi`.

— La confidentialité des données captées `SDi` est modélisée à l'aide du prédicat `secret (SDi', sec_k, {ARJ, MDC)}`, qui est géré par l'identifiant de protocole `sec_k`. Le paramètre `SDi` est gardé secret pour `ARJ` et `MDC`.

— L'authentification mutuelle est réalisée via `witness` et `request`, c'est-à-dire `witness (ARJ, MDC, auth_n, Hmac)`, `request (MDC, ARJ, auth_n, Hmac')`, `witness (MDC, ARJ, auth _d, Ti)`, `request (ARJ, MDC, auth_d, Ti')`.
`witness (ARJ, MDC, auth_n, Hmac')` déclare que l'agent `ARJ` prétend être l'homologue de l'agent `MDC`, en s'accordant sur la valeur `Hmac'`. `auth_n` est le nom de

l'authentification `Hmac'` indiquée dans la section `goals` alors que `request (MDC, ARJ, auth_n, Hmac')` déclare que l'agent `MDC` accepte la valeur `Hmac'` et se repose maintenant sur la garantie que l'agent `ARJ` existe et est d'accord sur cette valeur.

— L'environnement contient les constantes globales et la composition d'une ou plusieurs sessions. L'intrus participe en tant que session concrète à l'exécution du protocole.

Résultat de la vérification

Les résultats de la vérification des back-ends OFMC et CL-AtSe sont disponibles dans l'annexe A.1.2. Ils montrent que notre protocole est sûr, c'est-à-dire sécurisé, contre les attaques actives et passives.

4.4.2.2 Vérification avec l'outil ProVerif

Comme les spécifications et les résultats sont disponibles dans l'annexe A.2, nous ne présentons dans cette section que les informations les plus importantes.

Spécification pi calculus appliqué du protocole

Les scripts pi calcul appliqué sont présentés dans l'annexe A.2.1.

— Les clés secrètes `sk`, `acki` et les données captées `SDI` sont déclarées secrètes pour l'attaquant à l'aide du mot `[private]`. `ch` est le canal public où MDC et ARJ échangent leurs messages et `hk` est une fonction de hachage utilisée pour calculer HMAC.

— La confidentialité des données captées `SDI` est vérifiée avec `query attacker (SDI)`.

— L'authentification mutuelle entre MDC et ARJ est modélisée avec la définition de quatre événements mappés dans les sous-processus de MDC et ARJ et les requêtes suivantes :

```
event beginARJ(bitstring).
event endARJ(ticket).
```

```
event beginMDC(bitstring).
event endMDC(ticket).
query x: bitstring; inj-event(beginMDC(x)) ==> inj-event(beginARJ(x)).
query x: ticket; inj-event(endARJ(x)) ==> inj-event(endMDC(x)).
query x: bitstring, y:ticket; inj-event(endARJ(y)) ==> inj-event(beginARJ(x)) && inj-event(beginMDC(x)).
```

— Dans le processus principal, les sous-processus MDC et ARJ s'exécutent en parallèle. `!` indique un nombre illimité de processus :

```
process((!MDC(ID, TI, acki)) | (!ARJ(ID, HMAC, TI, SDI)))
```

Résultat de la vérification Les résultats disponibles dans l'annexe A.2.2 montrent que le protocole préserve le confidentialité des données captées (`SDI`) et que l'authentification mutuelle entre MDC et ARJ est réussie.

Conclusion

Dans ce chapitre, nous avons décrit de manière détaillée le protocole de sécurité que nous avons proposé pour la collecte des données. Ce dernier est un protocole sans serveur (serverless) qui permet d'assurer la confidentialité des données captées par des nœuds RFID actifs (ARs) et collectées par des MDCs, de bout en bout depuis leur mesure par les nœuds RFID capteurs jusqu'à leur délivrance à la station de base. Ce protocole assure également une authentification mutuelle entre les nœuds ARs et les MDCs et il garantit l'intégrité des données envoyées pendant la phase de collecte ainsi que la « Perfect Forward Secrecy ». Pour tenir compte des ressources limitées des ARs et des MDCs, nous avons utilisé des opérations cryptographiques légères telles que les fonctions de hachage et HMAC.

Pour une meilleure présentation de cette solution, nous avons tout d'abord décrit un modèle du système de collecte de données permettant de définir les exigences en termes

de sécurité et de performance et les modèles de menaces et d'attaques. Enfin, nous avons fourni une analyse de sécurité informelle ainsi qu'une validation avec des jeux de sécurité pour démontrer que notre proposition est sécurisée contre diverses attaques. Deux vérifications formelles du protocole proposé avec les outils AVISPA et ProVerif ont également été effectuées et ont montré que le protocole est sûr et répond aux exigences de sécurité exigées préalablement.

L'évaluation des performances de notre protocole a été réalisée en se basant sur les résultats d'une implémentation sur des dispositifs très limités en ressources afin de monter qu'il satisfait aussi les exigences de performance. Le chapitre suivant présente les détails de cette évaluation.

Chapitre 5

Implémentation et évaluation de performances

Introduction

Comme nous avons montré au chapitre 4 que le protocole proposé satisfait aux exigences de sécurité souhaitées, il faut maintenant analyser ses performances et c'est ce à quoi nous nous attachons dans ce chapitre. Ainsi, nous analysons les performances de notre proposition pour chacune des entités impliquées dans le système de collecte de données en termes de surcharge d'espace mémoire et de temps de calcul. Ensuite, nous illustrons en détails l'interaction des nœuds ARs et du MDC ainsi que l'évolution de leurs mémoires au cours de la phase de mission (phase de mesure et de collecte). Enfin, nous montrons que le protocole proposé satisfait aux exigences de performance à la fois avec une analyse informelle et avec une évaluation basée sur les résultats expérimentaux obtenus sur des périphériques extrêmement contraints (c'est-à-dire, des cartes à puce).

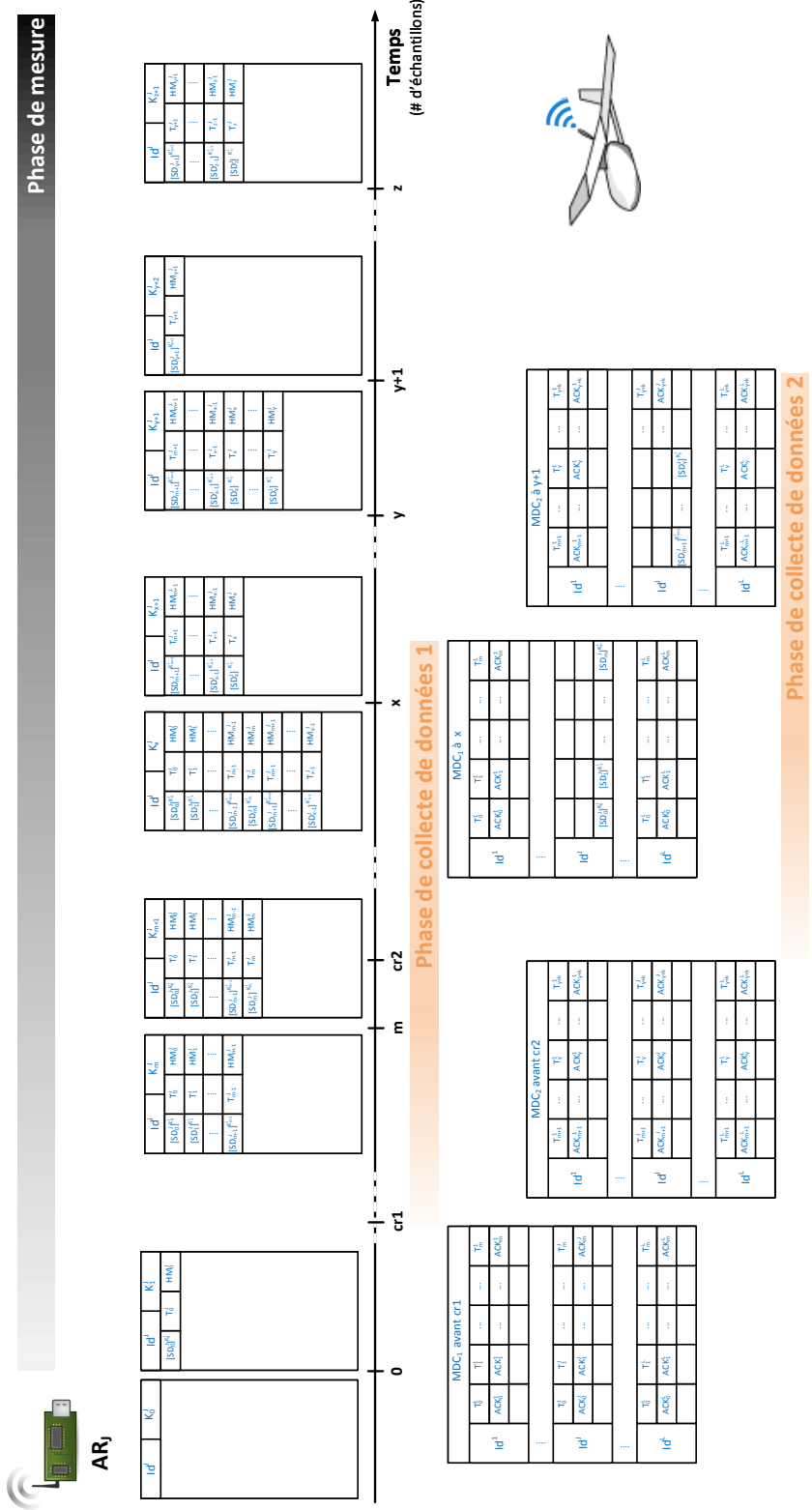


FIGURE 5.1 – Évolution des mémoires des ARs et MDCs durant la phase de mission

5.1 Scénario illustratif de l'évolution de la mémoire des participants au cours des différentes phases

Nous considérons un exemple où deux MDCs collectent des données à partir de AR_j. MDC_1 est autorisé à collecter uniquement les m premières données mesurées ($[SD_0^j]^{K_0^j}$, ..., $[SD_m^j]^{K_m^j}$) tandis que MDC_2 a la permission de collecter les données ($[SD_{m+1}^j]^{K_{m+1}^j}$, ..., $[SD_{y+k}^j]^{K_{y+k}^j}$) comme illustré figure 5.1.

À Temps < 0, c'est-à-dire avant le déploiement sur le terrain pour la mission, AR_j n'a que ses paramètres initiaux (c'est-à-dire Id^j, K_0^j). Lorsque Temps $= 0$, la phase de mesure commence et AR_j capte ses premières données, les chiffre, calcule T_0^j et ACK_0^j pour calculer HM_0^j, puis stocke $[SD_0^j]^{K_0^j}$, T_0^j et HM_0^j dans sa mémoire et poursuit les mêmes opérations jusqu'à la fin de cette phase (la fin de la phase de mesure, définie en fonction des paramètres de la mission).

— [cr1, x] : intervalle de temps pour la phase de collecte de données de MDC_1
  — Temps $<$ cr1 [1], MDC_1 n'a que ses paramètres initiaux (c'est-à-dire les accréditations), sans donnée captée dans sa mémoire.
  — À Temps = cr1, MDC_1 a commencé sa mission de collecte de données mais n'a pas encore communiqué avec AR_j.
  — À Temps = m, bien que AR_j ait capté toutes les données que MDC_1 a le droit de collecter, le MDC n'a pas encore commencé la collecte de données de AR_j (par exemple, ils ne se trouvent peut-être pas à portée de communication).
  — À Temps = cr2, MDC_1 n'a pas encore commencé la collecte de données à partir de AR_j, et pourtant un second MDC, MDC_2, est envoyé.

1. On notera que cr représente la demande de collecte (« collection request »), c'est-à-dire le début des activités du MDC dans la zone de la mission pour commencer la collecte de données. Le nombre concaténé après cr correspond au numéro du MDC.

- À différents moments ∈]cr2, x[, MDC_1 a communiqué avec AR_j pour commencer la collecte de données.
- À Temps = x, MDC_1 a collecté toutes les données qu'il est autorisé à collecter (c'est-à-dire les données captées avant Temps = m) et il retourne à la BS. Il est à noter que AR_j a continué à capter des données pendant la phase de collecte (ce n'est pas une phase bloquante). La section 5.3.5 contient des estimations réalistes sur le temps nécessaire pour la collecte de données et sur la fréquence de mesures des données qui illustrent clairement que les ARs doivent et peuvent continuer à capter des données.

— [cr2, y+k] : intervalle de temps pour la phase de collecte de données de MDC_2

- À Temps < cr2, comme pour MDC_1, MDC_2 n'a que ses paramètres initiaux (accréditations permettant de collecter des données de m+1 à y+k) sans donnée captée dans sa mémoire.
- À Temps = cr2, MDC_2 a commencé sa mission de collecte de données mais n'a pas encore communiqué avec AR_j.
- À Temps = x, MDC_2 n'a toujours pas commencé la collecte de données de AR_j.
- À différents moments ∈]x, y+1[, MDC_2 a communiqué avec AR_j pour commencer la collecte de données.
- À Temps = y+1, MDC_2 a collecté toutes les données qu'il était autorisé à collecter et qui étaient disponibles sur AR_j (c'est-à-dire que $[SD^j_{m+1}]^{K^j_{m+1}}$ à $[SD^j_y]^{K^j_y}$). Au lieu de revenir à la BS, MDC_2 se déplacera vers d'autres ARs et reviendra plus tard car il lui reste k accréditations pour AR_j.
- À Temps = z, si MDC_2 n'est pas encore revenu pour collecter les données restantes de AR_j pour lesquelles il possède encore des accréditations (données

captées dans [y+1, y+k]). Il reviendra plus tard ou s'il doit revenir à la BS en raison d'un faible niveau d'énergie, un autre MDC sera lancé par la BS pour démarrer une phase de collecte afin d'obtenir ces données.

Comme illustré figure 5.1, AR_j supprime les données captées après chaque collecte de données afin de libèrer de l'espace mémoire pour les futures données à capter.

5.2 Analyses informelles de performances

Les exigences définies dans la section ont été atteintes puisque :

— un nœud AR, AR_j, nécessite seulement – par données captées (SD_i^j) – :

  — 1 exécution d'un chiffrement sur des données captées, $[SD_i^j]^{K_i^j}$, afin de garantir leur confidentialité ;

  — 2 exécutions sur des fonctions de hachage pour calculer le ticket associé T_i^j pour l'authentification du MDC et la clé ACK_i^j (utilisée comme clé pour calculer HM_i^j) ;

  — 1 exécution de la fonction HMAC pour calculer le HM_i^j susmentionné afin de garantir l'intégrité des données envoyées lors de la phase de collecte ;

  — 1 exécution supplémentaire de la fonction HMAC pour calculer la prochaine clé unique, K_{i+1}^j.

— un MDC, MDC_l, ne doit effectuer qu'une exécution de la fonction HMAC par donnée captée chiffrée reçue, $[SD_i^j]^{K_i^j}$, à l'aide de la clé appropriée ACK_i^j (trouvée dans les informations d'accréditations qu'il a reçues de la BS pour le AR_j) afin de vérifier l'intégrité des données collectées et d'obtenir une authentification implicite de AR_j.

Pour les MDCs, la plupart des calculs ont été effectués par le KDC dans la phase de pré-mission, ce qui leur permet de consacrer la plus grande partie de leur temps et de leur énergie à se rapprocher des nœuds ARs pour collecter efficacement des données, au lieu de passer du temps et de consommer de l'énergie à effectuer des calculs cryptographiques.

5.3 Implémentation du protocole, évaluation des performances et discussion des résultats

Au lieu de choisir arbitrairement des périphériques ARs particuliers, pour évaluer l'efficacité de notre proposition, nous avons décidé de procéder à des mesures sur les périphériques les plus fortement limités en ressources, à savoir les cartes à puce, afin de nous assurer que notre proposition convient aux nœuds ARs disposant de ressources supplémentaires. Ce travail permet de disposer d'une base solide pour extrapoler les résultats escomptés en effectuant des opérations en utilisant d'autres types d'ARs. Pour donner un ordre d'idées, les ressources contraintes actuellement disponibles sur une carte sont :

a) pour les mémoires de l'ordre de 1 à 2 ko de RAM, de 32 à 64 ko de mémoire non volatile (EEPROM ou Flash) et de 128 ko pour la ROM ;

b) pour le processeur, une architecture 16-bits et une fréquence interne d'une trentaine de MHz.

5.3.1 Choix des périphériques cibles

Pour implémenter notre protocole et mesurer ses performances, parmi les différentes technologies disponibles pour cartes à puce, comme illustré figure 5.2, nous avons choisi la plate-forme Java Card [191] pour laquelle nous avons développé sous l'environnement

Eclipse disposant du plugin JCOP développé par IBM Zurich pour permettre un développement et un déploiement plus rapide. Le choix de cette technologie est plutôt pertinent puisque les décisions récentes de Oracle sont de viser pour cette technologie les périphériques IoT [192] avec la publication de la version 3.1 des spécifications [193, 194], ce qui est totalement compatible avec les Active RFID qui sont l'objet de nos travaux.

FIGURE 5.2 – La technologie et l'outil de développement choisis

5.3.1.1 Présentation de la technologie Java Card

La technologie Java Card est une technologie qui permet de programmer en Java des périphériques à mémoire limitée (essentiellement des cartes à puce). Cette technologie a été lancée en 1996 par Sun Microsystems et depuis le rachat de ce dernier, elle est maintenant détenue par Oracle Inc. qui vend des licences à plusieurs fabricants. L'intérêt majeur de cette technologie est, en plus de tous les avantages liés à la programmation orientée objets et à la portabilité liée à Java (« write once, run anywhere »), de cacher la complexité sous-jacente des contraintes des cartes à puce (persistance mémoire et gestion l'atomicité et des transactions, protocoles de communication bas-niveau, etc.). Cette technologie offre aussi la multiapplication, c'est-à-dire la possibilité d'héberger plusieurs applications, nommées applets dans le jargon Java Card, sur le même périphérique. Dans toutes les versions commercialisées actuellement, le multithreading n'étant pas inclus, les applications s'exécutent à tour de rôle. Seules les versions des spécifications Java Card 3.x Connected Edition proposent le multithreading, mais à notre connaissance aucune carte

les supportant n'a été proposée sur le marché.

