Securing wireless sensor and vehicular networks

Wafa Ben Jaballah

THÈSE

PRÉSENTÉE À

L´ UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE EN MATHÉMATIQUES ET INFORMATIQUE

par

Wafa Ben Jaballah

Pour obtenir le grade de **DOCTEUR**

Université de Bordeaux, France

SPÉCIALITÉ : INFORMATIQUE

SECURING WIRELESS SENSOR AND VEHICULAR NETWORKS

SOUTENUE LE : 08 Janvier 2014

Devant le Jury de Thèse :

Mauro Conti	Président du jury, Université de Padoue
Frederic Cuppens	Rapporteur, Telecom Bretagne
Francine Krief	Examinateur, Institut Polytechnique de Bordeaux
Maryline Laurent	Rapporteur, Telecom SudParis
Mohamed Mosbah	Directeur de thèse, Institut Polytechnique de Bordeaux
Habib Youssef	Directeur de thèse, Université de Sousse

To the memory of my beloved mother,
to the love of my family,
and to the love of the other people to whom I did not dedicate enough time.

"We must have perseverence and above all confidence in ourselves. We must believe that we are gifted for something and that this thing must be attained."

–Marie Curie–

TABLE OF CONTENTS

Page

Page

Page

List of Tables

List of Figures

ABSTRACT

Wireless sensor and vehicular networks play an important role in critical military and civil applications, and pervade our daily life. However, security concerns constitute a potential stumbling block to the impeding wide deployment of sensor networks and vehicular communications. This dissertation studies communication security for Wireless Sensor Networks (WSNs), and vehicular communication. To this aim, we address four important aspects. The first study addresses broadcast authentication in WSNs. We focus on key disclosure based schemes. We demonstrate that key disclosure delay induces an authentication delay, which could lead to a memory DoS attack. We then propose two broadcast authentication protocols for WSNs, which overcome the security vulnerability of existing solutions. The proposed schemes guarantee the efficient management of receiver's buffer, by employing a staggered authentication mechanism, and a Bloom filter data structure to reduce the communication overhead. We also validate our protocols under the AVISPA model checking tool, and we evaluate them with experiments under TinyOS. Our findings are that these protocols provide source authentication service while respecting the WSN constraints.

The second study addresses the storage issue in WSNs, in particular the Delayed Authentication Compromise attack (DAC). We first demonstrate that recently proposed schemes, which also address the DAC issue are vulnerable to two kinds of attacks: switch command attack (where an adversary pretends to "switch" two messages over time), and drop command attack (where an adversary just pretends to "hide" a message sent from the broadcaster). As a countermeasure against these attacks, we propose a new solution for broadcast authentication. Our analysis shows that our solution is effective in detecting both switch command and drop command attack, and—at the same time—is more efficient (in terms of both communication and computation) than the state of the art solutions.

In the third study, we address key management security in WSNs. We present novel symmetric-key-based authentication schemes which exhibit low computation and communication authentication overhead. Our schemes are built upon the integration of a reputation mechanism, a Bloom filter, and a key binary tree for the distribution and updating of the au-

thentication keys. Our schemes are lightweight and efficient with respect to communication and energy overhead.

The fourth study addresses security in vehicular communications. We focus on fast multi hop broadcast applications. We analyze the security threats of state of the art vehicular based safety applications. We demonstrate that these schemes are vulnerable to the position cheating attack, the replay broadcast message attack, and the interrupting forwarding attack. Then, we propose countermeasures for these threats. We hence propose a complete solution which is both fast and secure in broadcasting safety related messages: Fast and Secure Multi-hop Broadcast Algorithm (FS-MBA). Finally, we confirm the efficiency and feasibility of our proposals using an extensive set of simulations under NS-2 Simulator.

Keywords: Source Authentication in WSNs, Security, Secure Vehicular Communications, Key management, Attacks.

RESUME

Les Réseaux de Capteurs Sans Fils (RCSFs) et les réseaux véhiculaires sont de plus en plus répandus, et déployés dans des domaines d'applications variés tels que la santé, la surveillance environmentale, les applications d'alerte d'accident, et les applications militaires. Cependant, ces réseaux peuvent être sujets à des attaques, ce qui empêche leur utilisation à grande échelle. Cette thèse étudie la sécurité des communications pour les réseaux de capteurs sans fils, et les communications inter-véhiculaires. Dans ce but, nous abordons quatre aspects importants. La première étude porte sur l'authentification des messages diffusés dans les réseaux de capteurs. Nous nous concentrons sur les principaux schémas à base de divulgation de clés d'authentification. Nous démontrons que le délai de divulgation de clé induit un délai d'authentification, ce qui pourrait conduire à une attaque de mémoire de déni de service. Nous proposons ensuite deux protocoles d'authentification de la source dans les RCSFs, pour surmonter la vulnérabilité des solutions existantes. Les schémas proposés garantissent la gestion efficace de la mémoire tampon du récepteur, en utilisant un mécanisme d'authentification par niveau, et une structure de Filtre de Bloom afin de réduire le coût de communication. Ensuite, nous validons nos protocoles en utilisant l'outil de vérification AVISPA, et nous les évaluons avec des expérimentations dans l'environment TinyOS. Nous confirmons que ces protocoles fournissent un service d'authentification de la source tout en respectant les contraintes de RCSFs.

La seconde étude porte sur le problème de stockage au niveau des capteurs. Nous considérons en particulier l'attaque d'authentification différée "Delayed Authentication Compromise" (DAC) dans les RCSFs, qui permet à un attaquant d'utiliser une clé déjà divulguée pour signer d'autres messages. Nous montrons d'abord que les systèmes récemment proposés qui sont résistants également à l'attaque DAC sont vulnérables aussi à deux types d'attaques: attaque de permutation de commandes (où un adversaire prétend "permuter" deux messages au fil du temps), et l'attaque de rejet de commandes (où un adversaire semble "cacher" un message envoyé par la station de base). Nous proposons ensuite une nouvelle solution d'authentification. Notre analyse montre que notre solution est efficace pour détecter à la fois l'attaque de permutation de commandes et l'attaque de rejet de commandes, — et en même temps — est plus efficace (en termes de communication et de calcul) que les solutions existantes.

Dans la troisième étude, nous considérons le problème de la sécurité de la gestion des clés dans les réseaux de capteurs. Nous présentons de nouveaux schémas d'authentification à base de clés symétriques qui présentent un faible coût d'authentification et de communication. Nos systèmes sont construits en intégrant un mécanisme de réputation, un filtre de Bloom, et un arbre binaire de clés pour la distribution et la mise à jour des clés d'authentification. Nos schémas d'authentification sont efficaces en matière de communication et de consommation de l'énergie.

La quatrième étude porte sur la sécurité des communications véhiculaires. Nous nous concentrons sur les applications d'alerte d'accident. Nous analysons les menaces pour un ensemble d'algorithmes. Nous démontrons que ces systèmes sont vulnérables à l'attaque d'injection d'une fausse position, à l'attaque de rejeu de message d'alerte, et à l'attaque d'interruption de message d'alerte. Ensuite, nous proposons des contre-mesures à ces menaces. Nous avons donc proposé une solution qui est à la fois rapide et sécurisée pour les applications d'alerte d'accident : Un algorithme rapide et sécurisé pour la diffusion des messages en multi-saut (FS-MBA). Enfin, nous confirmons l'efficacité et la faisabilité des différents protocoles en effectuant un ensemble de simulations sous le simulateur NS-2.

Mots-Clés : Authentification de la Source dans les RCSFs, Sécurité, Sécurisation des Communications Véhiculaires, Gestion des Clés, Attaques.