Comme il est possible de le constater figure 5.3, l'architecture de la technologie Java Card est plutôt simple. Au dessus du hardware (microcontrolleur, c'est-à-dire microprocesseurs, mémoires et partie analogiques) et du système natif de la carte se trouve la première brique, la JCVM (Java Card Virtual Machine). Ce n'est pas exactement toute la JCVM, mais essentiellement l'interpréteur de bytecode. Comme Java Card n'est pas l'objet de la thèse, nous ne rentrerons pas plus dans les détails concernant le JCVM. En tout cas, cette brique peut être vue comme un processeur virtuel du point de vue des applications (applets) et quelque soit les cartes sur lesquelles elles sont installées si la JCVM est présente elles pourront fonctionner. Le fabricant d'une Java Card est donc essentiellement un implémenteur d'une JCVM pour une architecture et une puce donnée. Le second composant qu'utilise les applets sont les APIs Java Card. C'est au travers de ces APIs standard que les applications peuvent accèder à différents services présents sur la carte, notamment les services cryptographiques. Les APIs Java Card peuvent être vues comme une bibliothèque standard d'un système (par exemple la `libc` sous les systèmes Unix). Le dernier composant est le JCRE (Java Card Runtime Environment) qui est un peu le système d'exploitation virtuel du point de vue des applets. C'est en effet lui qui gère différents services systèmes (communications, gestion de la persistance, etc.). Sur les cartes actuelles, il est en étroite interaction avec le Card Manager introduit GlobalPlatform [2] pour la gestion des applets et en particulier leur installation (chargement) et que toutes les Java Cards supportent aujourd'hui.

Pour réaliser l'implémentation et l'évaluation de performances de notre protocole, il nous faut donc simplement développer une application Java Card et la charger sur une carte comme nous le présenterons en section 5.3.2.

2. GlobalPlatform est un consortium qui produit une série de spécifications visant à permettre la gestion de contenu de périphériques embarqués indépendemment de la technologie qu'ils utilisent).

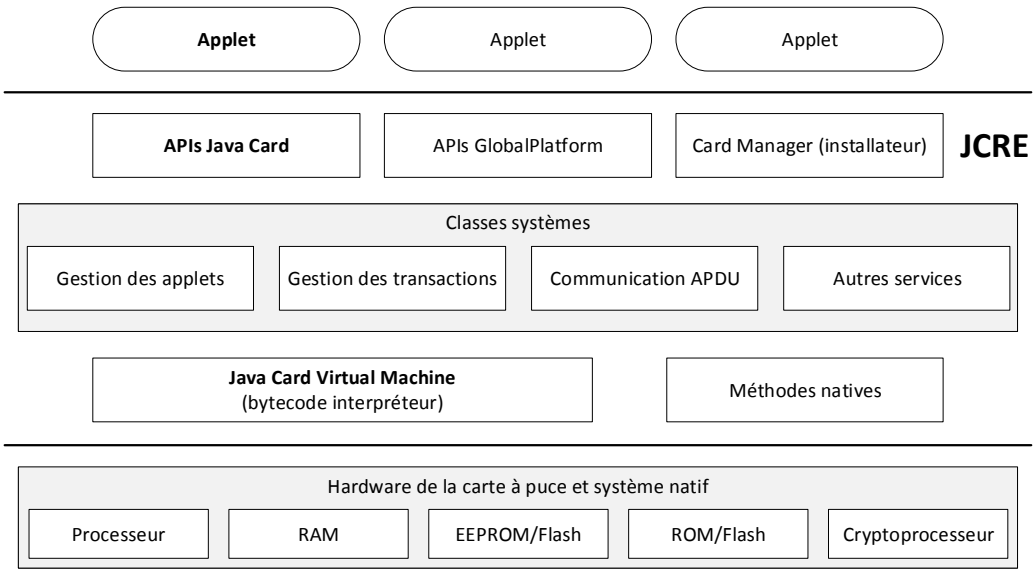


FIGURE 5.3 – Architecture de la technologie Java Card

5.3.1.2 Différents modèles de cartes testées

Nous avons testé les performances de l'implémentation sur douze cartes différentes provenant de quatre fabricants différents, produites à des dates différentes. S'il est vrai que certaines cartes Java utilisent un crypto-processeur pour accélérer les opérations de chiffrement, cela pourrait également être le cas avec certains des ARs considérés. Mais, cet avantage est contrebalancé par le fait que nous avons décidé de mener nos tests de performance sur des algorithmes cryptographiques lourds (AES-128 en mode CBC sans remplissage, « padding » pour l'algorithme de chiffrement, AES-128 MAC pour la fonction HMAC, et SHA-256 et SHA-1 pour la fonction de hachage) qui pourraient être plus légers dans le cas d'ARs. Comme le montrent les résultats de nos tests de performance, seules les cartes très récentes semblent utiliser une implémentation matérielle (hardware) de l'AES.

Le tableau 5.1 résume certaines caractéristiques des cartes testées. Comme les fabricants 'J', 'F', 'N' et 'U', qui n'ont pas de signification particulière.

Tableau 5.1 – Caractéristiques des cartes testées

Fabriquant	J		F			N						U
Modèle de carte	J1	J2	F1	F2	F3	N1	N2	N3	N4	N5	N6	U1
Architecture (bits)	16	16	16	32	32	16	16	16	16	16	16	16
Date estimée de production	2017	2017	2015	2014	2014	2004	2008	2006	2004	2002	2002	2009
Support AES-128 CBC NO PAD	✓	✓	✓	✓	❋	✓	✓	✓	✓	✓	✓	✓
Support AES-128 MAC	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Support SHA-1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Support SHA-256	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗

✓ : Supporté, ✗ : Non supporté, ❋ : Partiellement supporté.

5.3.1.3 Communication avec une Java Card

Pour communiquer avec une Java Card, il faut procéder exactement comme avec une autre carte. Il est nécessaire de disposer d'un lecteur de carte comme par exemple celui présenté figure 5.4. Il s'agit d'un lecteur à contact, car pour des raisons de simplicité mais aussi en raison du plus grand nombres de produits disponibles, nous avons fait nos mesures sur des cartes compatibles ISO7816 en utilisant l'interface contact et non celle contactless utilisant les protocoles de communication sans fil de l'ISO14443.

Figure 5.4 – Lecteur IDBridge CT30 (Gemalto)

Le lecteur que nous avons choisi est compatible CCID, un protocole standard pour la communication entre la machine et le lecteur et qui évite l'installation d'un pilote sous Windows puisque le standard est supporté nativement.

Pour communiquer, une carte à puce et un lecteur utilisent des TPDU (Transmission Protocol Data Unit), des protocoles de bas niveau dont nous souhaitons nous abstraire qui sont spécifiés dans l'ISO7816-3 (T=0, T=1). Pour communiquer nous utiliserons donc une application sur PC capable de se connecter avec le middleware compatible PC/SC [3] qui gère les connexions entre les applications utilisant les lecteurs de cartes à puce et ces derniers. Les messages qui seront échangés entre l'application utilisée coté PC et celle présente sur la carte sont les APDU (Application Protocol Data Unit), spécifié dans l'ISO7816-4. Le mode de communication entre un lecteur et une carte est un mode client-serveur, dans lequel la carte est le serveur. Elle attend donc des commandes pour fournir des réponses. Les commandes APDU envoyées par l'application transitent donc jusqu'au middleware PC/SC puis celles-ci sont transportées entre le PC et le lecteur via le protocole CCID ; à cette étape, le lecteur utilise les TPDU pour transporter et fournir la commande APDU à la carte via l'interface contact et attend de recevoir de la carte une réponse APDU sur cette même interface contact qu'il transmettra ensuite via CCID jusqu'au PC où elle sera redirigée vers l'application d'origine.

La communication entre une application sur le PC et une carte est donc un processus assez complexe pouvant engendrer une surcharge qu'il faudra prendre en compte lors de nos mesures de performance.

Pour des raisons de simplicité, comme illustré figure 5.5 et figure 5.8, nous utiliserons l'outil de communication présent dans le plug-in JCOP (le "Shell") pour dialoguer avec notre applet présente sur la carte.

3. Sous Windows, ce middleware se nomme PC/SC et sous Linux, PC/SC Lite.

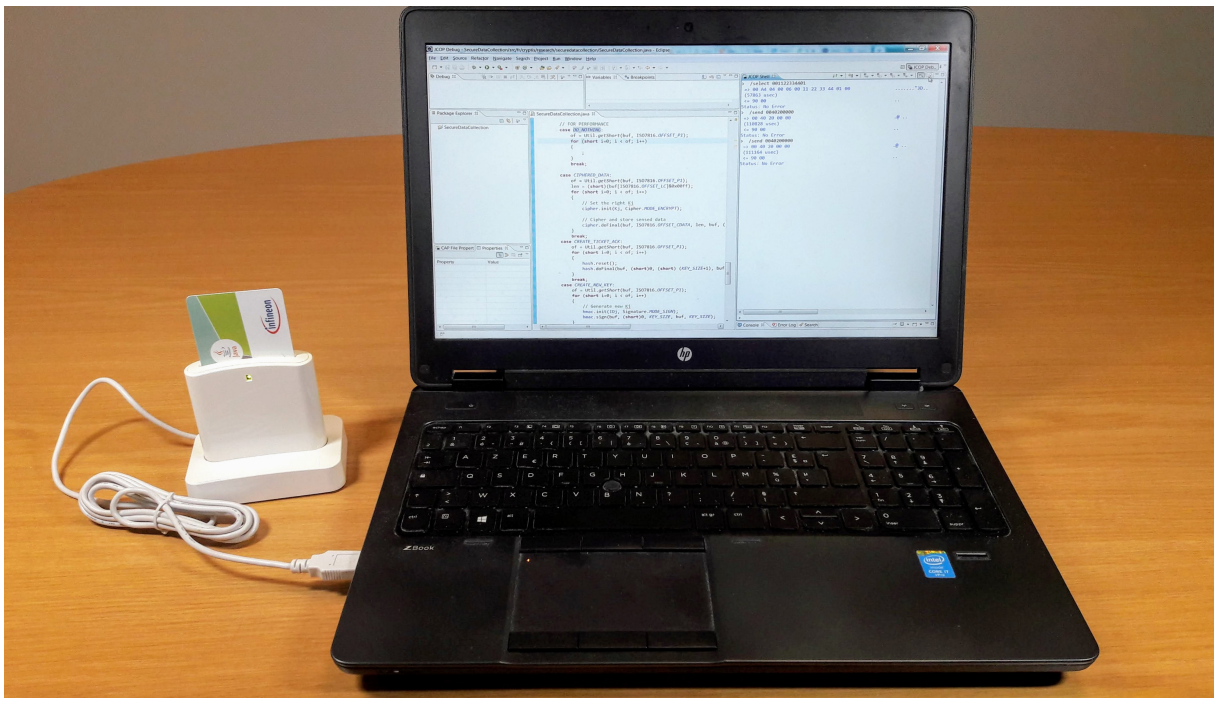


FIGURE 5.5 – Banc de test pour les mesures simples

5.3.2 Choix des outils de développement

Afin d'écrire et de déployer notre applet sur les Java Card, nous avons choisi d'utiliser l'environnement Eclipse disposant du plugin JCOP développé par IBM Zurich. En effet, celui-ci permet un développement et un déploiement rapide (une alternative était d'utiliser le JCDK (Java Card Development Kit) [195] et un outil compatible GlobalPlatform pour charger notre applet sur la Java Card).

La vue de développement Java Card est très simple comme on le constate figure 5.6. Après avoir créé un projet Java Card en spécifiant l'identifiant qu'aura notre application (AID pour « Application IDentifier ») sur la carte `001122334401` et la version des bibliothèques (APIs) à utiliser (pour nous Java Card 2.2.1), il ne reste plus qu'à écrire le code des différentes opérations. La totalité du code de l'applet permettant de simuler le protocole côté AR est disponible à l'annexe C.1 (un exemple d'exécution est disponible

à l'annexe C.2). Comme nous intéressons essentiellement aux performances, nous avons également écrit dans notre applet le code pour mesurer les performances des opérations présentées en section 5.3.3.

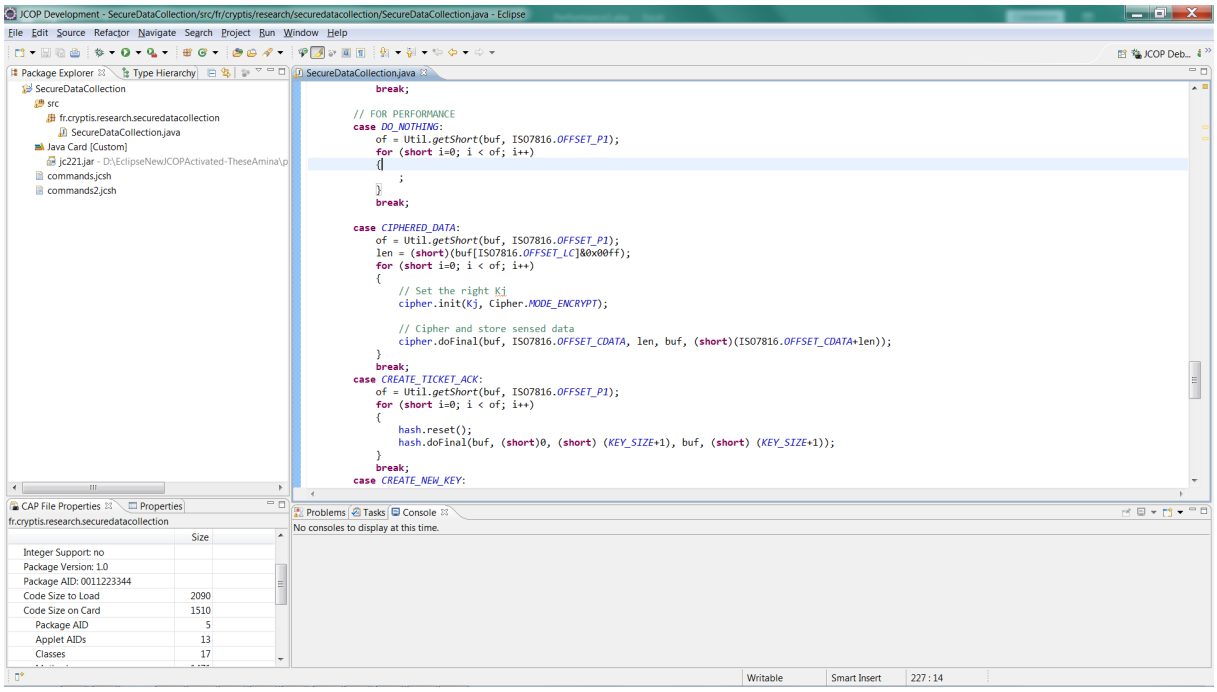


FIGURE 5.6 – Vue de développement sous Eclipse avec le plug-in JCOP

Une fois le code finalisé, il est possible de créer simplement une configuration d'exécution ou de debug afin de déployer et tester l'applet soit dans un émulateur proposé par le plug-in JCOP, soit sur une vraie carte comme nous le faisons figure 5.7.

Une fois l'applet chargée, pour la tester, comme illustré figure 5.8, il suffit d'utiliser le cadre de droite qui s'appelle le "Shell" afin d'interagir avec la carte en lui envoyant des commandes APDU et recevant ses réponses APDU. On notera que le "Shell" fournit le temps écoulé entre l'envoi de la commande et la réception de la réponse. Ce "Shell" est en fait ni plus ni moins qu'une application connectée via PC/SC au lecteur souhaité et donc à la carte.

Sur une carte multiapplicative, avant de pouvoir dialoguer avec son application, il est

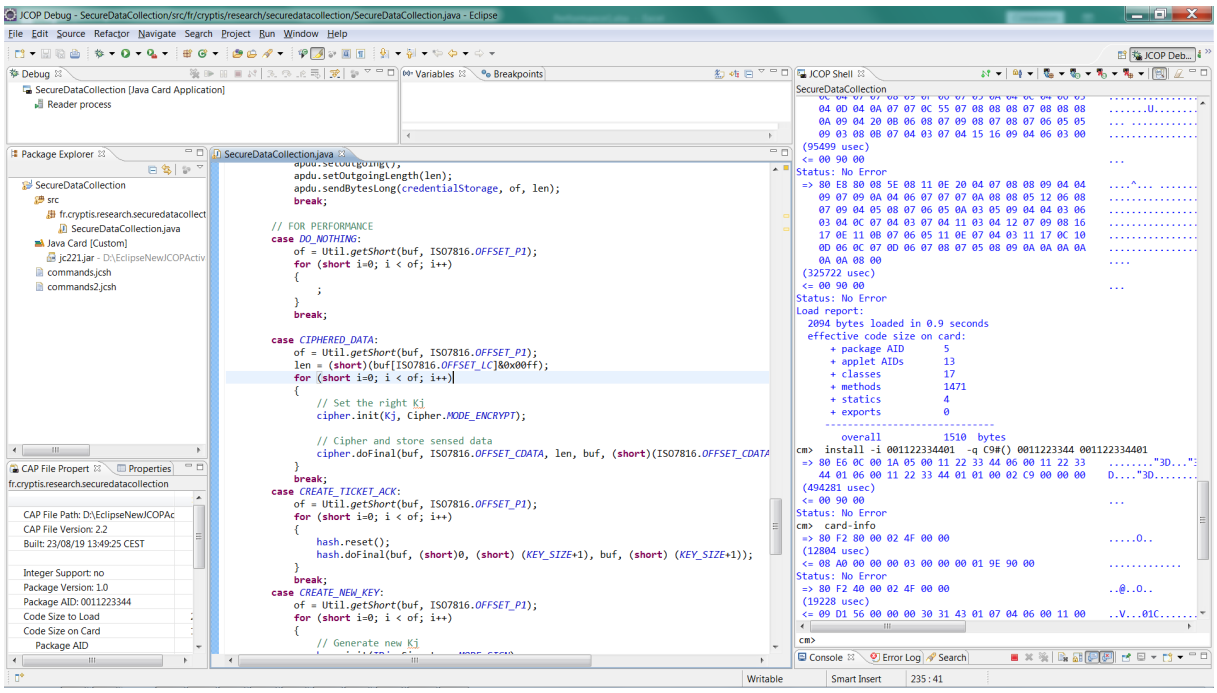


FIGURE 5.7 – Chargement et installation de l'applet sur la carte sous Eclipse avec le plug-in JCOP (vue debug)

nécessaire de la sélectionner. C'est donc tout naturellement que la première commande utilisée `/select` est une commande de script permettant de sélectionner notre application sur la carte. Cette commande de script calcule automatiquement la taille et envoie d'elle-même la bonne commande APDU :

```
/select 001122334401
```

Nous aurions pu obtenir le même résultat en utilisant la commande `/send`[4] suivi de la commande APDU suivante :

```
/send 00A4040006001122334401
```

Une fois l'application sélectionnée, toutes les autres commandes sont traitées par l'application (sauf une demande de sélection d'une autre application). À titre d'exemple, les

4. La commande `/send APDU` envoie de manière brute l'argument `APDU` en tant que commande APDU.

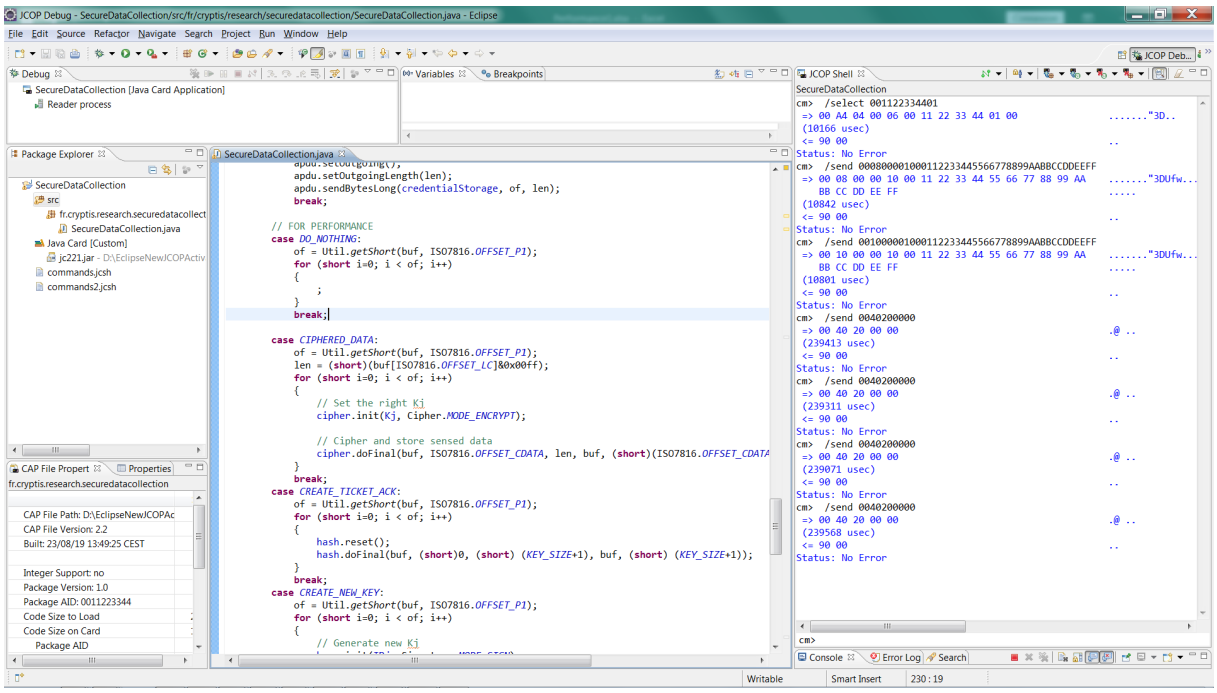


FIGURE 5.8 – Interaction avec notre applet depuis le “Shell” de la vue debug (Eclipse avec le plug-in JCOP)

deux commandes APDU suivantes initialisent simplement l'identifiant Id^j et la clé K_i^j (par exemple K_0^j) à la valeur `00112233445566778899AABBCCDDEEFF`.

```
/send 00080000100011223344556778899AABBCCDDEEFF
/send 00100000100011223344556778899AABBCCDDEEFF
```

La commande APDU suivante `0040200000` appelle simplement le cas `DO_NOTHING` du `switch` qui est visible sur le cadre du milieu. Cette dernière consiste à répéter `of` fois l'opération `;` (vide) où `of` vaut dans le cas de notre commande APDU `0x2000` soit `8192`. On notera figure 5.8 que la mesure de temps fourni par le “Shell” pour l'exécution successive de cette commande par la carte varie. Nous devons donc prendre ceci en compte dans notre méthodologie d'évaluation des performances présentée dans la section suivante.

5.3.3 Méthodologie pour évaluer la performance

Comme nous l'avons expliqué en section 5.3.1.3, afin d'éviter un biais dans les mesures de temps obtenues au niveau de l'application (par exemple, en raison de l'ordonnancement entre les différentes applications et services s'exécutant sur le PC) entre le temps d'envoi de la commande APDU et le temps de réception de la réponse APDU associée, la méthodologie adoptée a consisté à exécuter en boucle la même opération plusieurs milliers de fois en interne sur la carte pour calculer de la façon la plus exacte le temps moyen d'exécution pour une seule exécution. Pour déterminer précisément le temps interne nécessaire pour effectuer une opération spécifique, le temps d'exécution sur la carte de la boucle pour l'opération vide des milliers de fois a également été mesuré, moyenné et soustrait de la valeur précédente. Ainsi, sur chaque carte, nous avons effectué les mesures suivantes avec une précision de l'ordre de la microseconde :

— une opération vide ;

— le chiffrement avec l'algorithme AES-128 CBC sans remplissage pour des différentes tailles de données (plaintext), c'est-à-dire 16, 32, 64, 96 et 128 octets ;

— l'opération de hachage SHA-1, et SHA-256 lorsqu'elle est disponible, sur 17 octets (16 octets pour la taille de la clé K_i^j et 1 octet pour le caractère '1' ou '2') ;

— l'opération HMAC avec l'algorithme de MAC AES-128 pour différentes tailles de données (dans notre protocole, les données à traiter sont les données chiffrées envoyées par les ARs), c'est-à-dire 16, 32, 64, 96 et 128 octets.

Il est à noter que deux études visant à mesurer les performances sur différentes cartes Java ont été menées : le projet MESURE [196, 197] et le projet JCAlgTest [198]. Cependant, le premier portait principalement sur l'évaluation des performances des machines virtuelles, c'est-à-dire des instructions bytecode. Le travail effectué dans le cadre de ce

projet sur les API ne concernait pas les algorithmes cryptographiques, ce dont nous avions besoin. De plus, le code source n'est plus disponible. Le second projet est toujours actif et permet de mesurer les performances des algorithmes cryptographiques, mais uniquement sur des tailles de données spécifiques et non sur des tailles dont nous avions besoin. C'est un bon projet d'ingénierie logicielle, mais le modifier pour n'effectuer que des mesures spécifiques n'aurait pas été efficace. Après avoir examiné les deux projets précédents, notre méthodologie de mesure du temps d'exécution interne est valide et très proche de celle présentée dans les documents relatifs au projet MESURE [196].

Comme nous l'avons évoqué précédemment et en fin de section 5.3.2, l'exécution successive de la même commande APDU peut donner lieu à différentes mesures de temps (ces erreurs de mesures sont principalement liées à l'ordonnanceur de taches du sytèmes d'exploitation). Aussi, les résultats fournis dans la section 5.3.4 sont les moyennes de plusieurs exécutions (10 comme nous allons l'expliquer ci-dessous) afin de minimiser ces erreurs.

Par ailleurs, afin d'évaluer la surcharge entre les résultats obtenus sur le PC et le temps exact entre l'envoi de la commande à la carte et sa réponse au lecteur, comme illustré figure 5.9, nous avons utilisé un adapteur placé entre la carte et le lecteur qui nous permet de connecter un oscilloscope à n'importe quelle entrée de l'interface contact ISO7816 et en particulier l'I/O (Input/Output) pour faire cette mesure. Nous utilisons une seule voie de l'oscilloscope et nous connectons la masse sur le plot VSS (GND) et l'autre sur le plot I/O afin d'obtenir la courbe jaune que l'on peut voir figure 5.9. Pour obtenir des résultats précis, nous réglons le `TRIGGER` de l'oscilloscope sur la voie 1 pour un déclenchement au premier front montant au dessous de $\approx$ 2,5 V (symbole du trigger orange sur la droite de l'écran principal) puisque nous travaillons sur l'I/O entre 0 et 5 Volts. Ce front montant correspond à la fin du bit de `start` de la commande APDU (en fait TPDU mais il n'est pas nécessaire de détailler davantage ici).

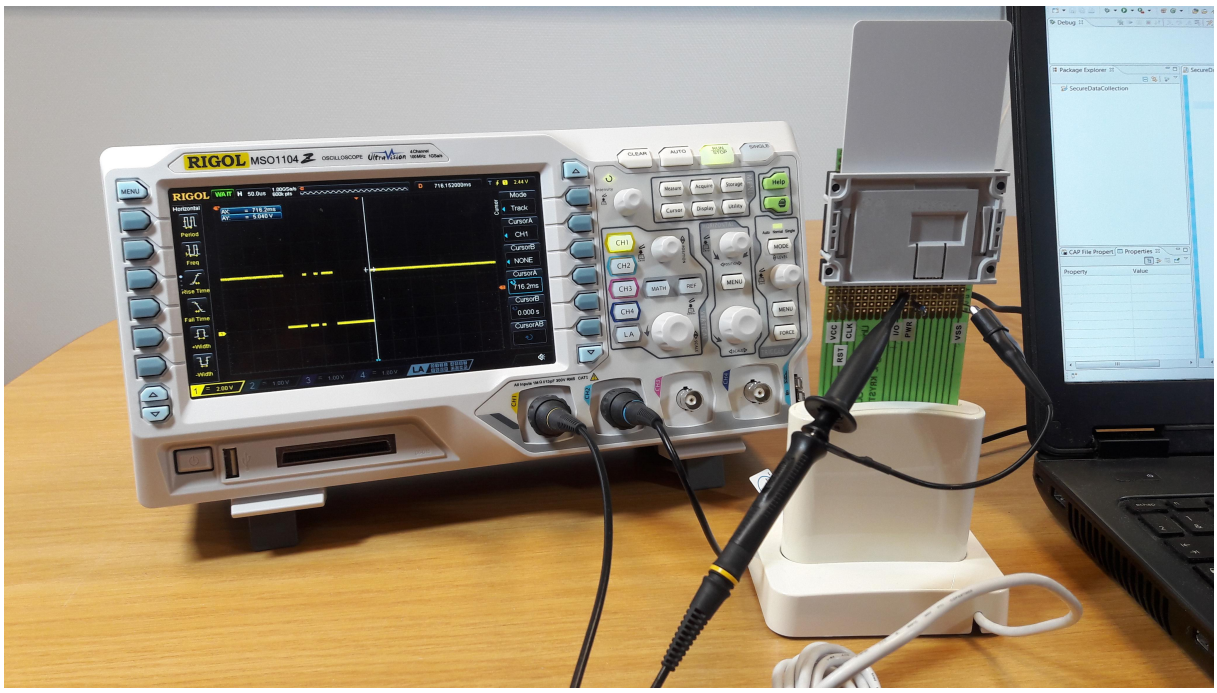


FIGURE 5.9 – Banc de test pour la validation des mesures

Une fois ce réglage effectué, nous envoyons la commande APDU à observer plusieurs fois de suite depuis le “Shell” afin de positionner la réponse APDU au milieu de l’écran et avec la meilleure résolution temporelle (c’est-à-dire que temporellement le début de la commande APDU se trouve en dehors de l’écran à gauche – c’est ainsi que figure 5.10 le petit symbole du trigger en haut à gauche de l’écran principal pointe sur la gauche pour expliquer où l’on « triggue », c’est-à-dire déclenche l’acquisition). Le motif de la voie 1 observable figure 5.10 correspond au « status word » `9000` pour lequel la carte qui indique que tout s’est bien passé. Comme on peut le voir sur cette mesure, la réponse APDU est reçu par le lecteur 716,2 ms après le début de l’envoi de la commande APDU.

Avant de considérer la surcharge nous présentons en tableau 5.2 les résultats des mesures obtenues par le “Shell” avec celles obtenues avec l’oscilloscope pour la carte F3 sur laquelle nous avons envoyé la commande APDU `0040200000` qui consiste à exécuter 8192 fois l’opération vide (ce qui est le meilleur cas car c’est une opération peu consommatrice

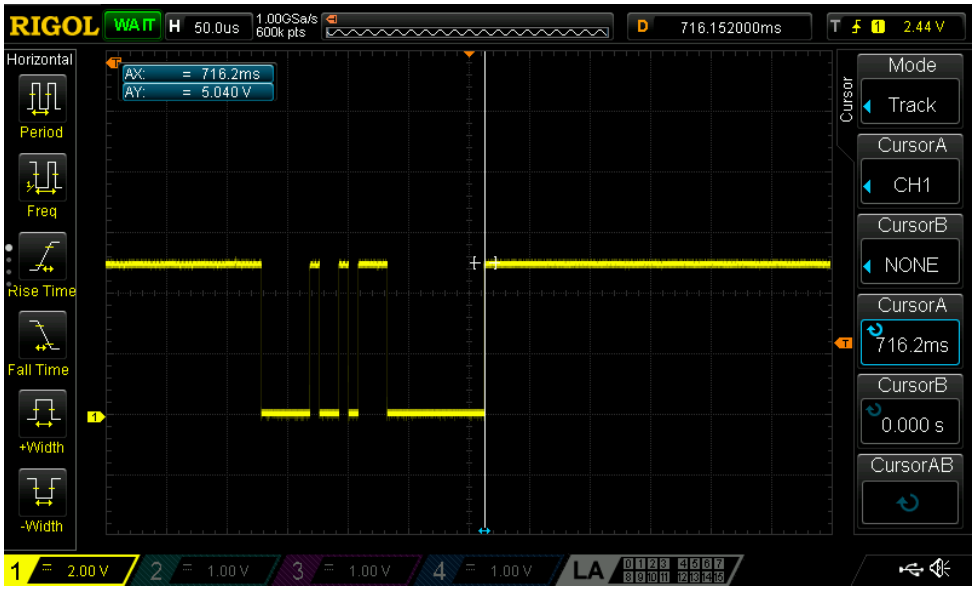


FIGURE 5.10 – Exemple d'un temps de réponse mesuré pour une carte avec l'oscilloscope

de temps et donc pour laquelle les différences entre les mesures seront le plus visible).

Tableau 5.2 – Comparaison des mesures des temps d'exécution sur la carte F3 de l'exécution de 8192 fois l'opération vide via le "Shell" et l'oscilloscope (μs)

	Moyen de mesure		**Comparaison**	
	"Shell"	**Oscilloscope**	Différence	Précision (%)
Exécution 1	718143	716300	1843	≈0,26
Exécution 2	717840	716100	1740	≈0,24
Exécution 3	718417	716400	2017	≈0,28
Exécution 4	718090	716200	1890	≈0,26
Moyenne	718122	716250	1872	**≈0,26**
σ	236	129	115	
Précision (%)	**≈0,03**	**≈0,02**		

Ainsi qu'on peut le constater tableau 5.2, la surcharge moyenne liée à l'ordonnanceur sur la machine est d'approximativement 0,26% du temps par rapport à la mesure réelle (celle réalisée avec l'oscilloscope). Nous pouvons donc considérer que prendre les mesures de temps du "Shell" est suffisant pour notre précision (rappelons nous que nous sommes dans ces calculs dans le cas le plus défavorable à nos mesures).

Concernant maintenant l'imprécision de mesure via le "Shell", lié en partie à l'ordonnanceur de la machine (mais aussi certainement à une latence au niveau du circuit asynchrone traitant les APDU sur la carte ou encore à la mise en œuvre de contre-mesures anti-synchronysation sur la carte – il y a une erreur de 0,02% même avec la mesure oscilloscope), elle est en moyenne de 0,03% pour les quatre mesures effectuées. C'est ainsi qu'afin de minimiser les erreurs sur les résultats qui seront présentés dans la section 5.3.4, nous avons effectué 10 fois les mêmes mesures pour les opérations présentées au début de la section (c'est-à-dire, opération vide, AES-128 CBC, SHA-1, SHA-256, AES-128 MAC – chacune étant répétée 8192 fois – et pour différentes tailles de données).

5.3.4 Performance pour les cartes testées

Les performances brutes sont fournies dans le tableau 5.3. Il est à noter que nous avons testé les algorithmes de chiffrement AES-128 et HMAC sur différentes tailles de données, car notre protocole n'impose pas de taille spécifique pour les données captées SD_i^j. Cependant, pour la fonction de hachage SHA-1/SHA-256, nous avons uniquement pris en compte une taille de 17 octets pour les données, puisque cela correspond à la taille des données à hacher pour une clé de chiffrement K_i^j de 128 bits (16 octets). La mesure de l'opération de dérivation de clé a également été incluse pour des raisons de complétude (même s'il s'agit d'une opération assez similaire au calcul HMAC).