RESUME DETAILLE

Au cours des dernières décennies, il y a eu une croissance importante dans le domaine de l'informatique ubiquitaire et omniprésente. En particulier, l'informatique ubiquitaire vise à rendre "plusieurs ordinateurs disponibles dans l'environnement physique, mais aussi transparents pour l'utilisateur" [1]. Pendant cette période, les récents progrès des systèmes électro-mécaniques, de la communication sans fil et de l'électronique numérique ont facilité la production de petits appareils performants et pas chers, tels que les téléphones intelligents, les PDA, les systèmes d'identification par radiofréquence (RFID), les Réseaux de Capteurs Sans Fils (RCSFs), et de nombreuses autres technologies. Ces appareils mobiles sont capables d'effectuer diverses opérations complexes sans perdre de vue les besoins des utilisateurs. Les appareils sont utilisés dans de grandes variétés d'applications telles que la surveillance de la santé, les systèmes de diagnositc, les alarmes de sécurité, la communication véhiculaire et la surveillance de l'environnement.

Dans de nombreux scénarios d'application, la sécurité est un besoin essentiel. Dans cette thèse, on se focalise sur la sécurité des communications dans les réseaux de capteurs sans fils (RCSFs), et les communications inter-véhiculaires. Nous allons mettre l'accent sur les exigences en sécurité pour ces deux types de technologies, et par la suite nous allons aborder les différentes motivations qui permettent de sécuriser les RCSFs et les communications inter-véhiculaires.

Authentification de la source L'authentification de la source des messages de diffusion d'une station de base à des capteurs est le paradigme le plus commun dans les réseaux de capteurs sans fil. Cette authentification est trés importante, car elle permet à la station de base de diffuser des requêtes et des messages aux capteurs et d'exploiter ainsi efficacement le réseau de capteurs sans fil. L'authentification permet aux récepteurs de vérifier que les messages reçus proviennent de la source prétendue, et n'ont pas été modifiés en cours de route. Le problème devient plus complexe dans le cas où d'autres récepteurs de données sont non fiables, et où les paquets perdus ne sont pas retransmis. L'authentification de la source des messages diffusés est donc l'un des services de sécurité les plus importants dans ces réseaux [2–8].

Un réseau de capteurs peut être déployé dans des environnements hostiles où il y a des attaques malveillantes. Dans une telle situation, la sécurité devient une des préoc-

cupations majeures. En tant que service de sécurité fondamental, l'authentification de la source des messages diffusés permet à un expéditeur de diffuser des données critiques et/ou des commandes de noeuds de capteurs d'une manière authentifiée. Par exemple un attaquant est incapable d'intercepter un message de l'expéditeur [9–15]. Toutefois, en raison des contraintes de ressources sur les noeuds de capteurs, les techniques d'authentification traditionnelles telles que les signatures numériques basées sur des clés publiques ne sont pas souhaitables. Dans [16], les auteurs ont développé μTESLA pour l'authentification de la source des messages diffusés dans les réseaux de capteurs basée sur la cryptographie symétrique, ce qui élimine la dépendance à la cryptographie à clé publique. Plusieurs schémas à base du protocole μTESLA ont été proposés pour étendre la capacité de protocole μTESLA [4–7, 17]. Malgré ces progrès récents, plusieurs questions ne sont toujours pas correctement traitées.

- **Authentification Immédiate des Messages**. Les schémas à base de μTESLA souffrent d'un retard d'authentification, en raison du retard de divulgation de la clé d'authentification. Ce processus se réfère à une asymétrie temporelle, où la clé est secrète au niveau de la station de base, et elle sera divulguée après certains intervalles de temps (délai de divulgation), rendant ainsi la clé publique. Dans ce scénario, les travaux dans [7, 18] prouvent que ces mécanismes sont vulnérables contre certaines attaques comme celles de déni de service.

- **Résistance aux Attaques de Déni de Service (DoS)**. Un attaquant peut lancer les attaques DoS sur les messages pour surcharger la mémoire tampon, et il exploite le délai d'authentification des messages pour intercepter des paquets de données. Bien que plusieurs solutions aient été proposées dans [7], celles-ci utilisent une bande passante importante ou bien nécessitent d'importantes ressources au niveau de l'expéditeur (en utilisant par exemple les énigmes cryptographiques) [19].

- **Coût de Communication**. Comme les capteurs ont des ressources limitées, le coût de communication est crucial. Ainsi, pour concevoir un système d'authentification efficace dans ces réseaux, le coût de communication doit être réduit.

L'Attaque de Délai d'Authentification Les schémas d'authentification qui se basent sur les mécanismes de μTESLA entraînent une vulnérabilité qui peut conduire à une attaque appelée "Delayed Authentication Compromise"(DAC). Considérons le scénario suivant: le récepteur doit identifier les messages reçus, de telle sorte que l'attaquant ne puisse pas modifier successivement les messages. Les mécanismes à base de divulgation de clé, ou bien à base de μTESLA ne sont pas candidats à être utilisés dans un tel scénario. En effet, dès que la clé est divulguée, l'attaquant peut utiliser cette clé pour modifier les messages enregistrés. Dans [18], ce processus d'attaque désigne l'attaque DAC. L'adversaire peut utiliser une clé déjà divulguée pour signer par exemple la commande “ouvrir une porte”, et l'enregistrer dans la mémoire d'un actionneur compromis. Une fois que la porte est ouverte, il sera difficile de prouver avec certitude ce qui a été compromis. Si la station de base est déclarée honnête, alors nous pouvons affirmer que le noeud est malveillant, mais si la station

de base elle-même peut être compromise, alors il n'est pas possible de déterminer qui est le malveillant, le noeud ou la station de base. Afin de contrer ces attaques du DAC, les auteurs dans [18] proposent trois algorithmes appelés PASS, TASS et PTASS. Cependant, plusieurs questions ne sont toujours pas correctement traitées.

- **Attaque de Suppression de Commande**. Un noeud malveillant peut facilement supprimer/rejeter une commande transmise par la station de base. Un noeud récepteur ne pourra pas détecter si la station de base n'a pas envoyé une commande, ou bien si la commande a été supprimée par un noeud malveillant.
- **Attaque de Permutation de Commande**. Les protocoles (TASS et PTASS) sont également vulnérables à l'attaque de permutation de commande qui vise à changer l'ordre des messages envoyés par la station de base. Prenons le scénario où la station de base doit envoyer la commande "initialiser" suivie par la commande "fermer la porte", tandis que l'attaquant souhaite ouvrir la porte. En utilisant TASS et PTASS, le noeud malveillant est capable de changer l'ordre des commandes, poussant le récepteur à exécuter la commande "fermer la porte", puis d'exécuter la commande "initialiser", ce qui va rouvrir la porte.
- **Authentification Immédiate des Messages**. En utilisant le protocole TASS, le récepteur doit attendre la fin d'un intervalle de temps afin d'authentifier les messages reçus. Cela peut conduire également à des attaques DoS. PTASS propose un moyen d'envoyer des messages de haute priorité qui sont immédiatement authentifiés. Toutefois, cette fonctionnalité engendre une augmentation de la communication et de la transmission.

Dans de nombreuses applications, il est important de protéger les communications entre les noeuds de capteurs pour maintenir l'authentification, la confidentialité et l'intégrité.

En particulier, les efforts de recherche pertinents ont abordé la technique de clé de groupe afin de localiser le calcul, et aussi pour diminuer les coûts de communication dans les réseaux de capteurs sans fil [20]. La technique à base d'un arbre de clés est utilisée pour sécuriser les communications de groupe. Plusieurs techniques de gestion sécurisée de clé de groupe ont été proposées pour sécuriser les communications multicast [21, 22]. Dans un système de gestion de clé de groupe typique [23], il y a un tiers de confiance, dit centre de distribution de clés (KDC). Cette entité unique et fiable est responsable de la génération et de la distribution de clés aux membres de groupe. Dans [21], les auteurs proposent un système de gestion de clés de groupe se basant sur une hiérarchie des clés topologiques. Il génère un arbre de clés en se référant à la topologie de réseau de capteurs. Dans [24, 25], les auteurs proposent un polynôme de degré t avec le terme constant du polynôme étant la clé secrète, en calculant t termes distincts du polynôme et les stockant. Dans [26], les auteurs ont conçu un système utilisant un arbre de clé pour gérer les membres du groupe. Par ailleurs, dans [26], la technique de clé de groupe repose sur une refonte de système de groupe centralisé basé sur l'arbre logique de clés dans les réseaux de capteurs. Cette technique n'est pas pratique pour les réseaux de capteurs, puisqu'elle utilise des opérations complexes.