La figure 5.11 récapitule les performances de différents algorithmes sur les cartes testées pour la plus petite taille de données pour chaque opération (16 octets pour AES-128 CBC sans remplissage, 16 octets pour AES-128 MAC et 17 octets pour SHA-1 et SHA-256). En première approche, on peut noter que les meilleurs résultats ont été obtenus avec les cartes les plus récentes (J1, J2, F1, F2). F3 ne peut pas vraiment être considérée car elle ne supporte pas deux algorithmes importants. U1 est une carte récente, mais compte tenu

Tableau 5.3 – Temps interne consacré à l'exécution des différentes opérations sur différentes tailles de données sur les cartes testées (μs)

Opération / Modéle de carte		J1	J2	F1	F2	F3	N1	N2	N3	N4	N5	N6	U1
Algorithme	**Données (octets)**												
AES-128 CBC NO PAD	16	1110	980	967	2797	4948	8670	8109	15068	16135	15401	14177	19420
AES-128 CBC NO PAD	32	1250	1151	1137	4041	7719	9139	8318	16156	17321	16523	15184	22663
AES-128 CBC NO PAD	64	1530	1494	1479	6528	13275	9970	8568	17826	19154	18244	16734	29161
AES-128 CBC NO PAD	96	1809	1835	1820	9017	18818	10804	8816	19495	20971	19952	18280	35659
AES-128 CBC NO PAD	128	2090	2180	2161	11506	▼	11636	9066	21131	22772	21656	19820	42151
SHA-1	17	2136	2678	1314	1374	1145	4995	4719	11475	12338	11718	10741	25092
SHA-256	17	5211	5937	5946	1378	✗	✗	✗	✗	✗	✗	✗	✗
HMAC (AES-128 MAC)	16	1080	965	1050	2747	✗	9386	8948	15747	16851	16060	14785	19515
HMAC (AES-128 MAC)	32	1198	1118	1204	3976	✗	9758	9057	16479	17657	16812	15462	22737
HMAC (AES-128 MAC)	64	1431	1420	1513	6433	✗	10500	9277	17942	19268	18304	16817	29180
HMAC (AES-128 MAC)	96	1664	1722	1821	8890	✗	11243	9497	19402	20877	19811	18153	35623
HMAC (AES-128 MAC)	128	1897	2024	2129	11346	✗	11986	9717	20865	22485	21305	19499	42054
Créer une nouvelle clé (HMAC)	16	1112	1002	1095	2825	✗	9430	8994	15852	16977	16167	14883	19535

✗ : Non supporté, ▼ : Erreur anormale.

du peu d'informations dont nous disposons, elle n'a pas été fabriquée par une entreprise expérimentée dans le développement et la production de cartes à puce, ce qui explique probablement sa faible performance.

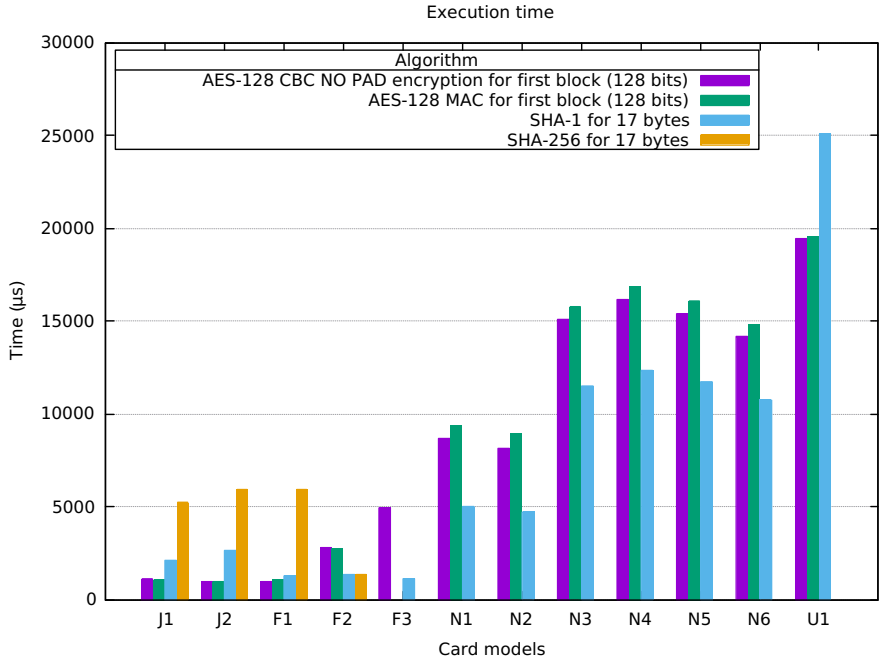


FIGURE 5.11 – Performances de différents algorithmes cryptographiques sur différentes Java Cards

À première vue, sur la base des résultats illustrés figures 5.11 et 5.12, il semble pertinent de ne considérer que les résultats des cartes J1, J2, F1, F2 pour extrapoler les performances attendues de notre protocole s'il fonctionnait sur des ARs. On peut remarquer que les cartes J1, J2 et F1 se trouvent exactement sur la même ligne dans la figure 5.12 et que par conséquent seule F1 apparaît. On peut également noter que les cartes dont les performances sont représentées par une ligne quasiment horizontale pour différentes tailles des données sont certainement équipées d'un crypto-processeur matériel AES (c'est-à-dire J1, J2, F1, N1, N2).

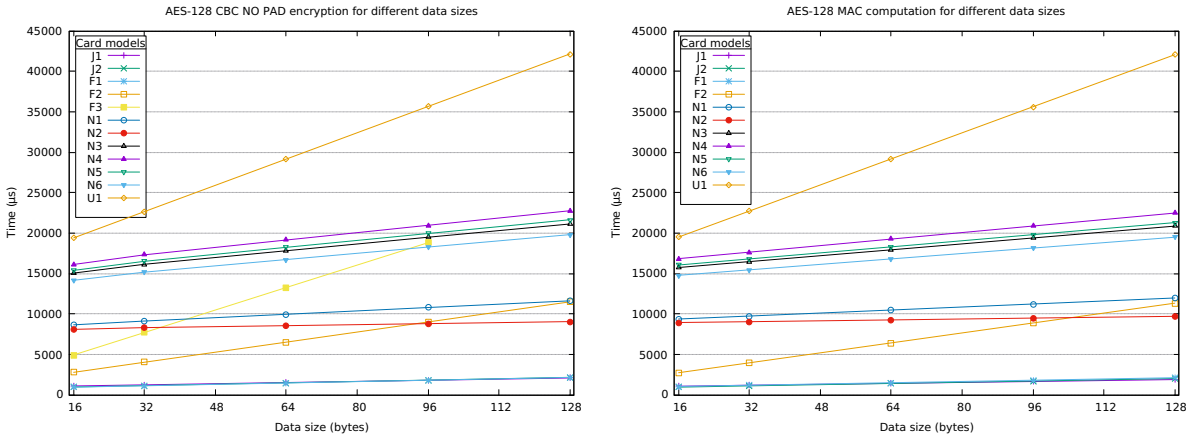


FIGURE 5.12 – Performances pour le chiffrement AES-128 CBC NO PAD et le calcul MAC pour différentes tailles de données

Cependant, les résultats donnés dans le tableau 5.4 montre que le temps interne d'exécution pour le chiffrement AES ou le calcul HMAC n'est pas proportionnel à la taille du premier bloc de données à traiter mais au temps nécessaire pour traiter le second bloc. En effet, le temps d'exécution est généralement linéaire en fonction du nombre de prochains blocs à traiter (dans notre contexte le nombre de blocs de 16 octets, soit 128 bits), car sur la carte (et sur d'autres dispositifs), lorsqu'une opération cryptographique est effectuée, il y a tout d'abord une phase d'initialisation qui prend un temps donné, quelle que soit la taille des données à traiter dans les phases ultérieures de l'algorithme cryptogra-

phique. Donc, pour obtenir une estimation précise des performances attendues de notre protocole, il faudra en tenir compte. En se basant sur la figure 5.13 et sur le tableau 5.4 qui fournissent la surcharge réelle pour le prochain bloc à traiter pour les algorithmes de chiffrement et de HMAC pour chaque carte, la carte N2 pourrait également être considérée comme un candidat à retenir. Encore une fois, il faut noter que les cartes J1 et N2 d'une part et que J2 et F1 d'autre part se trouvent exactement sur les mêmes lignes sur la figure 5.13 et que seules N2 et F1 sont visibles. Étant donné que la surcharge par bloc est presque constante pour J1, J2, F1, N1 et N2, cela semble confirmer l'hypothèse selon laquelle ces cartes seraient équipées d'un crypto-processeur matériel. En résumé, les cartes suivantes récemment produites par des fabricants de cartes à puce bien connus peuvent être prises en compte pour nos extrapolations : J1, J2, F1, N1 et N2.

Tableau 5.4 – Surcoût par bloc supplémentaire de 128 bits pour le chiffrement AES-128 CBC NO PAD et les calculs MAC AES-128 (μs)

Modèle de carte / **Surcoût**	J1	J2	F1	F2	F3	N1	N2	N3	N4	N5	N6	U1
AES-128 CBC NO PAD/bloc	140	171	171	1244	2774	424	137	866	948	894	806	3247
AES-128 MAC/bloc	117	151	154	1228	✗	371	110	731	805	749	673	3220

✗ : Non supporté.

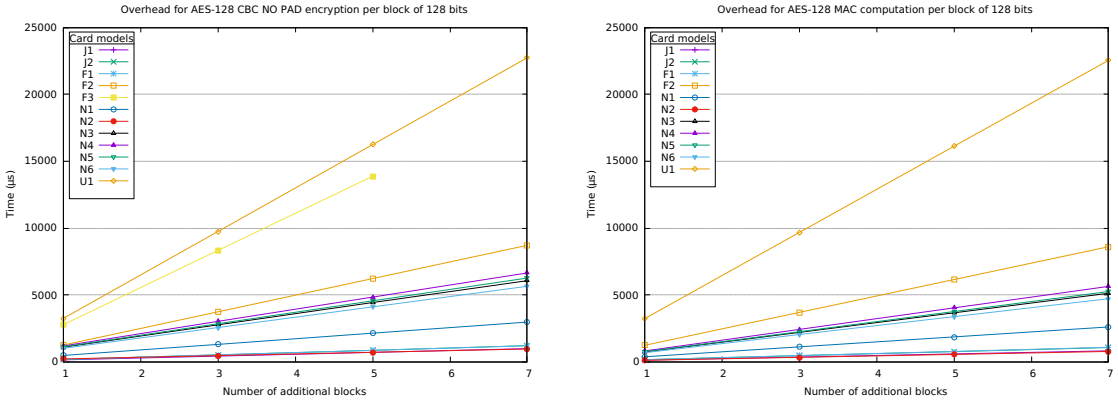


FIGURE 5.13 – Surcoût pour les blocs suivants (128 bits) pour le chiffrement AES CBC et le calcul du HMAC AES MAC

5.3.5 Évaluation sur un cas réaliste

D'après les résultats précédents pour J1, J2, F1, N1 et N2, le tableau 5.5 fournit les performances que l'on peut attendre d'un AR pour les différentes opérations de base (les valeurs présentées sont des moyennes de résultats mesurés).

Tableau 5.5 – Temps interne attendu pour exécuter les différentes opérations sur un AR (μs)

AES-128 CBC NO PAD		**SHA1**	**SHA-256**	**HMAC (AES-128 MAC)**		**Créer une nouvelle clé (HMAC)**
Premier bloc (16 octets)	**par bloc supplémentaire** (16 octets)	(17 octets)	(17 octets)	**Premier bloc** (16 octets)	**par bloc supplémentaire** (16 octets)	(16 octets)
3967	209	3168	5698	4468	181	4327

Pour calculer la consommation de mémoire pour chaque élément de données captées SD_i^j, pour simplifier supposons que la taille est un multiple de 16 octets – c'est-à-dire la taille du bloc –, soit celle de SD_i^j et des T_i^j et HM_i^j : c'est-à-dire la taille de $SD_i^j + 32$ octets.

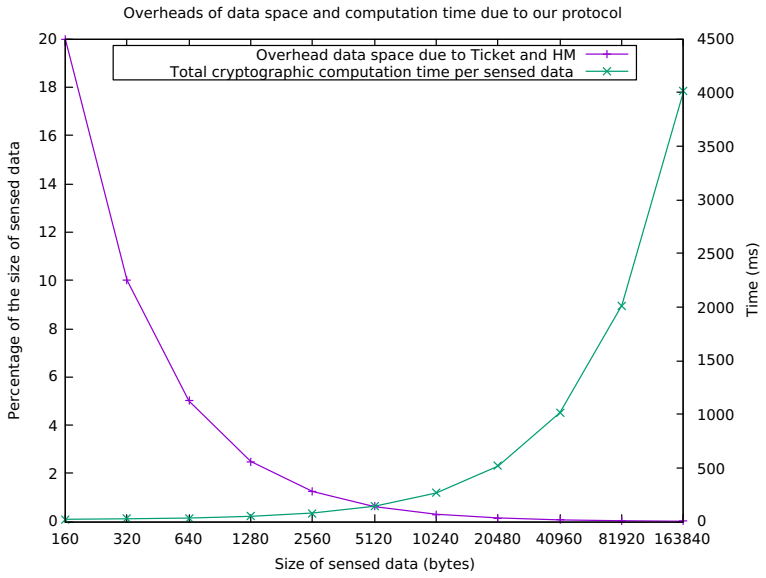


FIGURE 5.14 – Surcoûts généraux d'espace de données et de temps de calcul cryptographique pour différentes tailles de données mesurées

Comme illustré figure 5.14, pour laquelle une échelle logarithmique est utilisée pour l'axe des ordonnées qui représentent la taille des données captées, la surcharge mémoire

de notre solution est très faible (moins de 1%) lorsque la taille des données captées est supérieure à 3,2 ko. En outre, il est à noter que le temps de calcul dû aux opérations cryptographiques est très raisonnable pour les tailles de données détectées inférieures à 40 ko (inférieure à 1 seconde) sachant que nos résultats sont basés sur les performances de cartes à puce.

Pour évaluer la pertinence de notre proposition, nous avons considéré que la taille des données était de 20480 octets pour les opérations de mesure. Si nous supposons qu'un AR est équipé d'une mémoire non volatile de 16 Go (il s'agit là d'une hypothèse raisonnable, car les cartes SD de cette taille sont très courantes et peu coûteuses) et qu'un échantillon de données détectées est prélevé toutes les 10 secondes, alors un AR peut stocker les données captées pendant plus de 90 jours sans que la collecte de données ne se produise. Si un MDC collecte des données avant que la mémoire ne soit saturée, ce qui est préférable, et même toujours possible à 90 jours. La mesure toutes les 10 secondes est pertinente car 518 ms sont nécessaires pour chiffrer les données et calculer T_i^j, ACK_i^j, HM_i^j et la clé de chiffrement suivante K_{i+1}^j. Ensuite, l'AR peut s'endormir pour économiser de l'énergie et prolonger la vie de sa batterie.

Il convient de noter que, dans notre protocole, le MDC doit effectuer une seule opération de HMAC. Il est donc efficace et dans le pire des cas (c'est-à-dire en utilisant les performances des algorithmes cryptographiques lourds mesurés sur des cartes à puce), un MDC prendrait 236 ms par 20480 octets de données captées reçues. Cela signifierait qu'un MDC aurait besoin d'un peu plus de 30 minutes pour collecter toutes les données d'un nœud AR ayant une mémoire pleine si la stratégie appliquée par le MDC consiste à vérifier l'intégrité de chaque élément de données reçu. Toutefois, le MDC étant plus puissant que le nœud AR, ce temps peut être diminué d'un facteur supérieur à 10. Notre protocole peut également être adapté à différentes politiques pour les calculs HM_i^j, en fonction des

exigences de performance. Par exemple, un scénario peut nécessiter que seuls certains HM_i^j soient calculés lorsque le MDC les demande. Si c'est le cas, il n'est pas efficace énergétiquement pour l'AR de calculer tous les HM_i^j à l'avance ; cela peut se faire lors de la phase de collecte des données au prix d'une latence un peu supérieure car les données à envoyer nécessiteront le calcul HM_i^j. En effet, si cela n'est pas nécessaire pour chaque élément de données captées, l'AR peut différer le calcul qu'il effectue à la phase de mesure en ne gardant en mémoire que ACK_i^j pour calculer le HM_i^j lorsque cela sera nécessaire. En termes de surcharge mémoire, cette solution est similaire à celle analysée ci-dessus, puisque ACK_i^j et HM_i^j ont la même taille, mais elle serait clairement plus efficace énergétiquement même si en revanche c'est au détriment d'un niveau de sécurité inférieur. Il est à noter que dans la section 4.4, nous avons décidé de présenter et d'analyser la version du protocole avec le plus haut niveau de sécurité. Du coté des MDCs, notre protocole satisfait aussi les exigences de performance énoncées section 4.2.4 et par conséquent les résultats expérimentaux valident l'efficacité de notre proposition de protocole serverless de collecte sécurisée des données dans le contexte des ARs.

Conclusion

Dans ce chapitre, nous avons évalué l'aptitude de notre proposition à satisfaire les exigences de performance par le biais d'une évaluation complète des performances de nos opérations sur les périphériques les plus limités en ressources, dont les résultats ont été extrapolés aux nœuds ARs. Nous avons pu déterminer que la surcharge de mémoire était faible tant que les données détectées étaient suffisamment volumineuses et que le temps requis pour effectuer des opérations cryptographiques était absolument compatible avec le processus de mesure et permettait aux nœuds ARs de passer en mode veille tant que

la fréquence d'échantillonnage n'est pas trop élevée.

Nous pouvons donc maintenant conclure que notre proposition de protocole de collecte de données est à la fois sécurisée et efficace pour les scénarios sans serveur impliquant des nœuds ARs.

Conclusion et perspectives

Comme nous avons pu le constaster lors du premier chapitre de ce mémoire, les nœuds RFID actifs, qui ont été le premier type de RFID avec la technologie IFF lors de la seconde guerre mondiale, ont vu leur développement supplanté par la technologie RFID passive pendant plus d'un demi siècle, pour aujourd'hui retrouver une place prépondérante dans ce qu'on nomme couramment l'IdO (IoT). Dans ce contexte, nous avons pu voir qu'il n'existe pas un unique type de nœuds ARs, mais une multitude de périphériques qui, s'ils partagent des caractéristiques communes que nous avons définies, ont aussi leurs propres spécificités afin de répondre aux besoins des applications. Ainsi, nous avons détaillé leur architecture, leurs critères (fréquence, technologies de communication, sources d'énergie, etc.) et leur mode de fonctionnement dont l'un des plus prometteurs pour la plupart des contextes applicatifs utilisant les nœuds ARs est le mode sans serveur. C'est pourquoi lors de la suite de nos travaux nous nous sommes intéressés à proposer une solution de sécurité flexible et efficace pour la collecte de données sans serveur dans un environnement hostile.