Bien que de nombreux travaux dans [9,11] aient proposé des mécanismes d'authentification de la source dans les réseaux de capteurs sans fil, le problème est toujours difficile puisqu'il faut réaliser un compromis entre la sécurité et les ressources de calcul limitées.

Sécurité des communications inter-véhiculaires On s'intéresse dans cette thèse à la sécurité des communications entre véhicules. En effet, la communication entre véhicules est parmi les applications les plus prometteuses [27, 28]. De nombreuses applications sont possibles, en particulier, les systèmes d'alerte de dangers.

En résumé, plusieurs algorithmes de diffusion multi-sauts ont été proposés. Ces algorithmes partagent généralement un ensemble de propriétés, se situant dans une même classe de solutions. Malheureusement, ils ont tous été développés sans tenir compte des aspects de sécurité. La sécurité est un problème fondamental, qui ne devrait pas être négligé [29]. En effet, les attaquants pourraient exécuter des actions malveillantes pour injecter des fausses informations ou de fausses alarmes.

Contributions Les principales contributions de cette thèse se résument comme suit.

- **Contribution 1 : Authentification de la source des messages diffusés dans les RCSFs**. Nous identifions le problème de l'authentification de la source des messages diffusés dans les RCSFs. Nous détectons une vulnérabilité de sécurité pour les protocoles à base de clé symétrique qui se basent sur μTESLA. Cette vulnérabilité pourrait conduire à une attaque de déni de service affectant la mémoire des capteurs. Nous proposons ensuite plusieurs schémas à clé symétrique permettant de limiter les impacts de cette attaque. Nous atteignons notre objectif en intégrant plusieurs techniques telles que le mécanisme d'authentification par niveau afin de réduire le nombre de paquets forgés dans le tampon de récepteur. Nous concevons un système basé sur un filtre de Bloom qui permet de concevoir un protocole avec un coût de communication faible. Nous validons formellement nos solutions de sécurité, et nous évaluons leurs performances sous le simulateur TOSSIM. Cette contribution est présentée dans le Chapitre 2.

- **Contribution 2 : Authentification différée dans les RCSFs**. Tout d'abord, nous montrons que les protocoles existants qui sont résistants à l'attaque DAC sont soumis à deux attaques possibles : l'attaque de rejet/suppression de commandes, et l'attaque de permutation de commandes. Deuxièmement, nous proposons une nouvelle solution qui est résistante à l'attaque DAC, à l'attaque de rejet/suppression et à l'attaque de permutation de commandes. En plus, notre solution est plus efficace par rapport à l'état de l'art, ce qui permet également l'authentification immédiate des messages. Cette contribution est bien détaillée dans le Chapitre 3.

- **Contribution 3 : Communications du Groupe dans les RCSFs**. Le problème de la fourniture de l'authentification de la source dans les RCSFs a été un obstacle pour leur déploiement à grande échelle. Nous présentons un schéma basé sur une clé

symétrique ayant un faible coût de calcul et de communication. Nos systèmes sont construits en intégrant un mécanisme de réputation, un filtre de Bloom, et un arbre binaire de clés pour la distribution et la mise à jour des clés d'authentification. Nos schémas sont efficaces en matière de communication et de consommation d'énergie. Cette contribution est décrite dans le Chapitre 4.

- **Contribution 4 : Sécurité des communications entre véhicules**. Nous analysons la sécurité d'un algorithme représentant la diffusion rapide multi-saut (FMBA) [30,31] comme un état de l'art des applications de sécurité basées sur les communications entre véhicules. Nous proposons des contre-mesures pour gérer les menaces de sécurité. En particulier, nous nous concentrons sur l'une des principales menaces: la possibilité d'attaquer le protocole pour entraver son fonctionnement. Nous identifions les problèmes et proposons des contre-mesures possibles. Les solutions proposées et identifiées pour FMBA peuvent être également adaptées à d'autres protocoles/algorithmes, appartenant à la même catégorie générale d'applications. Cette contribution est bien détaillée dans le Chapitre 5.

CHAPTER 1

Introduction

Over the last few decades there have been tremendous developments in the area of pervasive and ubiquitous computing. In particular, the ubiquitous computing aims to make "many computers available throughout the physical environment, but making them effectively invisible to the user" [1]. During this era, recent advances in Micro Electro Mechanical Systems (MEMS), in wireless communication and in digital electronic made it possible to produce small, cheap and "smart" devices such as smart-phones, PDAs, Radio Frequency IDentification systems (RFID), Wireless Sensor Networks (WSNs), and many other technologies. These portable devices are capable of performing various complex operations while keeping in view the needs of users. These devices are being used in wide varieties of applications such as health monitoring, diagnostic systems, safety alarms, vehicular communication, and environment monitoring.

Moreover, annually, road crashes result in almost 120000 fatalities and 2.4 million injuries in the European Region [32]. Road traffic injuries represent the leading cause of death among adolescents and young adults [33], and the economic burden of road crashes is as much as 3% of gross domestic product. Nevertheless, many effective preventive strategies exist [32]. Therefore there is potential to take up the challenge and reduce the burden of road traffic injuries. The basic approach consists of using advanced technologies that can prevent vehicles from being involved in accidents. In this direction, one of the most promising techniques is based on the use of inter vehicular communication (IVC) [27,28,34]. Many applications are possible in this context, yet local danger warning systems remain the most prominent ones. Secure communication is a crucial aspect that must not be overlooked. Indeed, attackers might run malicious actions to inject false information or alarm, thus rendering ineffective the safety application [35–41].

In this thesis, we focus on the security issues of Wireless Sensor Networks and vehicular communications [42–44], introduced in the following section.

1.1 Wireless Sensor Networks and Vehicular Communications

In the following, we present Wireless Sensor Networks as well as their specific constraints.

1.1.1 Wireless Sensor Networks

In this thesis, a sensor device is a small device that is able to sense environmental data (sound, light, temperature, etc.). The device is also able to communicate with other sensor nodes in its communication range and compute the sensed/received data. A set of these sensor devices deployed in a given area constitutes a network with no pre-established architecture, also called Wireless Sensor Network (WSN). The usefulness of this type of network comes from the collaboration of a big amount of nodes. In a WSN hundreds or thousands of nodes are usually deployed in a large area where they can sense data, compute and communicate the collected data in a very efficient and distributed way. Sensor nodes are different compared to other traditional wireless devices, they do not communicate directly with a Base Station (BS)- a device that does not have the limitations of a sensor node- but mainly with other sensor nodes. The sensed data are locally computed and forwarded to the BS. WSN applications are in different fields: building surveillance, battlefield monitoring, fire prevention, and so on. Most WSN applications require security (confidentiality, integrity, authenticity, and availability) as a fundamental building block. If we consider a WSN deployed in order to detect an area, e.g, for the detection of poisonous gas that could potentially be released during a concert or a big sport event. In this scenario, if the network is not secure we could have worse damages. In order to develop useful security mechanisms, it is necessary to know and understand the constraints of WSNs first. In the following, we present the challenges specific to WSNs.

Challenges The following features (resource constraints, unreliable communication, network constraints, and vulnerability to novel attacks) make it particularly challenging to protect communication security in WSNs.

- **Resource Constraints**. In WSNs, the nodes are battery-powered, thus they only have a limited energy supply. This again requires the security design to be efficient regarding both communication and computation overheads. In most cases, sensors only have very limited memory, which further narrows down the security design choices. All these resource constraints require that the security design can only be efficient and lightweight; otherwise it will not be practical for WSNs.
- **Unreliable Communication**. Unreliable communication is another threat to sensor security. The security of the network relies heavily on a defined protocol, which in turn depends on communication.
  - **Unreliable Transfer**. Normally the packet-based routing of the sensor network is connection less and thus inherently unreliable. Packets may get damaged due to channel errors or dropped at highly congested nodes. The result is lost or missing packets. Furthermore, the unreliable wireless communication channel also results in damaged packets. A higher channel error rate also forces the software developer to devote resources to error handling. More importantly, if

the protocol lacks the appropriate error handling it is possible to lose critical security packets. This may include, for example, a cryptographic key.