La proposition que nous avons présentée dans la seconde partie de ce mémoire résulte de l'étude des solutions de sécurité proposées dans la littérature pour les nœuds RFID actifs, conduite au second chapitre. Nous avons pu constater que la plupart des menaces et attaques applicables à la technologie RFID passive l'étaient aussi pour la technologie

RFID active. En revanche, comme les nœuds ARs disposent d'un peu plus de ressources que les tags passifs, il est possible d'utiliser des algorithmes cryptographiques un peu plus évolués (mais toujours légers). En particulier, nous nous avons analysé les solutions de sécurité de la littérature pour les protocoles sans serveur et nous avons constaté qu'il n'y avait pas de solutions satisfaisantes pour la collecte de données en particulier dans le cas d'un environnement hostile dans lequel les nœuds sont susceptibles d'être capturés.

Le chapitre 2 nous a aussi permis de constater que les protocoles de sécurité sont souvent l'objet de failles ou d'erreurs découvertes dans la plupart d'entre eux bien des années plus tard. Ainsi, nous avons souhaité utiliser des outils de vérification formelles de protocoles pour notre proposition afin de procéder à la vérification de son exactitude et de sa conformité vis-à-vis des propriétés revendiquées pour pouvoir l'utiliser en toute confiance. Nous avons donc détaillé au chapitre 3 les différentes méthodes de vérification formelle des protocoles de sécurité.

Sur la base des travaux présentés dans la première partie de ce mémoire, nous avons pu lors de la seconde partie proposer un protocole de sécurité sans serveur de collecte de données qui prend en charge des opérations cryptographiques légères telles que les fonctions de hachage et HMAC pour répondre aux exigences de sécurité et de performance souhaitées. Ce protocole a été déployé dans un scénario très exigeant puisque les nœuds ARs opèrent dans un environnement hostile. Il met en jeu différents ARs, comme des drones qui jouent le role de MDC et les nœuds capteurs qui procèdent aux mesures des données. Comme nous l'avons montré et validé avec des outils de vérification formelle (AVISPA et ProVerif) au chapitre 4, notre protocole assure non seulement la confidentialité de bout en bout des données mesurées (des capteurs à la station de base), l'authentification mutuelle entre les nœuds ARs et les MDCs, mais également l'intégrité des données envoyées pendant la phase de collecte.

Au chapitre 5, nous avons démontré la pertinence de notre proposition à répondre aux exigences de performance au travers d'une évaluation complète des performances de notre protocole sur les dispositifs les plus limités en ressources (c'est-à-dire des cartes à puce), dont les résultats ont été extrapolés aux nœuds ARs considérés. Nous avons ainsi pu déterminer que la surcharge mémoire était faible tant que les données mesurées étaient suffisamment volumineuses et que le temps requis pour effectuer les opérations cryptographiques était absolument compatible avec le processus de mesure et que cela permettait même aux nœuds ARs de passer en veille tant que la fréquence d'échantillonnage souhaitée n'était pas trop élevée.

En conclusion, sur la base de nos résultats de performance et des analyses et preuves de sécurité, nous pouvons affirmer que notre protocole de collecte sécurisée des données est à la fois sûr et efficace même lorsqu'il est utilisé dans un environnement hostile. Dès lors, il s'applique sans aucun problème à des contextes moins exigeants.

De notre point de vue, si aujourd'hui les nœuds ARs de l'actuel IdO (IoT) sont essentiellement fixes, demain ils seront mobiles pour permettre l'avènement de l'IdOM (Internet des Objets Mobiles ou M-IoT) et il faudra prendre en compte les problématiques de mobilité. Alors que les experts prévoient une croissance exponentielle du nombre de périphériques dans l'IdO/IdOM, malgré le déploiement de futures technologies de communication comme la 5G, les nœuds ARs devront être sobres en terme d'utilisation de la bande passante, au risque de ne plus pouvoir communiquer.

Si notre proposition n'est certes pas une réponse universelle, elle peut être un début de solution puisqu'elle prend en compte partiellement la mobilité de nœuds (les MDC et les MAR) et qu'elle comporte un nombre d'échanges minime.

Bibliographie

[1] Wisp. https ://wisp5.wikispaces.com/WISP+Home., .

[2] Phase iv. , https ://www.phaseivengr.com/., .

[3] Thinfilm. http ://www.thinfilm.no/, .

[4] Benjamin Khoo. Rfid as an enabler of the internet of things : Issues of security and privacy. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pages 709–712. IEEE, 2011.

[5] Kamran Ahsan, Hanifa Shah, and Paul Kingston. Rfid applications : An introductory and exploratory study. *arXiv preprint arXiv :1002.1179*, 2010.

[6] Arnaud Vena. *Contribution au développement de la technologie RFID sans puce à haute capacité de codage.* PhD thesis, Université de Grenoble, 2012.

[7] Dat Son Nguyen. *Développement des capteurs sans fil basés sur les tags RFID uhf passifs pour la détection de la qualité des aliments.* PhD thesis, Université de Grenoble, 2013.

[8] Identification Friend-or-Foe. https ://www.nrl.navy.mil/accomplishments/systems/friend-or-foe/, .

[9] EiC. Gillette to buy 500 million epc tags. *RFID Journal*, 2002.

[10] John Wehr. The wal-mart effect : Retailer mandates rfid tagging of product shipments and others follow suit. *SecureIDNews*, 2004.

[11] Edmund W Schuster, Stuart J Allen, and David L Brock. *Global RFID : the value of the EPCglobal network for supply chain management*. Springer Science & Business Media, 2007.

[12] Klaus Finkenzeller. *RFID handbook : fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. John Wiley & Sons, 2010.

[13] P Kumar, HW Reinitz, J Simunovic, KP Sandeep, and PD Franzon. Overview of rfid technology and its applications in the food industry. *Journal of food science*, 74(8) :R101–R106, 2009.

[14] Vipul Chawla and Dong Sam Ha. An overview of passive rfid. *IEEE Communications Magazine*, 45(9), 2007.

[15] NFC FORUM. What is NFC ? https ://nfc-forum.org/what-is-nfc/, .

[16] Tags and Features. http ://www.rfidhand.de/tagfeatures.html.

[17] Wavelinx. LOW FREQUENCY TAGS. http ://wavelinx.in/wave/clear-disc-tag/.

[18] LibBest. Library RFID System - HF RFID Tag. http ://www.rfid-library.com/eng_hftag.html/, .

[19] UHF Alien H3 9654 RFID label. UHF . https ://www.rfidcardfactory.com/uhf-alien-h3-9654-rfid-label_p299.html, .

[20] Linda Castro and Samuel Fosso Wamba. An inside look at rfid technology. *Journal of Technology Management & Innovation*, 2(1) :128–141, 2007.

[21] Michel Baudin and Arun Rao. Rfid applications in manufacturing, 2000.

[22] Mark Roberti. Hp takes rfid end to end. *RFID Journal White Paper Library*, 2006.

[23] Suhong Li, John K Visich, Basheer M Khumawala, and Chen Zhang. Radio frequency identification technology : applications, technical challenges and strategies. *Sensor Review*, 26(3) :193–202, 2006.

[24] Claire Swedberg. Hospital uses rfid for surgical patients. *RFID Journal*, 2005.

[25] Mary Catherine O'Connor. Toronto-area hospital kicks off asset-tracking pilot. *RFID J*, 2006.

[26] Elisabeth Ilie-Zudor, Zsolt Kemény, Fred Van Blommestein, László Monostori, and André Van Der Meulen. A survey of applications and requirements of unique identification systems and rfid techniques. *Computers in Industry*, 62(3) :227–252, 2011.

[27] Jonathan Collins. Dod tries tags that phone home. *RFID Journal*, 2006.

[28] Alan D Smith. Exploring radio frequency identification technology and its impact on business systems. *Information Management & Computer Security*, 13(1) :16–28, 2005.

[29] EMVCo. Contactless Specifications. Online, 2019.

[30] David Dressen. Considerations for rfid technology selection. *Atmel Applications Journal*, 3 :45–47, 2004.

[31] Ari Juels, Ronald Rivest, and Michael Szydlo. The blocker tag : Selective blocking of rfid tags for consumer privacy. In ., pages 103–111, 01 2003.

[32] Melanie R Rieback, Bruno Crispo, and Andrew S Tanenbaum. Rfid guardian : A battery-powered mobile device for rfid privacy management. In *Australasian Conference on Information Security and Privacy*, pages 184–194. Springer, 2005.

[33] Damith C Ranasinghe, Daniel W Engels, and Peter H Cole. Low cost rfid systems : confronting security and privacy. *Paper Auto-ID Labs White Paper Journal*, 1, 2005.

[34] Stephen A Weis, Sanjay E Sarma, Ronald L Rivest, and Daniel W Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *Security in pervasive computing*, pages 201–212. Springer, 2004.

[35] Hung-Yu Chien and Che-Hao Chen. Mutual authentication protocol for rfid conforming to epc class 1 generation 2 standards. *Computer Standards & Interfaces*, 29(2) :254–259, 2007.

[36] Hung-Yu Chien. Sasi : A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity. *IEEE transactions on dependable and secure computing*, 4(4) :337–340, 2007.

[37] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M Estévez-Tapiador, and Arturo Ribagorda. Lmap : A real lightweight mutual authentication protocol for low-cost rfid tags. In *Proc. of 2nd Workshop on RFID Security*, page 06, 2006.

[38] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M Estevez-Tapiador, and Arturo Ribagorda. M 2 ap : a minimalist mutual-authentication protocol for low-cost rfid tags. In *International conference on ubiquitous intelligence and computing*, pages 912–923. Springer, 2006.

[39] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M Estevez-Tapiador, and Arturo Ribagorda. Emap : An efficient mutual-authentication protocol for low-cost rfid tags. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 352–361. Springer, 2006.

[40] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for rfid systems using the aes algorithm. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 357–370. Springer, 2004.

[41] Batbold Toiruul and KyungOh Lee. An advanced mutual-authentication algorithm using aes for rfid systems. *International Journal of Computer Science and Network Security*, 6(9) :156–162, 2006.

[42] Collins Mtita. *Lightweight serverless protocols for the internet of things*. PhD thesis, Institut National des Télécommunications, June 2016.

[43] Sujesha Sudevalayam and Purushottam Kulkarni. Energy harvesting sensor nodes : Survey and implications. *IEEE Communications Surveys & Tutorials*, 13(3) :443–461, 2011.

[44] Abhiman Hande, Raj Bridgelall, and Ben Zoghi. Vibration energy harvesting for disaster asset monitoring using active rfid tags. *Proceedings of the IEEE*, 98(9) :1620–1628, 2010.

[45] Abhiman Hande, Raj Bridgelall, and Dinesh Bhatia. Energy harvesting for active rf sensors and id tags. In *Energy Harvesting Technologies*, pages 459–492. Springer, 2009.

[46] Amin Rida, Li Yang, and Manos M Tentzeris. *RFID-enabled sensor design and applications*. Artech House, 2010.

[47] Atmosic. https://www.atmosic.com/, 2019. Online ; accédé le 29 Août 2019.

[48] Maurizio Di Paolo Emilio. The next wave of iot bluetooth devices might not have batteries. *EETimes*, 2019.

[49] Raed Abdulla and Widad Ismail. Survey of wsn technology based reliable and efficient active rfid. In *Communications (MICC), 2013 IEEE Malaysia International Conference on*, pages 116–121. IEEE, 2013.

[50] Minoru Katayama, Hiroshi Nakada, Hitoshi Hayashi, and Masashi Shimizu. Survey of rfid and its application to international ocean/air container tracking. *IEICE Transactions on communications*, 95(3) :773–793, 2012.

[51] Priyanka Rawat, Kamal Deep Singh, Hakima Chaouchi, and Jean Marie Bonnin. Wireless sensor networks : a survey on recent developments and potential synergies. *The Journal of supercomputing*, 68(1) :1–48, 2014.

[52] Jialiu Lin, Yunhuai Liu, and Lionel M Ni. Sida : self-organized id assignment in wireless sensor networks. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*, pages 1–8. IEEE, 2007.

[53] Semiconductor n nrf51822 bluetooth smart beacon kit. http ://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822-Bluetooth-Smart-Beacon-Kit., .

[54] ASensor. http ://wiki.aprbrother.com/wiki/ASensor.

[55] https ://www.frenchweb.fr/apple-pay-sappuie-sur-bpce-et-carrefour-pour-se-lancer-en-france/246890.

[56] Zhao Y. NFC-WISP website. https ://nfc-wisp.wikispaces.com/.

[57] Yi Zhao, Joshua R Smith, and Alanson Sample. Nfc-wisp : A sensing and computationally enhanced near-field rfid platform. In *RFID (RFID), 2015 IEEE International Conference on*, pages 174–181. IEEE, 2015.

[58] Andrej Simko. Security of car keys. http ://www.slideshare.net/Andrejimko/security-of-car-keys, 2014.

[59] What is iBeacon ? What are iBeacons ? . http ://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/.

[60] Eddystone vs Physical Web vs iBeacon : Why Eddystone will Rule the Beacon Space. https ://blog.beaconstac.com/2016/08/eddystone-vs-physical-web-vs-ibeacon-why-eddystone-will-rule-the-beacon-space/.

[61] Johanna Williams. Telosb Sensor Network – Helpful in Many Different Applications. https ://telosbsensors.wordpress.com/, 2014.

[62] AeroScout T5 Sensor Tags. https ://www.extronics.com/product/t5-sensor-tag/.

[63] Shigeng Zhang, Xuan Liu, Jianxin Wang, Jiannong Cao, and Geyong Min. Energy-efficient active tag searching in large scale rfid systems. *Information Sciences*, 317 :143–156, 2015.

[64] PC Jain and KP Vijaygopalan. Rfid and wireless sensor networks. *Proceedings of ASCNT*, pages 1–11, 2010.

[65] Hai Liu, Miodrag Bolic, Amiya Nayak, and Ivan Stojmenovic. Taxonomy and challenges of the integration of rfid and wireless sensor networks. *IEEE network*, 22(6) :., 2008.

[66] Matthias Bartholmai, Markus Stoppel, Sergej Petrov, Stefan Hohendorf, and Thomas Goedecke. Two application examples of rfid sensor systems-identification and diagnosis of concrete components and monitoring of dangerous goods transports. *ACTA IMEKO*, 4(2) :85–89, 2015.

[67] Lionel M Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P Patil. Landmarc : indoor location sensing using active rfid. *Wireless networks*, 10(6) :701–710, 2004.

[68] Jeffrey Hightower, Roy Want, and Gaetano Borriello. Spoton : An indoor 3d location sensing technology based on rf signal strength. ., page ., 2000.

[69] Yiyang Zhao, Yunhao Liu, and Lionel M Ni. Vire : Active rfid-based localization using virtual reference elimination. In *Parallel Processing, 2007. ICPP 2007. International Conference on*, pages 56–56. IEEE, 2007.

[70] Raed Abdulla. A conceptual study of long range active rfid system for reliable data communication. *IET*, page ., 2014.

[71] Horng-Lin Shieh, Chi-Chang Huang, Fu-Siang Lyu, Zh-Chun Zhang, and Ting-Syu Zheng. An emergency care system using rfid and zigbee techniques. In *Computer, Consumer and Control (IS3C), 2016 International Symposium on*, pages 65–68. IEEE, 2016.

[72] Henrik Ljøgodt Moen. A study of wi-fi rfid tags in citywide wireless networks : How well do wi-fi rfid tags work in outdoor wi-fi networks and is it possible to build commercial services based on wi-fi rfid tags in citywide wireless networks ? Master's thesis, Institutt for telematikk, 2007.

[73] Bing Wang, Mehdi Toobaei, Rodney Danskin, Thanaporn Ngarmnil, Larry Pham, and Harry Pham. Evaluation of rfid and wi-fi technologies for rtls applications in healthcare centers. In *Technology Management in the IT-Driven Services (PICMET), 2013 Proceedings of PICMET'13 :*, pages 2690–2703. IEEE, 2013.

[74] Majid Baghaei Nejad, Zhuo Zou, David S Mendoza, and Li-Rong Zheng. Remotely uhf-powered ultra wideband rfid for ubiquitous wireless identification and sensing. In *Development and Implementation of RFID Technology*, page . InTech, 2009.

[75] Fausto G Costa, Jó Ueyama, Torsten Braun, Gustavo Pessin, Fernando S Osório, and Patrícia A Vargas. The use of unmanned aerial vehicles and wireless sensor network in agricultural applications. In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pages 5045–5048. IEEE, 2012.

[76] Bruno S Faiçal, Fausto G Costa, Gustavo Pessin, Jó Ueyama, Heitor Freitas, Alexandre Colombo, Pedro H Fini, Leandro Villas, Fernando S Osório, Patrícia A Vargas, et al. The use of unmanned aerial vehicles and wireless sensor networks for spraying pesticides. *Journal of Systems Architecture*, 60(4) :393–404, 2014.

[77] Milan Erdelj, Michał Król, and Enrico Natalizio. Wireless sensor networks and multi-uav systems for natural disaster management. *Computer Networks*, 2017.

[78] Tao Wu, Panlong Yang, Yubo Yan, Xunpeng Rao, Ping Li, and Wanru Xu. Orsca : Optimal route selection and communication association for drones in wsns. In *Advanced Cloud and Big Data (CBD), 2017 Fifth International Conference on*, pages 420–424. IEEE, 2017.

[79] Hui-Ru Cao, Zhi Yang, and Ye-Qian Li. A mobile wsn sink node using unmanned aerial vehicles : Design and experiment. *International Journal of Network & Mobile Technologies*, 10(3), 2016.

[80] Vishal Sharma, Ilsun You, and Rajesh Kumar. Energy efficient data dissemination in multi-uav coordinated wireless sensor networks. *Mobile Information Systems*, 2016, 2016.

[81] Imad Jawhar, Nader Mohamed, Jameela Al-Jaroodi, and Sheng Zhang. A framework for using unmanned aerial vehicles for data collection in linear wireless sensor networks. *Journal of Intelligent & Robotic Systems*, 74(1-2) :437, 2014.