  - **Conflicts**. Even if the channel is reliable, the communication may still be unreliable. This is due to the broadcast nature of the wireless sensor network. If packets meet in the middle of transfer, conflicts will occur and the transfer itself will fail. In a high density sensor network, this can be a major problem. More details about the effect of wireless communication can be found at [45].
  - **Latency**. Multi-hop routing, network congestion, and node processing can lead to greater latency in the network, thus making it difficult to achieve synchronization among sensor nodes. The synchronization issues can be critical to sensor security where the security mechanism relies on critical event reports and cryptographic key distribution. Real-time communication issues in wireless sensor networks are discussed in [46].

- **Network Constraints**. Wireless networks use wireless open channels, therefore an adversary can easily eavesdrop on all the network communications, as well as arbitrarily injecting messages and launching jamming attacks at different network layers. This means that the security design has to take into account both passive and active attacks. Wireless Sensor Networks are distributed in nature, therefore centralized security solutions cannot be an option. This also means that these environments are vulnerable to various DoS attacks. WSNs in particular are often very large in scale, which in turn imposes scalability requirements on the security design of WSN applications.

- **Malicious attacks**. WSNs are prone to several kinds of novel attacks as stated in Section 1.3. For instance, an adversary could inject some bogus information in the network and let the honest nodes believe that it is an authentic participant in the network, thereby acquiring all the information traversed in the network.

In the following, we present vehicular communications systems and some major challenges specific to this technology.

1.1.2 Vehicular Communications

Vehicles will be equipped with novel computing, communications, and sensing capabilities. These will increase and support a number of applications that enhance the transportation safety and efficiency. A key aspect of vehicular communications systems is to expand the time horizon of information relevant to driving safety. Based on their own sensing and on information received from nearby peers and Road-Side-Units (RSUs), vehicles can anticipate, detect, and avoid dangerous situations. For example, messages about accidents, timely notifications about upcoming lane changes, and alerts about unsafely approaching vehicles, can be highly beneficial. Interconnected vehicles not only collect information about themselves and their environment, but they also exchange this information in real time with

other vehicles. Security is a fundamental problem in this context which should not be overlooked [29]. Indeed, attackers might run malicious actions to inject false information or alarm, thus rendering ineffective the safety application [35–41]. In the following, we discuss some major constraints of vehicular communications that could affect security.

Challenges The user requirements, the operational conditions, and constraints on vehicular communication systems make security a hard problem [47]. The most significant challenges specific to the vehicular communications range from network volatility, to liability and privacy, delay-sensitive applications, and heterogeneity.

- **Network Volatility**. Let us consider two vehicles traveling on the highway that may remain for a limited period of time, within a few wireless hops, or within their transceiver range. This means that vehicular networks lack the relatively long-lived context. As a result, password-based establishment of secure channels, gradual development of trust by using a circle of trust, or secure communication only with a handful of endpoints, may be not practical for securing vehicular communication.
- **Heterogeneity**. Security solutions should retain flexibility, efficiency, and operability. Vehicular communication technologies as well as their supported applications are various and heterogeneous. Nodes are possibly equipped with external devices as Geographical Positioning Service (GPS), digital audio, or cellular transceivers. A secure solution should be inter operable.
- **Delay-Sensitive Applications**. Safety applications impose strict deadlines for message delivery and are time-sensitive. Security design must take these constraints into consideration. Protocols should be lightweight and robust to denial of service (DoS) attacks. An adversary can generate a high volume of bogus information and consumes resources, which could result in message delay.
- **Liability vs. Privacy**. Vehicular communication is envisioned as an excellent opportunity to obtain hard-to-refute data that can assist legal investigation, (e.g., in car accidents). The identification of the vehicles as sources of messages should be possible and without ambiguity. Moreover, informations related to coordinates, time intervals, and associated vehicles, should be possible to extract.
- **Scalability**. An important challenge is the scale of the network, which is roughly a billion of vehicles around the world. This makes the design to provide cryptographic keys a challenge, taking into account that there are also various authorities governing transportation systems. Moreover, the security scheme should take into account the existence of collisions or retransmissions when the vehicle density increases.

All the malicious attacks, and the specific features interleaved together, impose many challenging requirements for the security design in WSNs and vehicular communications. Sophisticated techniques and careful design are demanded to balance among all these competing and sometimes even conflicting requirements as desired by the WSN applications or vehicular applications.

In the following, we report an overview of security issues in Wireless Sensor Networks and vehicular communications [42–44]. In particular, Section 1.2 presents the typical security requirements under the context of WSNs and vehicular communications. Section 1.3 describes the main attacks in WSNs and vehicular communications. Section 1.4 presents some of the main defensive measures. Section 1.5 presents the contributions of our work. The following chapters of this thesis provide a deeper discussion of some aspects presented in these sections.

1.2 Security Requirements and Related Issues

In the following, we report the major security requirements and related issues for WSNs and vehicular communications [42–44, 48].

Confidentiality Data confidentiality is an important issue in network security. Confidentiality ensures that the content of the message being transferred is never disclosed to unauthorized entities. Network transmission of sensitive information, such as military information, requires confidentiality.

When we focus on sensor networks, the confidentiality relates to the following:

- In many applications nodes communicate highly sensitive data, e.g., key distribution, therefore it is important to build a secure channel in a wireless sensor network.
- A sensor network should not leak sensor readings to its neighbors.
- To protect against traffic analysis, public sensor information such as sensor identities and public keys, should be encrypted.

To achieve confidentiality, a standard approach is to encrypt the data with a secret key that is only intended for a receiver to possess.

Authenticity Authenticity enables a node to make sure the identities of its communicating entities. Thus, an adversary could not masquerade another entity, and disseminate forged messages [44]. An adversary can inject additional packets. So the receiver needs to ensure that the data originates from the correct source. Moreover, the authentication service is necessary for many administrative tasks (e.g., network reprogramming or controlling sensor node duty cycle). Message authentication is crucial for many applications in sensor networks and vehicular communications. Data authentication can be achieved through a purely symmetric mechanism in case of two-party communication. In this case, the sender and the receiver share a common secret key to compute the Message Authentication Code (MAC) of all communicated data. In a multicast scenario, this approach is not feasible. Thus some approaches have been proposed in WSNs. Adrian Perrig et al. propose a key chain distribution system called μTESLA [13,49]. The basic idea of μTESLA is to use a key chain to generate and disclose authentication keys. Moreover, in μTESLA, initially before the authentication of broadcast messages, some initial information must be unicast to each

sensor node. Liu and Ning [5] propose Multi-level-μTESLA that broadcasts the key chain commitments rather than using μTESLA's unicasting technique.

Integrity Integrity ensures that a message being transferred is never corrupted or modified by an adversary without being detected. An adversary may be unable to steal information with the implementation of confidentiality. However, this does not mean the data is safe. In fact, the adversary can change the data. For instance, a malicious node may add some fragments or manipulate the data within a packet, and this new packet can then be transmitted to the original receiver. Data loss or damage can even occur also due to harsh communication environment, and without the presence of a malicious node. Thus, integrity ensures that any received data has not been altered in transit.

Data Freshness Data freshness means that the data is recent, and that it ensures that no old messages have been replayed. Even when confidentiality and integrity services are provided, we need to ensure the freshness of each message. For instance, shared-key approaches need data freshness, and the keys should be changed over time. As it takes time for the new shared keys to be propagated to the entire network, an adversary can use a replay attack. Thus, it is easy to disrupt the normal behavior of the node, when the node is not aware of the new key change time. A standard approach to solve this problem is to add a nonce, or another time-related counter, to ensure data freshness.

Availability Availability ensures the survivability of network services despite denial of service (DoS) attacks [7]. A DoS attack could be launched at any layer of the network and could be of various forms. At the network layer, an adversary could disrupt the routing protocol and disconnect the network. At the application layer, an adversary may bring down high-level services such as network broadcast and multicast. Despite the importance and the spread of WSNs and vehicular communications, providing satisfactory security protection has never been an easy task. This is because WSNs not only suffer from various malicious attacks; but also are subject to many resource and network constraints as compared to traditional wireless networks.