[82] Nemai Chandra Karmakar. *Advanced RFID Systems, Security, and Applications*. IGI Global, Hershey, PA, USA, 1st edition, 2012.

[83] Dirk Henrici. *RFID Security and Privacy : Concepts, Protocols, and Architectures (Lecture Notes in Electrical Engineering)*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[84] Miodrag Bolic, David Simplot, and Ivan Stojmenovic. *RFID systems : research trends and challenges*. Wiley, Chichester, West Sussex, 2010.

[85] Gildas Avoine and Philippe Oechslin. *Financial Cryptography and Data Security : 9th International Conference, FC 2005, Roseau, The Commonwealth Of Dominica, February 28 – March 3, 2005. Revised Papers*, chapter RFID Traceability : A Multilayer Problem, pages 125–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[86] Simon. Garfinkel, Ari. Juels, and Ravi. Pappu. RFID privacy : an overview of problems and proposed solutions. *Security Privacy, IEEE*, 3(3) :34–43, May 2005.

[87] A. Karygiannis, T. Phillips, and A. Tsibertzopoulos. RFID Security : A Taxonomy of Risk. In *Communications and Networking in China, 2006. ChinaCom'06. First International Conference on*, pages 1–8, 2006.

[88] Ding Zhen-hua, Li Jin-tao, and Feng Bo. A taxonomy model of RFID security threats. In *Communication Technology, 2008. ICCT 2008. 11th IEEE International Conference on*, pages 765–768, Nov 2008.

[89] L. Mirowski, J. Hartnett, and R. Williams. An RFID Attacker Behavior Taxonomy. *Pervasive Computing, IEEE*, 8(4) :79–84, Oct 2009.

[90] Aikaterini Mitrokotsa, Melanie Rieback, and Andrew Tanenbaum. Classifying rfid attacks and defenses. *Information Systems Frontiers*, 12 :491–505, 2010.

[91] Aikaterini Mitrokotsa, Michael Beye, and Pedro Peris-Lopez. *Unique Radio Innovation for the 21st Century*, chapter Security Primitive Classification of RFID Attacks, pages 39–63. Springer, 2011.

[92] Li Hong, Chen YongHui, and He Zhangqing. The Survey of RFID Attacks and Defenses. In *Wireless Communications, International Conference on Networking*

and Mobile Computing (WiCOM), 2012 8th, pages 1–4, Sept 2012.

[93] Marco Spruit and Wouter Wester. Rfid security and privacy : Threats and countermeasures. In ., volume ., 2013.

[94] J Aragones-Vilella, Antoni Ballesté, and Agusti Solanas. A brief survey on rfid privacy and security. In ., pages 1488–1493, 01 2007.

[95] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum. Rfid guardian : A battery-powered mobile device for rfid privacy management. In Colin Boyd and Juan Manuel González Nieto, editors, *Information Security and Privacy*, pages 184–194, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[96] Mojtaba Alizadeh, Mazdak Zamani, Ali Rafiei Shahemabadi, Jafar Shayan, and Ahmad Azarnik. A survey on attacks in rfid networks. ., 2014.

[97] N. Saxena, M. B. Uddin, J. Voris, and N. Asokan. Vibrate-to-unlock : Mobile phone assisted user authentication to multiple personal rfid tags. In *2011 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 181–188, March 2011.

[98] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[99] Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm. *Fast Software Encryption : 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, chapter New Lightweight DES Variants, pages 196–210. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[100] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. Present : An ultra-lightweight block cipher. In *Procee-*

dings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems, CHES '07, pages 450–466, Berlin, Heidelberg, 2007. Springer-Verlag.

[101] Christophe Cannière, Orr Dunkelman, and Miroslav Knežević. Katan and ktantan – a family of small and efficient hardware-oriented block ciphers. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 272–288, Berlin, Heidelberg, 2009. Springer-Verlag.

[102] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt Robshaw. The led block cipher. In *Proceedings of the 13th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'11, pages 326–341, Berlin, Heidelberg, 2011. Springer-Verlag.

[103] Jong Hyuk Park. Security analysis of mcrypton proper to low-cost ubiquitous computing devices and applications. *Int. J. Communication Systems*, 22(8) :959–969, 2009.

[104] M.-J.O. Saarinen. The BlueJay Ultra-Lightweight Hybrid Cryptosystem. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, pages 27–32, May 2012.

[105] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *Algorithmic Number Theory : Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, chapter NTRU : A ring-based public key cryptosystem, pages 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[106] Jan Pelzl, Thomas Wollinger, Jorge Guajardo, and Christof Paar. *Cryptographic Hardware and Embedded Systems*, chapter Hyperelliptic Curve Cryptosystems : Closing the Performance Gap to Elliptic Curves, pages 351–365. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[107] Victor S Miller. Use of elliptic curves in cryptography. In *Lecture Notes in Computer Sciences ; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[108] Martin Hell, Thomas Johansson, and Willi Meier. Grain&#58 ; a stream cipher for constrained environments. *Int. J. Wire. Mob. Comput.*, 2(1) :86–93, May 2007.

[109] Steve Babbage and Matthew Dodd. *New Stream Cipher Designs : The eSTREAM Finalists*, chapter The MICKEY Stream Ciphers, pages 191–209. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[110] Christophe De Canniere and Bart Preneel. Trivium specifications. *eSTREAM, ECRYPT Stream Cipher Project*, 2006.

[111] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Assche. *Advances in Cryptology – EUROCRYPT 2013 : 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, chapter Keccak, pages 313–314. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[112] Jian Guo, Thomas Peyrin, and Axel Poschmann. *Advances in Cryptology – CRYPTO 2011 : 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, chapter The PHOTON Family of Lightweight Hash Functions, pages 222–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[113] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. *Cryptographic Hardware and Embedded Systems – CHES 2011 : 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings*, chapter spongent : A Lightweight Hash Function, pages 312–325. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[114] Jean-Philippe. Aumasson, Luca. Henzen, Willi. Meier, and María Naya-Plasencia. *Cryptographic Hardware and Embedded Systems, CHES 2010 : 12th International Workshop, Santa Barbara, USA, August 17-20, 2010. Proceedings*, chapter Quark : A Lightweight Hash, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[115] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. Lamed - a prng for epc class-1 generation-2 rfid specification. *Comput. Stand. Interfaces*, 31(1) :88–97, January 2009.

[116] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti. Multiple-polynomial LFSR based pseudorandom number generator for EPC Gen2 RFID tags. In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, pages 3820–3825, Nov 2011.

[117] Honorio Martín, Enrique San Millán, Luis Entrena, Julio César Hernández Castro, and Pedro Peris-Lopez. AKARI-X : A pseudorandom number generator for secure lightweight systems. In *17th IEEE International On-Line Testing Symposium (IOLTS 2011), 13-15 July, 2011, Athens, Greece*, pages 228–233, 2011.

[118] Kalikinkar Mandal, Xinxin Fan, and Guang Gong. Warbler : A lightweight pseudorandom number generator for EPC C1 Gen2 passive RFID tags. *International Journal of RFID Security and Cryptography*, 2(1) :82–91, December 2013.

[119] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A formal approach to distance-bounding rfid protocols. In *International Conference on Information Security*, pages 47–62. Springer, 2011.

[120] Adnan Abu-Mahfouz and Gerhard P Hancke. Distance bounding : A practical security solution for real-time location systems. *IEEE transactions on industrial informatics*, 9(1) :16–27, 2013.

[121] Yunhui Zhuang, Anjia Yang, Gerhard P Hancke, Duncan S Wong, and Guomin Yang. Energy-efficient distance-bounding with residual charge computation. *IEEE Transactions on Emerging Topics in Computing*, 2017.

[122] Pascal Urien and Selwyn Piramuthu. Elliptic curve-based rfid/nfc authentication with temperature sensor input for relay attacks. *Decision Support Systems*, 59 :28–36, 2014.

[123] Shingo Kinoshita, Miyako Ohkubo, Fumitaka Hoshino, Gembu Morohashi, Osamu Shionoiri, and Atsushi Kanai. Privacy enhanced active rfid tag. *Cognitive Science Research Paper-University of Sussex CSRP*, 577 :100, 2005.

[124] Piotr Szczechowiak and Martin Collier. Tinyibe : Identity-based encryption for heterogeneous sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pages 319–354. IEEE, 2009.

[125] Hsi-Wen Wang, Ren-Guey Lee, Chun-Chieh Hsiao, and Guan-Yu Hsieh. Active rfid system with cryptography and authentication mechanisms. *Journal of Information Science and Engineering*, 26(4) :1323–1344, 2010.

[126] Jiliang Zhou. Efficient and secure routing protocol based on encryption and authentication for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 9(4) :108968, 2013.

[127] Mohammad Fal Sadikin and Marcel Kyas. Rfid-tate : Efficient security and privacy protection for active rfid over ieee 802.15. 4. In *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, pages 335–340. IEEE, 2014.

[128] Yalin Chen and Jue-Sam Chou. Ecc-based untraceable authentication for large-scale active-tag rfid systems. *Electronic Commerce Research*, 15(1) :97–120, 2015.

[129] HC Leligou, P Karkazis, Th Zahariadis, T Velivasaki, Th Tsiodras, C Matrakidis, and S Voliotis. The impact of indirect trust information exchange on network performance and energy consumption in wireless sensor networks. In *ELMAR, 2011 Proceedings*, pages 153–156. IEEE, 2011.

[130] Han Yu, Zhiqi Shen, Chunyan Miao, Cyril Leung, and Dusit Niyato. A survey of trust and reputation management systems in wireless communications. *Proceedings of the IEEE*, 98(10) :1755–1772, 2010.

[131] Saurabh Ganeriwal, Laura K Balzano, and Mani B Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(3) :15, 2008.

[132] Riaz Ahmed Shaikh, Hassan Jameel, Brian J d'Auriol, Heejo Lee, Sungyoung Lee, and Young-Jae Song. Group-based trust management scheme for clustered wireless sensor networks. *IEEE transactions on parallel and distributed systems*, 20(11) :1698–1712, 2009.

[133] Xiaoyong Li, Feng Zhou, and Junping Du. Ldts : A lightweight and dependable trust system for clustered wireless sensor networks. *IEEE transactions on information forensics and security*, 8(6) :924–935, 2013.

[134] Félix Gómez Mármol and Gregorio Martínez Pérez. Providing trust in wireless sensor networks uses a bio-inspired technique. *Journal on communication*, pages 86–94, 2008.

[135] Haiguang Chen, Huafeng Wu, Xi Zhou, and Chuanshan Gao. Agent-based trust model in wireless sensor networks. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 3, pages 119–124. IEEE, 2007.

[136] Chiu C Tan, Bo Sheng, and Qun Li. Severless search and authentication protocols for rfid. In *Pervasive Computing and Communications, 2007. PerCom'07. Fifth Annual IEEE International Conference on*, pages 3–12. IEEE, 2007.

[137] Ji Young Chun, Jung Yeon Hwang, and Dong Hoon Lee. Rfid tag search protocol preserving privacy of mobile reader holders. *IEICE Electronics Express*, 8(2) :50–56, 2011.

[138] Masoumeh Safkhani, Pedro Peris-Lopez, Nasour Bagheri, Majid Naderi, and Julio César Hernández Castro. On the security of tan et al. serverless rfid authentication and search protocols. *RFIDSec*, 7739 :1–19, 2012.

[139] Luon-Chang Lin, Shih-Chang Tsaur, and Kai-Ping Chang. Lightweight and serverless rfid authentication and search protocol. In *Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on*, volume 2, pages 95–99. IEEE, 2009.

[140] Chin-Feng Lee, Hung-Yu Chien, and Chi-Sung Laih. Server-less rfid authentication and searching protocol with enhanced security. *International Journal of Communication Systems*, 25(3) :376–385, 2012.

[141] Jialiang He, Youjun Xu, and Zhiqiang Xu. Secure and private protocols for serverless rfid systems. *International Journal of Control and Automation*, 7 :131–142, 02 2014.

[142] Md Endadul Hoque, Farzana Rahman, Sheikh I Ahamed, and Jong Hyuk Park. Enhancing privacy and security of rfid system with serverless authentication and search protocols in pervasive environments. *Wireless personal communications*, 55(1) :65–79, 2010.

[143] Miaolei Deng, Weidong Yang, and Weijun Zhu. Weakness in a serverless authentication protocol for radio frequency identification. In *Mechatronics and Automatic Control Systems*, pages 1055–1061. Springer, 2014.

[144] Mohsen Pourpouneh, Rasoul Ramezanian, and Fatemeh Salahi. An improvement over a server-less rfid authentication protocol. *International Journal of Computer Network and Information Security*, 7(1) :31, 2014.

[145] Shahab Abdolmaleky, Shahla Atapoor, Mohammad Hajighasemlou, and Hamid Sharini. A strengthened version of a hash-based rfid server-less security scheme. *Advances in Computer Science : an International Journal*, 4(3) :18–23, 2015.

[146] Sheikh Iqbal Ahamed, Farzana Rahman, and Endadul Hoque. Erap : Ecc based rfid authentication protocol. In *Future Trends of Distributed Computing Systems, 2008. FTDCS'08. 12th IEEE International Workshop on*, pages 219–225. IEEE, 2008.

[147] Parikshit N Mahalle, Bayu Anggorojati, Neeli Rashmi Prasad, and Ramjee Prasad. Identity establishment and capability based access control (iecac) scheme for internet of things. In *Wireless personal multimedia communications (WPMC), 2012 15th international symposium on*, pages 187–191. IEEE, 2012.

[148] Gene Tsudik. Ya-trap : Yet another trivial rfid authentication protocol. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 4–pp. IEEE, 2006.

[149] Jongho Won, Seung-Hyun Seo, and Elisa Bertino. A secure communication protocol for drones and smart objects. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 249–260. ACM, 2015.

[150] R. N. Akram, K. Markantonakis, K. Mayes, P. Bonnefoi, D. Sauveron, and S. Chaumette. A secure and trusted protocol for enhancing safety of on-ground airplanes

using uavs. In *2017 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 1–17, April 2017.

[151] Li Zhou, Jinfeng Ni, and Chinya V Ravishankar. Supporting secure communication and data collection in mobile sensor networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12. IEEE, 2006.

[152] Collins Mtita, Maryline Laurent, Damien Sauveron, Raja Naeem Akram, Konstantinos Markantonakis, and Serge Chaumette. Serverless protocols for inventory and tracking with a UAV. *CoRR*, abs/1708.05417, 2017.

[153] Collins Mtita, Maryline Laurent, and Jacques Delort. Efficient serverless radio-frequency identification mutual authentication and secure tag search protocols with untrusted readers. *IET Information Security*, 10(5) :262–271, 2016.

[154] AS Poornima and BB Amberker. Secure data collection using mobile data collector in clustered wireless sensor networks. *IET wireless sensor systems*, 1(2) :85–95, 2011.

[155] Matteo Avalle, Alfredo Pironti, and Riccardo Sisto. Formal verification of security protocol implementations : a survey. *Formal Asp. Comput.*, 26(1) :99–123, 2014.

[156] Bruno Blanchet. Security protocol verification : Symbolic and computational models. In *Proceedings of the First international conference on Principles of Security and Trust*, pages 3–29. Springer-Verlag, 2012.

[157] Catherine Meadows. Formal methods for cryptographic protocol analysis : emerging issues and trends. *IEEE Journal on Selected Areas in Communications*, 21(1) :44–54, 2003.

[158] Cas Cremers and Sjouke Mauw. *Operational Semantics and Verification of Security Protocols*. Springer Publishing Company, Incorporated, 2012.

[159] Katharina Pfeffer. Formal Verification of a LTE Security Protocol for Dual-Connectivity An Evaluation of Automatic Model Checking Tools , 2014.

[160] D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society.

[161] Jean E. Martina. Verification of security protocols based on multicast communication, 2012.

[162] Juan C. Pimentel and Raúl Monroy. Formal Support to Security Protocol Development : A Survey. *Computacion y Sistemas*, 12(1) :89–108, 2008.

[163] David Monniaux. Analysis of cryptographic protocols using logics of belief : an overview, 2002.

[164] Michael Burrows, Martin Abadi, and Roger Michael Needham. A logic of authentication. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 426(1871) :233–271, 1989.

[165] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8 :18–36, 1990.

[166] Lu Ma and Jeffrey J. P. Tsai. Formal Verification Techniques For Computer Communication Security Protocols, 2001.

[167] Li Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 234–248, May 1990.

[168] Stephen H. Brackin. A HOL extension of GNY for automatically analyzing cryptographic protocols. In *Ninth IEEE Computer Security Foundations Workshop, March 10 - 12, 1996, Dromquinna Manor, Kenmare, County Kerry, Ireland*, page 62, 1996.

[169] Jan Wessels. Application of ban-logic, 2001.

[170] Mariskha Tri Adithia. The difference between the ban logic and the gny logic, 2012.

[171] Kieran Healy, Tom Coffey, and Reiner Dojen. A Comparative Analysis of State-Space Tools for Security Protocol Verification. In *WSEAS Transactions on Information Science and Applications, Volume 1, Issue 5*, pages 1256–1261, 2004.

[172] Gavin Lowe. Casper : A compiler for the analysis of security protocols. *Journal of Computer Security*, 6(1-2) :53–84, 1998.

[173] David L. Dill, Andreas J. Drexler, Alan J. Hu, and C. Han Yang. Protocol Verification as a Hardware Design Aid. In *In IEEE International Conference on Computer Design : VLSI in Computers and Processors*, pages 522–525. IEEE Computer Society, 1992.

[174] E. M. Clarke, S. Jha, and W. Marrero. Verifying Security Protocols with Brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4) :443–487, October 2000.

[175] Mathieu Turuani. *Term Rewriting and Applications : 17th International Conference, RTA 2006 Seattle, WA, USA, August 12-14, 2006 Proceedings*, chapter The CL-Atse Protocol Analyser, pages 277–286. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[176] David Basin, Sebastian Mödersheim, and Luca Viganò. OFMC : A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3) :181–208, 2004.

[177] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In *Proceedings of the 17th International Conference on Computer Aided Verification*, CAV'05, pages 281–285, Berlin, Heidelberg, 2005. Springer-Verlag.