In a WSN environment, adjusting traditional encryption algorithms to fit within the network will introduce some extra costs [7]. To this aim, some approaches modify the code to reuse as much code as possible. Other approaches make use of additional communication to achieve the same goal. Some approaches force strict limitations on the data access, or propose a central point scheme to simplify the algorithm. However, these approaches are not feasible in WSNs and weaken the availability of a sensor network. The first reason is that additional computation consumes additional energy, and the data will be no longer available when the battery of the sensor is depleted. The second reason is that additional communication also consumes more energy. A third reason is that a single point of failure will be introduced if using the centralized scheme. This threatens the availability of the network and motivates the research on security.

Time Synchronization Most wireless network applications, often require a scalable time synchronization service enabling data consistency and coordination [50]. In sensor network applications, in order to conserve power, an individual sensor's radio may be turned off for periods of time. In [51], the authors propose a flooding time synchronization protocol (FTSP), especially tailored for applications, requiring stringent precision on resource limited wireless platforms. The FTSP protocol uses periodic flooding of synchronization messages. It is based on using MAC-layer time-stamping and comprehensive error compensation including clock skew estimation. The first version of the protocol FTSP was first implemented on the Berkley Mica2 platform, and then on TelosB motes [52].

Self Organization Self Organizing a network with specific assignment of roles and tasks to the devices based on their wireless connectivity or sensing characteristics is an important problem [53]. Self organization involves abstracting the communicating entities into an easily controllable network infrastructure. Cluster or connected dominating set, tree, grid, or mesh based organizations are typical. For example, in a sensor network, the dynamics of the whole network inhibits the idea of pre-installation of a shared key between the base station and all sensor nodes [54]. Thus, the nodes must self-organize to conduct key management and build a trust relation among them. In the context of symmetric encryption techniques [54, 55], several random key pre-distribution schemes have been devised. Moreover, in the context of public-key distribution, several public-key cryptography techniques have been proposed. Self organization is very necessary since when this service is lacking in a network, the damage resulting from an attack may be devastating.

1.3 Attacks

In this section, we give an overview of some attacks on wireless sensor networks, and vehicular communications.

Attacks on Wireless Sensor Networks Sensor networks are particularly vulnerable to several types of attacks. Attacks can be performed in a variety of ways ranging from denial of service to delayed authentication compromise, physical attacks, sybil attacks, wormhole attacks, and so on. In the following, we present some major attacks on WSNs, that could also be applied to vehicular communications.

- **Denial of Service Attacks**.

  A standard attack is simply to jam a node or a set of nodes. Jamming consists in this case to the transmission of a radio signal that interferes with the radio frequencies being used by the network [56]. There are two forms of jamming a network: Constant jamming and Intermittent jamming. The former (Constant jamming) involves the complete jamming of the entire network. No messages are able to be sent or received. In Intermittent jamming, nodes are able to exchange periodically but not consistently.

This can have an impact on the sensor network when exchanged messages between nodes are time sensitive [56].

Denial of service attacks assume a particular importance in wireless sensor networks, where it is not possible to afford the computational overhead necessary in implementing many of the typical defensive strategies of traditional computing. Attacks can be made on the link layer itself. For instance, an attacker may simply intentionally violate the communication protocol, e.g., ZigBee [57], or IEEE 801.11b (Wi-Fi) protocol, and then transmit messages continually in an attempt to generate collisions. Thus, the collisions would require the retransmission of any packet affected by the collision. Using this technique, an attacker can easily deplete a node's power supply by doing many retransmissions.

The routing layer is also susceptible to DoS attacks. A node can take advantage of a multi hop network by simply refusing to route messages constantly or intermittently.

In the transport layer, the malicious node can opt to flooding. Flooding can be as simple as sending many connection requests to a susceptible node. In this case, resources must be allocated to handle the connection request. As a result, the node's resources will be exhausted. Finally, a denial of service attack can be performed against the specific application level protocol.

We address the DoS attack in Chapter 2, and we demonstrate that authentication delay in μTESLA based schemes could lead to a memory DoS attack. In Chapter 5, we address also this kind of attack when analyzing the security threats to a state of the art multi-hop broadcast algorithm for vehicular communications.

- **Delayed Authentication Compromise Attack**. In order to present the delayed authentication compromise attack, let us consider the scenario when the receiver needs to log the received messages, in such a way that an attacker cannot successively modify the messages. Key disclosure based schemes (μTESLA based schemes) are not candidates to be used in such scenario. Indeed, as soon as the key is disclosed, the attacker may use that key to modify the logged messages. In [18], this attack process refers to Delayed Authentication Compromise (DAC) attack. The adversary may use an already disclosed key to sign for instance the command "open the valve", and save it in the memory of a compromised actuator. Once noticed that the valve has been opened, it will be difficult to prove with certainty who has been compromised. If the base station is trusted, we can state that the node is cheating; however if the base station itself can be compromised, then it is not possible to determine who is cheating, the node or the base station. We present the DAC attack in Chapter 3, and we demonstrate that several issues are still not properly addressed.

Attacks on Vehicular Communications The major attacks on vehicular communications could be classified into Bogus Information, Cheating Positioning information, ID disclosure of other vehicles, Denial of Service, and Masquerade [39]. In the following we present a brief overview of these attacks.

- **Bogus information.** One or several legitimate members of the network send out false information to misguide other vehicles about traffic conditions. In order to cope with such misbehavior, the received data from a given source should be verified by correlating and comparing them with those received from other sources.
- **Cheating on positioning information.** Injection of a false position by a malicious vehicle pretending to be at a claimed position.
- **ID disclosure of other vehicles.** This is to track their location. A global entity can monitor trajectories of targeted vehicles and use this data for many purposes, we could take the example of some car rental companies that track their own cars.
- **Denial of Service.** Wood and Stankovic define the denial of service attack as "any event that diminishes or eliminates a network's capacity to perform its expected function" [56]. The attacker may want to bring down the Inter-Vehicular Communication (IVC) or even cause an accident. Example of attacks include channel jamming and aggressive injection of dummy messages.
- **Masquerade.** The attacker claims to be another vehicle by using false identities.

In Chapter 5, we study a state of the art algorithm for vehicular safety, and we demonstrate its vulnerability to different attacks.

1.4 Defensive Measures

A set of security primitives should be included to solve the security problems, and improve the robustness and reliability of the network. In order to create secure communication channels, a set of cryptographic primitives is needed; and the security credentials used by those primitives must be distributed using key management systems [62]. Moreover, additional services such as trust management and self-healing should exist [63, 64]. They can help to protect the core protocols of the network: routing, time synchronization, and aggregation. Finally other aspects such as distributed computing, secure location, secure mobile base station location need to be protected, if included inside a sensor network.

In this section, we describe some of the main security defensive measures ranging from key establishment, which lays the foundation for different security aspects, to more specific measures.

Key Establishment One security aspect that deserves an important attention in networks is the area of key management [65–69]. In traditional networks, key establishment is done using public-key protocols such as the Diffie-Helman protocol [70]. Most of the traditional techniques, however, are unsuitable in low power devices. Wireless sensor networks are unique in this aspect due to their size, mobility, and computational/power constraints. This, coupled with the typical operational constraints of WSNs, makes secure key management an absolute necessity in most wireless sensor network designs. The problem with

asymmetric cryptography in WSNs, is that it is typically too computationally intensive for the individual nodes in a sensor network. Different researchers show that it is feasible to implement asymmetric cryptography, with the right selection of algorithms [44,71–73]. Two of the most used techniques to implement public-key cryptosystems are RSA and Elliptic Curve Cryptography (ECC) [74]. In [73], Malan et al. demonstrate a working implementation of Diffie-Hellman based on the Elliptic Curve Discrete Logarithm Problem.

Despite the actual use of implementation of asymmetric cryptography on sensor devices, symmetric cryptography induces low computational complexity compared to asymmetric cryptography. Symmetric cryptography based schemes use a single shared key known only between the two communicating parties. One major shortcoming of symmetric cryptography is the key exchange problem: Two communicating hosts must somehow know the shared key before they can communicate securely. One case is that all nodes share the same master key, thus when a node is compromised, all the communications are also compromised. A second case, when all the pair of nodes share a different symmetric key. As a result, each node has to store a huge amount of keys. Different approaches of random key pre-distribution schemes have also been proposed to resolve the problems related to these two cases [23,54,69,75,76].

Relevant research efforts addressed the group key technique in order to localize computation, and also to decrease communication overhead in wireless sensor networks [20], for example in situational awareness, border protection, asset tracking, and digital battlefield. Key tree structure is considered in previous literature to secure group communications. Several secure group key management techniques have been proposed to support secure multicast [21,22]. In a typical group key management scheme [23], there is a trusted third party, known as Key Distribution Center (KDC). This single trusted entity is responsible for generating and distributing keys securely to the group members. In [21], authors propose a group key management scheme called Topological Key Hierarchy (TKH). It generates a key tree by using the sensor network topology with consideration of subtree-based key tree separation and wireless multicast advantage. In [26], the authors conceived a scheme using a key tree to manage group members. Their technique reposes on a centralized group rekeying scheme, and it is based on logical key tree. Their scheme is based on a group key management, that is not practical in sensor networks since it uses complex operations. Although many works in [9,11] discussed source authentication in wireless sensor networks, and especially group key for secure group communications, the problem is still challenging because we have to manage the trade-off between acceptable levels of security and, conserving scarce resources, in particular energy needed for network operations. We focus on Chapter 2 on key disclosure based schemes for source authentication service. In Chapter 4, we present security protocols based on group key management.

Defending Against DoS Attacks One strategy in defending against the classic jamming attack is to identify the jammed part of the network and effectively route around the unavailable portion. Wood and Stankovic [56] describe a two-phase approach where the nodes along the perimeter of the jammed region report their status to their neighbors who then collaboratively define the jammed region and simply route around it. At the MAC

layer, nodes might use a MAC admission control that is rate limiting to handle jamming. The network will ignore those requests designed to exhaust the power supply of a node. However, this is not fool-proof as the network must be able to handle any legitimately large traffic volume. A sending node can send the message along multiple paths to increase the likelihood that the message will eventually arrive at its destination. This has the advantage of effectively dealing with nodes that may not be malicious, but rather may have simply failed as it does not rely on a single node to route its messages.

In the transport layer, in order to resolve the flooding denial of service attack, authors in [77] propose a mechanism that uses client puzzles. The aim is to discern a node's commitment to make the connection by using some of their own resources. Aura et al. [77] suggest that a server should force a client to commit its own resources first. This strategy would likely be effective as long as the client has computational resources comparable to those of the server.

In Chapter 2, we address the memory DoS attack in WSNs. Then, we propose a new and secure broadcast authentication scheme. Our proposal uses a staggered authentication mechanism applied to WSNs. We address also three types of attacks in vehicular communications in Chapter 5, that could be classified as a Denial of Service attacks. We propose efficient countermeasures to mitigate the impact of these three attacks in Chapter 5.

Defending Against the Delayed Authentication Compromise Attack In order to be resilient against DAC attacks, the authors in [18] propose three algorithms called Partially Accountable Signature Scheme (PASS), Totally Accountable Signature Scheme (TASS), and Prioritized Totally Accountable Signature Scheme (PTASS). PASS works only in particular cases, while TASS and PTASS are more widely applicable. In fact, the main idea behind these two protocols is as follows. The time is divided in intervals. Moreover, the protocols associate the authentication key both to the time interval and to the message content. The base station builds a Direct Acyclic Graph (DAG) composed by a Merkle Tree and several Merkle-Winternitz Signatures [78]. When the base station wants to send a command, it sends several nodes of this DAG, in such a way to create a one time signature of the command. The receivers can authenticate the signature, but an attacker cannot use this signature to authenticate other messages. We address the problem of DAC attack in Chapter 3, and propose an efficient and secure scheme that is also resistant to this kind of attack.

Defending Against the Position Cheating Attack in Vehicular Communications Accurate information on position is crucial for IVC based vehicular safety applications. To this aim, detection mechanisms have been proposed in this context to recognize nodes cheating about their location [79–84]. Position verification approaches can be grouped into two main categories [79,85]: infrastructure based and infrastructure-less based approaches. The approaches using infrastructure based solutions use special hardware dedicated infrastructure to verify the position of other vehicles. In the infrastructure-less approaches, solutions use parameter based and model based approaches.

We address position cheating detection in vehicular communications in Chapter 5. We propose a new cheating position detection mechanism based on collaboration of vehicles.

Secure Broadcast Authentication Broadcast authentication from a sink or base station to sensors is the most common paradigm in Wireless Sensor Networks and of great importance, as it enables the base station to disseminate queries and messages to the sensors and thus efficiently operate the wireless sensor network. Broadcast authentication enables receivers to verify that received messages originated with the claimed source, and were not modified en-route. The problem becomes more complex in common settings, where other receivers of the data are untrusted, and where lost packets are not retransmitted. Broadcast authentication is hence one of the most important security services in such networks [2–8].

As a fundamental security service, broadcast authentication enables a sender to broadcast critical data and/or commands to sensor nodes in an authenticated way such that an attacker is unable to forge any message from the sender [9–15]. However, due to the resource constraints on sensor nodes, traditional broadcast authentication techniques such as public key based digital signatures are not desirable. Perrig et al. developed μTESLA for broadcast authentication in sensor networks based on symmetric cryptography [16], which removes the dependence on public key cryptography. Several multi-level μTESLA schemes have been proposed to extend the capability of the original μTESLA protocol [4–7]. In Chapter 2, we detail secure broadcast authentication service in Wireless Sensor Networks, and especially based on μTESLA schemes. Despite the recent advances, we show in Chapter 2, that several issues are still not properly addressed in WSNs.

- **Immediate message authentication**. μTESLA based schemes suffer from authentication delay, due to the delay of the disclosure of the authentication key. This process refers to temporal asymmetry, where the key is secret at the base station, and it will be disclosed after some time intervals (disclosure delay), and thus the key will be public. In this scenario, researches in [7, 18] prove that this kind of schemes are vulnerable against some attacks as DoS attacks [7], and Delay Authentication Compromise attack [18].

- **Resistance to DoS attacks**. An attacker may launch DoS attacks by flooding a target node with messages to overload its buffers, and to exploit the message authentication delay to forge data packets. Though several solutions have been proposed in [7], they either use substantial bandwidth or require significant resources at senders (cryptographic puzzles) [19].

- **Low communication overhead**. As devices are resource constrained, the communication overhead is of great importance. Thus, to devise an efficient secure broadcast authentication scheme in these networks, this overhead should be minimized.

1.5 Main Contributions

The main contributions of this thesis are summarized as follows.

- **Contribution 1: Broadcast Authentication in WSNs**. We identify the problem of broadcast authentication in WSNs and point out a serious security vulnerability to the symmetric-key based μTESLA-like schemes, that could lead to a memory DoS attack. We then propose several symmetric key based schemes to address the proposed problem with minimized computational and communication costs. We achieve our goal by integrating several buildings blocks such as the staggered authentication mechanism in order to reduce the number of forged packets in the receiver's buffer, and to allow the scheme to be more resistant. We devise a Bloom filter based scheme that allows a low communication overhead. We also analyze the performance and security resiliency of the proposed schemes. This contribution is discussed in Chapter 2.

- **Contribution 2: Delayed Authentication Compromise in WSNs**. First, we show that existing protocols that are resilient to DAC attack (TASS and PTASS) are subject to two possible attacks that we refer to as the drop command attack and the switch command attack. Second, we propose a new solution that is resilient against the DAC, the drop command attack and the switch command attack, and when compared to the state of the art solution is even more efficient, allowing also immediate message authentication. This contribution is discussed in Chapter 3.