[178] Casimier J. Cremers. *Scyther - Semantics and Verification of Security Protocols.* PhD thesis, Eindhoven University of Technology, 2006.

[179] Dawn Xiaodong Song, Sergey Berezin, and Adrian Perrig. Athena : A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1/2) :47–74, 2001.

[180] F.J Thayer . Fabrega, J.C. Herzog, and J.D. Guttman. Strand spaces : why is a security protocol correct ? In *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, pages 160–171, May 1998.

[181] Bruno Blanchet. *Foundations of Security Analysis and Design VII : FOSAD 2012/2013 Tutorial Lectures*, chapter Automatic Verification of Security Protocols in the Symbolic Model : The Verifier ProVerif, pages 54–87. Springer International Publishing, Cham, 2014.

[182] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 696–701. Springer, 2013.

[183] A.K. Mohtaram. *Classification, Formalization and Automatic Verification of Untraceability in RFID Protocols.* PhD thesis, École Polytechnique de Montréal, http ://publications.polymtl.ca/1094/, JUL 2013.

[184] Catherine Meadows. The {NRL} protocol analyzer : An overview. *The Journal of Logic Programming*, 26(2) :113 – 131, 1996.

[185] Graham Steel, Alan Bundy, and Monika Maidl. Attacking the Asokan–Ginzboorg Protocol for Key Distribution in an Ad-Hoc Bluetooth Network Using CORAL, Oct 2003.

[186] Lawrence C. Paulson. *Isabelle : a generic theorem prover*. Lecture notes in computer science. Springer-Verlag, Berlin, New York, 1994.

[187] Raja Naeem Akram, Konstantinos Markantonakis, Keith Mayes, Pierre-François Bonnefoi, Amina Cherif, Damien Sauveron, and Serge Chaumette. A secure and trusted channel protocol for uavs fleets. In Gerhard P. Hancke and Ernesto Damiani, editors, *Information Security Theory and Practice*, pages 3–24, Cham, 2018. Springer International Publishing.

[188] Cita Furlani. FIPS 186-3 : Digital Signature Standard (DSS). Online, June 2009.

[189] Edmund M. Clarke. *The Birth of Model Checking*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[190] Amina Cherif, Malika Belkadi, and Damien Sauveron. A lightweight and secure data collection serverless protocol demonstrated in an active rfids scenario. *ACM Trans. Embed. Comput. Syst.*, 18(3) :27 :1–27 :27, April 2019.

[191] Oracle Inc. Java Card Technology. https://www.oracle.com/technetwork/java/embedded/javacard/overview/index.html. Online ; accessed 20 May 2019.

[192] Patrick Van haver. Unveiling Java Card 3.1 : New I/O and Trusted Peripherals. https://blogs.oracle.com/javaiot/unveiling-java-card-31%3a-new-io-and-trusted-peripherals, November 2018. Online ; accessed 20 May 2019.

[193] Oracle Inc. Oracle Java Card Boosts Security for IoT Devices at the Edge. https://www.oracle.com/corporate/pressrelease/oracle-java-card-boosts-security-011619.html, January 2019. Online ; accessed 20 May 2019.

[194] Oracle Inc. Java Card 3.1 Documentation. https://docs.oracle.com/en/java/javacard/3.1/, January 2019. Online ; accessed 20 May 2019.

[195] Oracle Inc. Java Card Development Kit. https://www.oracle.com/technetwork/java/embedded/javacard/downloads/javacard-sdk-2043229.html. Online ; accessed 20 May 2019.

[196] Pierre Paradinas, Julien Cordry, and Samia Bouzefrane. Performance evaluation of java card bytecodes. In Damien Sauveron, Konstantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, pages 127–137, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[197] CNAM-CEDRIC. The mesure project. http://http://mesure.gforge.inria.fr, 2006. Accessed : 2018-06-16.

[198] Petr Švenda. The jcalgtest project. http://http://mesure.gforge.inria.fr, 2018. Accessed : 2018-06-16.

Annexes

.

A Vérification formelle du protocole de collecte sécurisée des données

A.1 AVISPA

A.1.1 Spécification du script HLPSL du protocole

```
role drone (MDC, ARJ: agent,
            Acki: symmetric_key,
            Ti: message,
            Hk: hash_func,
            SND, RCV: channel(dy))
played_by MDC
def=
 local State: nat,
       Hmac: message,
       SDi, Request: text,
```

```
       Ki          : symmetric_key
 const sec_k, auth_n, auth_d: protocol_id

 init State:=0
transition
1.   State=0 /\ RCV(start)                  =|> State':=2 /\ SND(Request)
2.   State=2 /\ RCV(ARJ.{SDi'}_Ki'.Hmac')
             /\ Hmac'={Hk({SDi'}_Ki')}_Acki =|> State':=4 /\ SND(Ti)
                                                          /\ secret(SDi', sec_k, {ARJ, MDC})
                                                          /\ witness(MDC, ARJ, auth_d, Ti)
                                                          /\ request(MDC, ARJ, auth_n, Hmac')
end role

role node (ARJ, MDC: agent,
           Ki: symmetric_key,
           Ti: message,
           Hk: hash_func,
           SND, RCV: channel(dy))
played_by ARJ
def=
 local State: nat,
       Hmac: message,
       SDi, Request: text
 const sec_k, auth_n, auth_d: protocol_id

 init State:=1
transition
1.   State=1 /\ RCV(Request') =|> State':=3 /\ SDi':= new() /\ SND(ARJ.{SDi'}_Ki.Hmac)
                                                          /\ secret(SDi', sec_k, {ARJ, MDC})
                                                          /\ witness(ARJ, MDC, auth_n, Hmac)
```

```
2.   State=3 /\ RCV(Ti')
             /\ Ti = Ti'      =|> State':=5 /\ request(ARJ, MDC, auth_d, Ti')
end role

role session (MDC,ARJ: agent,
              Acki, Ki: symmetric_key,
              Ti: message,
              Hk: hash_func)
def=
 local SN, RN, SD, RD: channel(dy)
 composition
       drone (MDC, ARJ, Acki, Ti, Hk, SD, RD)
    /\ node  (ARJ, MDC, Ki,Ti,Hk, SN, RN)
end role

role environment( )
def=
  const arj, mdc: agent,
        hk      : hash_func,
        t: message,
        ki, acki: symmetric_key,
 sec_k, auth_n, auth_d: protocol_id
 intruder_knowledge = {arj, mdc, i, hk}

 composition
     session (mdc, arj, acki, ki, t, hk)
  /\ session (arj, i, acki, ki, t, hk)
  /\ session (i, mdc, acki, ki, t, hk)
end role
```

```
goal
  secrecy_of sec_k
  authentication_on auth_d, auth_n
end goal

environment ()
```

A.1.2 Résultats

Le résultat du back-end OFMC est :

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/avispaVerif.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.02s
  visitedNodes: 4 nodes
  depth: 2 plies
```

Le résultat du back-end CL-AtSe est :

```
SUMMARY

  SAFE

DETAILS

  BOUNDED_NUMBER_OF_SESSIONS

  TYPED_MODEL

PROTOCOL

  /home/span/span/testsuite/results/avispaVerif.if

GOAL

  As Specified

BACKEND

  CL-AtSe

STATISTICS

  Analysed   : 6 states

  Reachable  : 1 states

  Translation: 0.01 seconds

  Computation: 0.00 seconds
```

A.2 ProVerif

A.2.1 Spécification du script pi calculus appliqué du protocole

```
(* ServerlessProtocol *)

type skey.

type ticket.
```

```
free ch: channel.

free request: bitstring.
free TI: ticket [private].
free acki: skey [private].
free sk: skey [private].
free HMAC: bitstring [private].
fun hk(bitstring): bitstring.
fun enc(bitstring, skey): bitstring.

const ID: bitstring.

(* Queries *)

(* verify the secret *)

free SDI: bitstring [private].

query attacker(SDI).

(* verify authentication *)

event beginARJ(bitstring).
event endARJ(ticket).
event beginMDC(bitstring).
event endMDC(ticket).

query x: bitstring; inj-event(beginMDC(x)) ==> inj-event(beginARJ(x)).
query x: ticket; inj-event(endARJ(x)) ==> inj-event(endMDC(x)).
query x: bitstring, y:ticket; inj-event(endARJ(y)) ==> inj-event(beginARJ(x)) &&
```

```
                                                    inj-event(beginMDC(x)).


(* Role of the Mobile Data Collector *)

let MDC (ID: bitstring, TI: ticket, acki: skey)=
out(ch, request);
in(ch, (m2: bitstring, m3: bitstring, m4: bitstring));
if ID = m2 then
let HMACD = enc(hk(m3), acki) in
if HMACD = m4 then
event beginMDC(HMACD);
out(ch, TI);
event endMDC(TI).


(* Role of the Active RFID node *)

let ARJ (ID: bitstring, HMAC: bitstring, TI: ticket, SDI: bitstring)=
in(ch, m1: bitstring);
event beginARJ(HMAC);
out(ch, (ID, enc(SDI, sk), HMAC));
in(ch, m5: ticket);
if TI = m5 then
event endARJ(TI).


(* Start process *)

process
```

```
    (
          (* Launch an unbounded number of sessions of the MDC AND ARJ *)
          (!MDC(ID, TI, acki)) | (!ARJ(ID, HMAC, TI, SDI))

    )
```

A.2.2 Résultats

```
-- Query not attacker(SDI[])
Completing...
Starting query not attacker(SDI[])
RESULT not attacker(SDI[]) is true.
-- Query inj-event(beginMDC(x)) ==> inj-event(beginARJ(x))
Completing...
Starting query inj-event(beginMDC(x)) ==> inj-event(beginARJ(x))
RESULT inj-event(beginMDC(x)) ==> inj-event(beginARJ(x)) is true.
-- Query inj-event(endARJ(x_10)) ==> inj-event(endMDC(x_10))
Completing...
Starting query inj-event(endARJ(x_10)) ==> inj-event(endMDC(x_10))
RESULT inj-event(endARJ(x_10)) ==> inj-event(endMDC(x_10)) is true.
-- Query inj-event(endARJ(y)) ==> (inj-event(beginARJ(x_11)) && inj-event(beginMDC(x_11)))
Completing...
Starting query inj-event(endARJ(y)) ==> (inj-event(beginARJ(x_11)) &&
                                         inj-event(beginMDC(x_11)))
RESULT inj-event(endARJ(y)) ==> (inj-event(beginARJ(x_11)) &&
```

B Vérification formelle du protocole STCP

B.1 Casper/FDR Script

```
#Free variables

datatype Field = Gen | Exp(Field, Num) unwinding 2

hkSE2, hkSE1, iMsg, rMsg, EnMaKey : Field

SE1, SE2, U: Agent

gSE1, gSE2: Num

nSE1, nSE2, SE1Val, SE2Val: Nonce

VKey: Agent->PublicKey

SKey: Agent->SecretKey

InverseKeys = (VKey, SKey), (EnMaKey, EnMaKey), (Gen, Gen), (Exp, Exp)

#Protocol description

0.    -> SE2 : SE1 [SE1!=SE2] <iMsg := Exp(Gen,gSE2)>

1. SE2 -> SE1 : SE2, nSE2, iMsg%hkSE2 <EnMaKey := Exp(hkSE2, gSE1); rMsg := Exp(Gen,gSE1)>

2. SE1 -> SE2 : nSE1, rMsg%hkSE1 <EnMaKey := Exp(hkSE1, gSE2)>

3. SE2 -> SE1 : nSE2, nSE1

4. SE1 -> SE2 : {{rMsg, U, nSE2}{SKey(U)}}{EnMaKey} [rMsg==hkSE2]

5. SE2 -> SE1 : {{iMsg,SE2, nSE1}{SKey(SE2)}}{EnMaKey} [iMsg==hkSE1]

6. SE1 -> SE2 : {{SE1OSHash, SE1, nSE2}{SKey(SE1)}}{EnMaKey}

#Actual variables

ADev1, ADev2, ME: Agent

GSE1, GSE2, GMalicious: Num

NSE1, NSE2, SE1VAL, SE2VAL, NMalicious: Nonce

#Processes

INITIATOR(SE2, SE1, U, SE2VAL, gSE2, nSE2) knows SKey(SE2), VKey

RESPONDER(SE1, SE2, U, SE1VAL, gSC, nSC) knows SKey(U), SKey(SC), VKey

#System
```

```
INITIATOR(ADev2, ADev1, ADev2Val, GSE2, NSE2)
RESPONDER(ADev1, ADev2, ADev1Val, GSE1, NSE1)

#Functions
symbolic VKey, SKey

#Intruder Information
Intruder = ME
IntruderKnowledge = {ADev2, ADev2, ME,
GMalicious, NMalicious, SKey(ME), VKey}

#Specification
Aliveness(SE2, SE1)
Aliveness(SE1, SE2)
Agreement(SE2, SE1, [EnMaKey])
Secret(SE2, EnMaKey, [SE1])
Secret(SE1, U, [SE2])

#Equivalences
forall x, y : Num . Exp(Exp(Gen, x), y) = Exp(Exp(Gen, y), x)
```

B.2 AVISPA Script

```
role   se_1 (A,B: agent,
             G : text,
       PK1,PKTM1: public_key,
     CSE1,CTMSE1: message,
 H,Hk, MAC, SIGN: hash_func,
       SND, RCV : channel(dy))


played_by A
     def=
         local   NS1, NS2, Rs1, Rs2          : text,
```

```
                State                       : nat,
                VR1, VR2                    : text,
                SAS1, SAS2, Sdata1, Sdata2  : text,
                Success                     : text,
                Svalid1, Svalid2            : text.text.text,
                Kdh, Ke, Ka                 : message,
                PK2, PKTM2                  : public_key,
                Scookie                     : message,
                CSE2, CTMSE2                : message

        const   sec_kdh1, sec_ke1, sec_ka1  : protocol_id
        init    State:= 0

transition
1.   State=0 /\ RCV(start) =|>
                State':=2 /\ NS1':= new() /\ Rs1' := new()
                          /\  SND(A.B.NS1'.exp(G, Rs1').VR1.Scookie)

2.   State=2 /\ RCV(B.A.NS2'.exp(G, Rs2').Sdata2'.Svalid2'.
             {SIGN(Sdata2')}_(inv(PK2')).{SIGN(Svalid2')}_(inv(PKTM2')).
             MAC(Ka'.{{SIGN(Sdata2')}_(inv(PK2')).
             {SIGN(Svalid2')}_(inv(PKTM2')).CSE2'.CTMSE2'}_Ke')
             .VR2.Scookie) =|>

     State':= 4   /\  SAS1':= new()
                  /\ Svalid1':= SAS1'.NS2'.NS1
                  /\  Sdata1':= H(A.B.exp(G, Rs2').exp(G, Rs1).NS2'.NS1)
                  /\ Kdh' := exp(exp(G, Rs2'), Rs1)
                  /\ Ke':= {Hk(NS1.NS2'.1)}_Kdh'
                  /\ Ka':= {Hk(NS1.NS2'.2)}_Kdh'
```

```
                /\ SND(A.B.Sdata1'.Svalid1'.{SIGN(Sdata1')}_(inv(PK1)).
                   {SIGN(Svalid1')}_(inv(PKTM1)).MAC(Ka'.
                   {{SIGN(Sdata1')}_(inv(PK1)).
                   {SIGN(Svalid1')}_(inv(PKTM1)).CSE1.CTMSE1}_Ke').Scookie)
                /\ witness ( A,B, ns1, NS1)

3. State= 4 /\ RCV(Success') =|>
             State':=6       /\ request (A,B, ns2, NS2)
                             /\ secret (Kdh,sec_kdh1, {A,B})
                             /\ secret(Ke, sec_ke1, {A,B})
                             /\ secret(Ka, sec_ka1, {A,B})

end role



role   se_2 (B,A: agent,
             G : text,
       PK2,PKTM2: public_key,
     CSE2,CTMSE2: message,
     H,Hk, MAC, SIGN: hash_func,
           SND, RCV : channel(dy))


played_by B
     def=
        local   NS2, NS1, Rs2, Rs1             : text,
        State                                  : nat,
          VR2, VR1                             : text,
          SAS2, SAS1, Sdata2, Sdata1           : text,
          Success                              : text,
          Svalid2, Svalid1                     : text.text.text,
```

```
          Kdh, Ke, Ka                          : message,
          PK1, PKTM1                           : public_key,
          Scookie                              : message,
          CSE1, CTMSE1                         : message
          const     sec_kdh2, sec_ke2, sec_ka2 : protocol_id

          init      State:= 1

transition
1.          State=1 /\ RCV(A.B.NS1'.exp(G, Rs1'). VR1.Scookie) =|>
            State':= 3 /\ NS2':= new()    /\ Rs2':=new()   /\  SAS2':= new()
                       /\ Sdata2':= H(B.A.exp(G, Rs1').exp(G, Rs2'). NS1'.NS2')
                       /\  Svalid2':= SAS2'.NS1'.NS2'
                       /\   Kdh' := exp(exp(G, Rs1'), Rs2')
                       /\ Ke' := {Hk(NS1.NS2'.1)}_Kdh'
                       /\ Ka' := {Hk(NS1.NS2'.2)}_Kdh'
                       /\ SND (B.A.NS2'.exp(G, Rs2'). Sdata2'.Svalid2'.
                       {SIGN(Sdata2')}_(inv(PK2)). {SIGN(Svalid2')}_(inv(PKTM2))
                       .MAC(Ka'.{{SIGN(Sdata2')}_(inv(PK2)).
                       {SIGN(Svalid2')}_(inv(PKTM2)).CSE2.CTMSE2}_Ke').VR2.Scookie)
                       /\ witness (B,A, ns2, NS2')

2.        State=3 /\ RCV  (A.B.Sdata1'.Svalid1'.{SIGN(Sdata1')}_(inv(PK1')).
                  {SIGN(Svalid1')}_(inv(PKTM1')).MAC(Ka.{{SIGN(Sdata1')}_(inv(PK1')).
                  {SIGN(Svalid1')}_(inv(PKTM1')).CSE1'.CTMSE1'}_Ke).Scookie)  =|>
             State':= 5 /\  request(B,A,ns1, NS1)
                        /\ SND(Success')
                        /\ secret (Kdh,sec_kdh2, {B,A})
                        /\ secret(Ke, sec_ke2, {B,A})
                        /\ secret(Ka, sec_ka2, {B,A})
```