- **Contribution 3: Secure Group Communications in WSNs**. The problem of providing source authentication in Wireless Sensor Networks (WSNs) has been a roadblock to their large scale deployment, and is still in its infancy. We present novel symmetric-key-based authentication schemes which exhibit low computation and communication authentication overhead. Our schemes are built upon the integration of a reputation mechanism, a Bloom filter, and a key binary tree for the distribution and updating of the authentication keys. Our schemes are lightweight and efficient with respect to communication and energy overhead. This contribution is discussed in Chapter 4.

- **Contribution 4: Securing Vehicular Safety Application**. We analyze the security of a representative algorithm Fast Multi-hop Broadcast Algorithm (FMBA) [30, 31] for state of the art IVC based safety applications, and propose countermeasures to handle the security threats. In particular, we focus on one of the main threats to safety application: the possibility to attack the protocol to impede its useful service. We focus especially on FMBA as it embodies both a state of the art solution and is a representative example of the IVC based vehicular safety applications class. Indeed, we identify the problems and propose possible countermeasures. The proposed solutions identified for FMBA can be adapted also to other protocols/algorithms, belonging to the same general class of applications. This contribution is discussed in Chapter 5.

1.6 Dissertation Outline

The organization of this dissertation is as follows. Chapter 2 presents key disclosure based broadcast authentication schemes. In Section 2.1, we introduce the notation and the background of the cryptographic mechanisms to be used. Section 2.2 presents the related work. In Section 2.3, we present the system model, adversary model, and design goals. Then, in Section 2.4, we propose two advanced schemes and detail the underlying design logic. Section 2.5 analyses the security of the proposed scheme using the AVISPA model checking tool. In Section 2.6, we evaluate the performance of the proposed schemes. In Section 2.7, we summarize the chapter.

Chapter 3 discusses delayed authentication compromise attack in WSNs. In Section 3.1, we discuss the related work. Section 3.2 presents the notation and background. In Section 3.3, we discuss the limitations and attacks for the state of the art. In Section 3.4, we propose our secure solution MASS. Sections 3.5 and 3.6 are the security and performance analysis of the proposed schemes, respectively. We summarize the chapter in Section 3.7.

Chapter 4 discusses efficient authentication schemes for group communications. Section 4.1 presents the system model. Section 4.2 is the related work. Section 4.3 details the proposed protocols. Section 4.4 presents the detailed security analysis, followed by the performance analysis in Section 4.5. In Section 4.6, we summarize the chapter.

Chapter 5 presents a secure and fast broadcast scheme for vehicular communications. Section 5.1 presents a background on fast multi-hop broadcast solutions for inter vehicular communication, and evaluates the related work in respect of these goals. Section 5.2 is the notation and preliminaries. Sections 5.3, 5.4 and 5.5 discuss the vulnerability of FMBA to three attacks: position cheating attack, replay broadcast message attack, and interrupting forwarding attack, respectively. Sections 5.6, 5.7, and 5.8 propose three countermeasures to the three attacks. Section 5.9 describes the secure proposal. Section 5.10 reports the simulation results of FMBA, FMBA under attacks, and FS-MBA. We give the summarization in Section 5.11.

Chapter 6 concludes the dissertation and offers some directions for future work that can be followed to continue researching the exposed problems.

CHAPTER 2

Key Disclosure based Source Authentication

In this chapter, we focus on key disclosure based source authentication schemes in WSNs. We demonstrate that key disclosure delay induces an authentication delay. For instance, a malicious node runs a memory denial of service attack, that leads to overload the buffer of the device, and then enforces the device to drop authenticated packets. False packets will remain in the receiver's buffer until the reception of the authenticated key. In order to mitigate this problem, we propose efficient schemes resistant to this kind of attack in WSNs. First, we propose a staggered Multi-level μTESLA called SML-μTESLA. Then, we come up with an approach based on Bloom Filter called BML-μTESLA protocol. We also validate the three key disclosure based schemes under the AVISPA model checking tool [97]. We evaluate SML-μTESLA and ML-μTESLA within the TinyOS operating system [98], TOSSIM and PowerTossim-Z simulators [99]. Our findings are that these protocols provide very good source authentication service while respecting the WSN constraints.

2.1 Notation and Preliminaries

In this section, we present the notation used throughout this chapter and some preliminary notions.

2.1.1 Notation

We use the following notation:

- S: Sender
- R: Receiver
- I: Intruder

- T_i: The i^{th} time interval corresponding to the generation of the i^{th} key of a high level key chain
- $T_{i,j}$: The time interval corresponding to the i^{th} high level key and the j^{th} low level key
- t_0: The start time
- $Time$: The current Time
- $M_{i,j}$: The j^{th} message in time interval T_i
- $M^{'}$: A primed variable $M^{'}$ always means the new value of M
- $M_1.M_2$: Message M_1 concatenated with message M_2
- CDM: Commitment Distribution Message
- inv: Inverse of a key. Given a public key returns private key
- MAC: Message Authentication Code
- $Hash$: Message authentication code function
- d: The disclosure key delay
- $S \rightarrow R$: M; Message M sent from S to R
- $K_{i,j}$: The authenticated key of the i^{th} low level chain in the j^{th} time interval
- K_i: The authenticated key corresponding to the high level chain in time interval T_i
- $K_{i,0}$: The first key of the i^{th} low level chain or the first key commitment of the i^{th} low level chain
- n_1: The number of keys in the low level key chain
- n_0: The number of keys in the high level key chain
- $Iknowledge$: Intruder Knowledge
- M_k: Message encrypted with key k
- K_{prevc1}: Set initially to the first key of the high level chain, and then it indicates the current key
- K_{prev2}: The value of the disclosed key in the received packet
- $K_{published}$: The disclosed key
- K_{prev}: It indicates the previous key, set initially to the first key of the low level key chain

- $act1$ /\ $act2$: This operator indicates in HLPSL language that these two actions $act1$ and $act2$ are operated simultaneously
- PRF: Pseudo Random Function
- d: The disclosure key delay. It is represented on terms of number of time intervals.

2.1.2 Preliminaries

In the following, we give a brief overview of the Bloom Filter data structure, and the key chain.

Bloom Filter A Bloom filter is a space-efficient data structure for representing a set. This structure is used in order to test membership queries. In order to represent a set $E = e_1, e_2, ..., e_n$ of n elements, a Bloom vector B of m bits can be used. The m bits are initially all set to 0. Moreover, this structure needs k independent hash functions h_1,..., h_k. These k hash functions range between 0 and $m - 1$, and each element is mapped to [0, ..., $m - 1$]. For each element e in E, the bits $h_i(e)$ are set to 1 for $1 \leq i \leq k$. In order to verify if an item e is in E, we test whether all bits $h_i(e)$ are set to 1. If yes, e is assumed to be a member of E. If not, e is not a member of E. A Bloom filter may suggest that an element e is in E even though it is not [100]. Figure 2.1 illustrates an example of Bloom Filter insertion. To query the membership of an item e' within E, the bits at indices $h_i(e')$ $(1 \leq i \leq k)$ are checked. If any of them is 0, then certainly $e' \notin E$. Otherwise, if all the bits are set to '1', $e' \in E$ with high probability. There is a possibility of error which arises due to the hashing collision that makes the elements in E collectively causing indices $h_i(e')$ being set to 1 even if $e' \notin E$. This is called a false positive. Note that there is no false negative in the Bloom Filter membership verification. The probability of a false positive [100] f_{prob} is then approximated to:

$$f_{prob} = (1 - \mathrm{e}^{\frac{-k \times n}{m}})^k$$

Key chain and its Application in Resource Constrained Networks One way key chain is a cryptographic primitive. The first use of this technique was for one time passwords [101]. This primitive is used in S/KEY one time password system [102]. After that, one-way chains are also used in many other applications. Figure 2.2 illustrates the generation of a one-way chain. In order to generate this chain, we randomly pick the last element of the chain, and we generate the chain by repeatedly applying a one way function f. Finally, v_0 is a commitment to the entire one-way chain, and we can verify any element of the chain through v_0. For example, in order to verify that an element v_i is indeed the element with index i of the hash chain, we check that $f^i(v_i) = v_0$. We reveal the elements of the chain in this order v_0, v_1, v_2, and v_n. The storage of this chain could be done in two manners. The first one is to create it all at one and store each element of the chain. The second manner