```
end role

role session (A,B: agent,
              G: text,
          H,Hk, MAC, SIGN: hash_func)
def=
    local         SA, RA, SB, RB :     channel(dy),
       CSE1, CSE2, CTMSE1, CTMSE2: message,
           PK1, PK2, PKTM1, PKTM2:  public_key

    composition
    se_1(A,B, G, PK1, PKTM1, CSE1, CTMSE1, H,Hk, MAC, SIGN, SA, RA)
       /\    se_2(B, A, G, PK2, PKTM2, CSE2, CTMSE2, H, Hk, MAC, SIGN, SB, RB)
end role

role environment( )  def=

         const   ns1, ns2                : protocol_id,
                 a, b                    : agent,
                 pk1, pk2, pki           : public_key,
                 g                       : text,
                 h, hk, mac, sign        : hash_func

         intruder_knowledge = {a, b, i, pk1, pk2, pki, inv(pki), g, h, hk, mac, sign}

         composition
                    session (a ,b, g, h, hk, mac, sign)
                /\  session (a ,i , g, h, hk, mac, sign)
                /\  session (i ,b , g,  h, hk, mac, sign)
end role
```

```
goal
secrecy_of    sec_kdh1,   sec_kdh2 , sec_ke1,    sec_ke2,    sec_ka1,    sec_ka2
authentication_on    ns1
authentication_on    ns2
end goal

environment ()
```

C Applet implémentant les opérations du protocole sécurisé de collecte des données

C.1 Code de l'applet

```
package fr.cryptis.research.securedatacollection;

import javacard.framework.Applet;
import javacard.framework.ISOException;
import javacard.framework.ISO7816;
import javacard.framework.APDU;
import javacard.framework.Util;
import javacard.security.AESKey;
import javacard.security.KeyBuilder;
import javacard.security.MessageDigest;
import javacard.security.Signature;
import javacardx.crypto.Cipher;


public class SecureDataCollection extends Applet {


    private final static byte APP_CLA = (byte)0x00;

    private final static byte INIT_ID = (byte)0x08;
    private final static byte INIT_KEY = (byte)0x10;
    private final static byte RECEIVE_SENSED_DATA = (byte)0x12;
    private final static byte SEND_CIPHERED_DATA = (byte)0x14;
    private final static byte RECEIVED_CIPHERED_DATA = (byte)0x16; // FOR DRONE
    private final static byte COMPUTED_CREDENTIALS = (byte)0x18; // FOR BS
    private final static byte GET_COMPUTED_CREDENTIALS = (byte)0x1A; // FOR BS



    private static byte KEY_SIZE;
    private static byte TICKET_SIZE;
    private static byte ACK_SIZE;
```

```
private static byte HMAC_SIZE;

// for performance
private final static byte DO_NOTHING = (byte)0x40;
private final static byte CIPHERED_DATA = (byte)0x42;
private final static byte CREATE_TICKET_ACK = (byte)0x44;
private final static byte CREATE_NEW_KEY = (byte)0x46;
private final static byte CREATE_NEW_HMAC = (byte)0x48;


// for debug
private final static byte RECEIVED_POS_TO_INSERT = (byte)0x32;
private final static byte RECEIVED_INDEX = (byte)0x34;
private final static byte RECEIVED_DATA_STORAGE = (byte)0x36;
private final static byte RECEIVED_IDJ = (byte)0x30;
private final static byte RECEIVED_KJ = (byte)0x38;
private final static byte RECEIVED_ACKJ = (byte)0x3A;


private AESKey IDj; // Change to HMACKey
private AESKey Kj;
private AESKey ACKj; // Change to HMACKey
private Signature hmac;
private MessageDigest hash;
private Cipher cipher;
private byte[] datastorage;
private byte index = 0;
private short posToInsert = 0;

private AESKey IDjB;
private AESKey KjB;
private byte[] credentialStorage;

public SecureDataCollection() {
    IDj = (AESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_AES, KeyBuilder.LENGTH_AES_128, false);
    Kj = (AESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_AES, KeyBuilder.LENGTH_AES_128, false);
    ACKj = (AESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_AES, KeyBuilder.LENGTH_AES_128, false);
    KEY_SIZE = (byte)(KeyBuilder.LENGTH_AES_128>>3);
```

```
        hmac = Signature.getInstance(Signature.ALG_AES_MAC_128_NOPAD, false);
        hash = MessageDigest.getInstance(MessageDigest.ALG_SHA, false);
        cipher = Cipher.getInstance(Cipher.ALG_AES_BLOCK_128_CBC_NOPAD, false);
        datastorage = new byte[(short)0x1000];
        //TICKET_SIZE  = (byte)0x20; //FOR SHA256
        TICKET_SIZE  = (byte)0x14; //FOR SHA1
        ACK_SIZE  = (byte)(ACKj.getSize()>>3);
        HMAC_SIZE = (byte) (128>>3);

        // For BS
        IDjB = (AESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_AES, KeyBuilder.LENGTH_AES_128, false);
        KjB = (AESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_AES, KeyBuilder.LENGTH_AES_128, false);
        credentialStorage = new byte[(short)((KEY_SIZE+TICKET_SIZE+ACK_SIZE)<<3)];
        //We store keys for deciphering * 8
    }

    public static void install(byte[] bArray, short bOffset, byte bLength) {
        // GP-compliant JavaCard applet registration
        new SecureDataCollection().register(bArray, (short) (bOffset + 1),
                bArray[bOffset]);
    }

    public void process(APDU apdu) {
        // Good practice: Return 9000 on SELECT
        if (selectingApplet()) {
            return;
        }

        byte[] buf = apdu.getBuffer();

        if (buf[ISO7816.OFFSET_CLA] != APP_CLA)
            ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);

        short of = 0;
        short len = 0;

        switch (buf[ISO7816.OFFSET_INS]) {
        case INIT_ID:
```

```
        len = apdu.setIncomingAndReceive();
        if (len!=KEY_SIZE)
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        IDj.setKey(buf, ISO7816.OFFSET_CDATA);
        break;
    case INIT_KEY:
        len = apdu.setIncomingAndReceive();
        if (len!=(short)(KEY_SIZE))
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        Kj.setKey(buf, ISO7816.OFFSET_CDATA);
        break;
    case RECEIVE_SENSED_DATA:
        len = apdu.setIncomingAndReceive();

        // Set the right Kj
        cipher.init(Kj, Cipher.MODE_ENCRYPT);

        // Cipher and store sensed data
        datastorage[posToInsert] = (byte)((len+TICKET_SIZE+ACK_SIZE)&0x00ff);
        posToInsert++;
        posToInsert += cipher.doFinal(buf, ISO7816.OFFSET_CDATA, len, datastorage, posToInsert);

        // Compute Tj
        Kj.getKey(buf, of);
        buf[KEY_SIZE] = (byte)'1';
        hash.reset();
        hash.doFinal(buf, of, (short) (KEY_SIZE+1), datastorage, posToInsert);
        posToInsert += TICKET_SIZE;

        // Compute ACKj
        buf[KEY_SIZE] = (byte)'2';
        hash.reset();
        hash.doFinal(buf, of, (short) (KEY_SIZE+1), buf, (short) (KEY_SIZE+1));
        Util.arrayCopyNonAtomic(buf, (short) (KEY_SIZE+1), datastorage, posToInsert, ACK_SIZE);
        posToInsert += ACK_SIZE;

        // Generate new Kj
        hmac.init(IDj, Signature.MODE_SIGN);
```

```
        hmac.sign(buf, of, KEY_SIZE, buf, KEY_SIZE);
        Kj.setKey(buf, KEY_SIZE);
        break;
    case SEND_CIPHERED_DATA:
        len = (short) (buf[ISO7816.OFFSET_P1] & 0x00ff);
        if (len == 0) {
            len = apdu.setIncomingAndReceive();
            len = (short) (datastorage[of] & 0x00ff);
            if(len == 0)
                ISOException.throwIt((short)0x6666); // no more data
            // Check the ticket sent by drone
            if (Util.arrayCompare(buf, ISO7816.OFFSET_CDATA, datastorage,
                                  (short) (len-TICKET_SIZE-ACK_SIZE+1), TICKET_SIZE)!=0)
                ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);

            Util.arrayCopyNonAtomic(datastorage, (short)(len+1),
                                    datastorage, (short)0, (short) (posToInsert-len));
            Util.arrayFillNonAtomic(datastorage,(short) (posToInsert-len), len, (byte)0);
            posToInsert -= (short)(len+1);
        }
        // There might be not data after deletion of previous data, so we need to keep it.
        len = (short) (datastorage[of] & 0x00ff);
        if(len == 0)
            ISOException.throwIt((short)0x6666); // no more data
        apdu.setOutgoing();
        apdu.setOutgoingLength(len);
        of = IDj.getKey(buf, (short) 0);
        apdu.sendBytes((short) 0, of);
        apdu.sendBytesLong(datastorage, (short) 1, (short) (len-TICKET_SIZE-ACK_SIZE));

        // Get ACKj
        ACKj.setKey(datastorage, (short) (len-ACK_SIZE+1));

        // Generate new HMAC
        hmac.init(ACKj, Signature.MODE_SIGN);
        of = hmac.sign(datastorage, (short) 1, (short) (len-TICKET_SIZE-ACK_SIZE), buf, (short) 0);
        apdu.sendBytes((short) 0, of);
        break;
```

```
case COMPUTED_CREDENTIALS:
    len = apdu.setIncomingAndReceive();
    if (len!=KEY_SIZE<<1)
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    IDjB.setKey(buf, ISO7816.OFFSET_CDATA);
    KjB.setKey(buf, (short)(ISO7816.OFFSET_CDATA+KEY_SIZE));

    len = 0;
    for (short i = 0; i < 8; i++){
        // Compute Tj
        KjB.getKey(buf, of);
        buf[KEY_SIZE] = (byte)'1';
        hash.reset();
        hash.doFinal(buf, of, (short) (KEY_SIZE+1), credentialStorage, (short) (len+KEY_SIZE));

        // Compute ACKj
        buf[KEY_SIZE] = (byte)'2';
        hash.reset();
        hash.doFinal(buf, of, (short) (KEY_SIZE+1), buf, (short) (KEY_SIZE+1));
        Util.arrayCopyNonAtomic(buf, (short) 0, credentialStorage, (short)len, (short)(KEY_SIZE));
        Util.arrayCopyNonAtomic(buf, (short) (KEY_SIZE+1), credentialStorage,
                                (short)(len+KEY_SIZE+TICKET_SIZE), (short)(ACK_SIZE));
        len += (short) (KEY_SIZE + TICKET_SIZE + ACK_SIZE);

        // Generate new Kj
        hmac.init(IDjB, Signature.MODE_SIGN);
        hmac.sign(buf, of, KEY_SIZE, buf, KEY_SIZE);
        KjB.setKey(buf, KEY_SIZE);
    }
    break;
case GET_COMPUTED_CREDENTIALS:
    len = (short)(KEY_SIZE + TICKET_SIZE + ACK_SIZE);
    of = (short) ((buf[ISO7816.OFFSET_P1] & 0x00ff) * len);
    apdu.setOutgoing();
    apdu.setOutgoingLength(len);
    apdu.sendBytesLong(credentialStorage, of, len);
    break;
```

```
// FOR PERFORMANCE
case DO_NOTHING:
    of = Util.getShort(buf, ISO7816.OFFSET_P1);
    for (short i=0; i < of; i++)
    {
        ;
    }
    break;

case CIPHERED_DATA:
    of = Util.getShort(buf, ISO7816.OFFSET_P1);
    len = (short)(buf[ISO7816.OFFSET_LC]&0x00ff);
    for (short i=0; i < of; i++)
    {
        // Set the right Kj
        cipher.init(Kj, Cipher.MODE_ENCRYPT);

        // Cipher and store sensed data
        cipher.doFinal(buf, ISO7816.OFFSET_CDATA, len, buf, (short)(ISO7816.OFFSET_CDATA+len));
    }
    break;
case CREATE_TICKET_ACK:
    of = Util.getShort(buf, ISO7816.OFFSET_P1);
    for (short i=0; i < of; i++)
    {
        hash.reset();
        hash.doFinal(buf, (short)0, (short) (KEY_SIZE+1), buf, (short) (KEY_SIZE+1));
    }
    break;
case CREATE_NEW_KEY:
    of = Util.getShort(buf, ISO7816.OFFSET_P1);
    for (short i=0; i < of; i++)
    {
        // Generate new Kj
        hmac.init(IDj, Signature.MODE_SIGN);
        hmac.sign(buf, (short)0, KEY_SIZE, buf, KEY_SIZE);
    }
```

```
        break;
    case CREATE_NEW_HMAC:
        of = Util.getShort(buf, ISO7816.OFFSET_P1);
        len = (short)(buf[ISO7816.OFFSET_LC]&0x00ff);

        for (short i=0; i < of; i++)
        {
            // we use this key but this is similar
            hmac.init(IDj, Signature.MODE_SIGN);
            hmac.sign(buf, (short)0, len, buf, len);
        }
        break;



    // FOR DEBUG
    case RECEIVED_INDEX:
        len = apdu.setOutgoing();
        if (len!=(short)(1))
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        apdu.setOutgoingLength(len);
        buf[0] = index;
        apdu.sendBytes(of, len);
        break;
    case RECEIVED_POS_TO_INSERT:
        len = apdu.setOutgoing();
        if (len!=(short)(2))
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        apdu.setOutgoingLength(len);
        Util.setShort(buf, of, posToInsert);
        apdu.sendBytes(of, len);
        break;
    case RECEIVED_DATA_STORAGE:
        of = Util.getShort(buf, ISO7816.OFFSET_P1);
        len = apdu.setOutgoing();
        apdu.setOutgoingLength(len);
        apdu.sendBytesLong(datastorage, of, len);
        break;
    case RECEIVED_IDJ:
```

```
            len = IDj.getKey(buf, of);
            apdu.setOutgoingAndSend(of, len);
            break;
        case RECEIVED_KJ:
            len = Kj.getKey(buf, of);
            apdu.setOutgoingAndSend(of, len);
            break;
        case RECEIVED_ACKJ:
            len = ACKj.getKey(buf, of);
            apdu.setOutgoingAndSend(of, len);
            break;
        default:
            // good practice: If you don't know the INStruction, say so:
            ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
    }
}
```

C.2 Simulation des opérations de l'AR

Ci-après nous donnons les commandes APDU à envoyer à la carte contenant l'applet pour simuler les opérations que réalisent un AR.

— Sélection de l'application :

`/select 001122334401`

La carte répond avec un `0x9000` signifiant que tout se passe correctement.

— Initialisation de l'identifiant Id^j et la clé K_0^j à la valeur `00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF` :

`/send 000800001000112233445566778899AABBCCDDEEFF`

`/send 001000001000112233445566778899AABBCCDDEEFF`

La carte répond à chaque commande par un `0x9000` signifiant que tout se passe correctement.

— Simulation de la mesure de données d'un capteur (valeur `01 23 45 67 89 AB CD EF 01 23 45 67 89 AB CD EF 01 23 45 67 89 AB CD EF 01 23 45 67 89 AB CD EF`) et de son envoi vers la carte :

```
/send 00120000200123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
```

La carte répond par un `0x9000` signifiant que tout se passe correctement.

— Envoi d'une commande pour regarder le contenu du tableau `datastorage` à des fins de debug (ici les 72 premiers octets, soit `0x48`) :

```
/send 0036000048
```

On obtient la réponse ci-après suivi d'un `0x9000` signifiant que tout s'est passé correctement.

```
<= 44 36 3E FF 6C DE 1A DE A8 B2 44 AD 2E 3C 4E BD
   C8 B1 26 0F 9A 63 1E A7 0E C0 F1 E0 EE 33 AF 93
   6A 2B B2 8C 74 36 68 EE 67 B8 96 94 FE 37 12 F9
   04 B9 99 55 E8 0B 86 62 5F 7E 8C B7 CC 65 80 46
   DB 40 D9 7A F1 00 00 00
```

La valeur `44` represente la taille des données qui suivent (soit 68 octets). La valeur de $[SD_i^j]^{K_i^j}$ est `36 3E FF 6C DE 1A DE A8 B2 44 AD 2E 3C 4E BD C8 B1 26 0F 9A 63 1E A7 0E C0 F1 E0 EE 33 AF 93 6A` alors que la valeur de T_i^j est `2B B2 8C 74 36 68 EE 67 B8 96 94 FE 37 12 F9 04 B9 99 55 E8` et celle de ACK_i^j est `0B 86 62 5F 7E 8C B7 CC 65 80 46 DB 40 D9 7A F1`. Les trois zéros qui suivent sont là car il n'y a eu pour l'instant qu'une seule donnée chiffrée.

— Simulation de l'envoi de la carte en réponse à la commande *Request* par le MDC :

```
/send 0014010000
```

On obtient la réponse ci-après suivi d'un `0x9000` signifiant que tout s'est passé correctement.

```
<= 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF
   36 3E FF 6C DE 1A DE A8 B2 44 AD 2E 3C 4E BD C8
   B1 26 0F 9A 63 1E A7 0E C0 F1 E0 EE 33 AF 93 6A
   AF 38 FA 96 C2 AF 54 59 04 90 95 9E 93 16 A9 14
```

On retrouve Id^j dont la valeur est `00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF`, les données chiffrées $[SD_i^j]^{K_i^j}$ et HM_i^j dont la valeur est `AF 38 FA 96 C2 AF 54 59 04 90 95 9E 93 16 A9 14`.

— Si le MDC envoie la mauvaise accréditation, comme par exemple la valeur `11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11`, alors la carte renvoie une erreur avec le `0x6982`.

— Si le MDC envoie la bonne accréditation, c'est-à-dire `2B B2 8C 74 36 68 EE 67 B8 96 94 FE 37 12 F9 04 B9 99 55 E8` alors la carte fournit d'autres données si elle en a de disponible et sinon elle retourne `0x6666` signifiant qu'elle n'a plus de données disponibles.