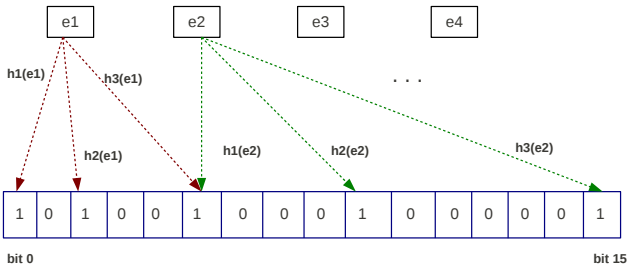


Figure 2.1. – A Bloom Filter with $n = 4$, $m = 16$ and $k = 3$

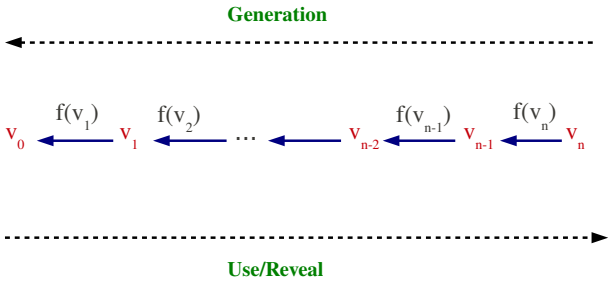


Figure 2.2. – An example of a One Way Key Chain

is to store the current key v_i, and compute any other key on demand. An hybrid approach was proposed to reduce the storage with a small computation penalty. The authors in [103] devise a storage efficient mechanism for one way chains. In fact, a one way chain with N elements only requires $\log(N)$ storage and $\log(N)$ computation to access an element.

2.2 Related Work

Security concerns constitute a potential stumbling block to the impeding wide deployment of sensor networks. Providing source authentication service in such networks is of

great importance. In fact, this security service enables a receiver of a message to confirm whether the message is from the pretended source. Source authentication has been recently suggested as an effective security service, and considerable research has been done on providing source authentication in WSNs [9–15,104]. Source authentication protocols presented in the literature have described different methods of carrying the authentication information of a message. In fact, the proposed solutions could be classified into two categories.

The first category is the signature based schemes [3, 105–109], which requires the use of asymmetric cryptographic primitives and pairing operations [110]. Public Key Cryptography (PKC) such as Elliptic Curve Cryptography (ECC) has been proposed for solving the problem of source authentication in WSNs. ECC is attractive for constrained wireless devices because the smaller keys result in memory, bandwidth, and computational savings [104]. A well-known key exchange algorithm for ECC is the Elliptic Curve Diffie-Hellman (ECDH) algorithm and ECC-based signatures can be generated and verified with the Elliptic Curve Digital Signature Algorithm (ECC-160) [111]. However, optimized signatures based on ECC and Identity schemes [106,112] suffer from high energy consumption as well as significant communication and computation costs. In fact, these schemes require two time consuming operations that are the point multiplication and pairing operation. The pairing operation has been used for securing resource-constrained sensor networks but it seems to be one of the most expensive operations in terms of computational complexity and memory requirements [113, 114]. Many Identity (ID) based signatures [106] require weil pairing [114] or tate pairing [113] computation which leads to a high computation cost and thus to a high level energy consumption. ECC based ID-Signature does not require pairing computations and uses only the ECC operations. Bellare-Namprempre-Neven signature (BNN-IDS) is further adapted to wireless sensor networks, and it needs three point multiplications for signature verification [115]. Further, Elliptic Curve Digital Signature Algorithm (ECDSA) [116] offers a moderate computation cost since it requires two point multiplications in order to verify signature. Thus, even though many optimized signature schemes are efficient, they are still unsuitable for most sensor network architectures. On the other hand, while signing each data packet provides good source authentication; it has high overhead, both in terms of time (to sign and verify), and in terms of bandwidth. This kind of solutions could be easily exploited for example by a malicious node to enforce the device doing highly cost false signature verifications, leading to computation Denial of Service attack. In particular, the battery of the device will be depleted in a very short time interval.

The second category of solutions is the symmetric key based schemes [5, 11, 13, 117, 118]. In particular, time asymmetry based schemes are good alternative to provide source authentication of messages in these constrained networks. μTESLA [13] is the first solution based on key disclosure delay in WSNs. This protocol requires symmetric cryptographic primitives and it is an adaptation to the TESLA protocol [49]. It requires a one way hash chain in order to generate authentication keys [49]. μTESLA uses one MAC attached to each packet which is generated by a new key disclosed at a later point in time. On receipt of such a message, a recipient stores the packet, waits for the disclosure of the key and

verifies its validity. If this verification fails, the packet is discarded. Moreover, μTESLA sends the key chain commitments, starting time, and duration of each time interval, using unicast, which is not scalable in large sensor networks. Furthermore, μTESLA suffers from authentication delay which is a vulnerability to many DoS attacks which over consumes scarce energy resources.

In order to enhance the scalability of μTESLA, multi-level μTESLA was proposed [5]. It consists of a multi-level key chain in order to be applied in a large WSN. Each chain in a higher level is used to authenticate the commitment (the first key) of the lower level, and the lowest level is used to authenticate the message. Multi- level key chains are used to extend the lifetime of authenticated broadcast [5]. Like μTESLA, multi-level μTESLA suffers from authentication delay of the data packets. In order to support a large number of broadcast senders, Merkle Tree-μTESLA [119] uses Merkle Tree using μTESLA instances as building blocks [120]. Though Merkle Tree-μTESLA is scalable in terms of senders, the revocation and addition of new senders is very hard since it needs the update of the Merkle Tree. Another approach uses compressed Bloom filter in order to authenticate the parameters of instances of μTESLA and supports many senders [121].

In [122], the authors use symmetric cryptography to build a batch broadcast authentication and present their protocol BABRA (Batch Broadcast Authentication). In fact, a batch is a burst of sequenced packets. Since BABRA eliminates the use of key chain, the keys are independent. Therefore, when the batch key is lost, BABRA requires mechanisms that provide resilience to the key loss; which is easily provided in μTESLA based schemes.

MultiMAC [123] is a broadcast source authentication mechanism based on multiple message authentication codes. It proposes a deterministic combinatorial key distribution scheme that provides source authentication service. MultiMAC offers the immediate authentication scheme. However, it suffers from the communication overhead per packet since a packet has to concatenate multiple message authentication codes, and also raises some scalability concerns.

Hence, we can summarize that the key problem of the μTESLA based protocols is related to the authentication delay. Real time applications suffer from this authentication delay, and also if we consider a malicious node in the network, it can exploit the authentication delay of the received messages, to forge messages, and broadcast them, thus leading to a denial of service attack that affects the memory of the device.

Many approaches were proposed in order to reduce this delay. For instance, the authors in [124] have proposed the immediate authentication mechanism in TESLA. A hash image of the content of each packet is concatenated in an earlier packet. However, this mechanism is not efficient since the source has to have a unique sending rate. Moreover, when a packet is lost, it cannot be immediately authenticated.

Another approach was proposed in [4] aiming to authenticate immediately the packets and not to wait for d time intervals. Note that d represents the key disclosure delay. This approach consists of using a message specific puzzle. This message is obtained by appending the μTESLA packet which is in fact a weak authentication mechanism. The added information consists of a hash value of a puzzle key and the initial μTESLA packet.

This is certainly an efficient approach to mitigate DoS attacks in wireless sensor networks but it requires a sender with high power.

Moreover, in order to reduce the authentication delay, the authors in [125] propose staggered TESLA. This mechanism consists of appending to the packet P_i sent in interval T_i, d MACs of the message generated by all the d keys in T_i until T_{i-d-1}. Staggered TESLA reduces the probability that a forged message remains in the receiver's buffer for d intervals. Thus, this approach reduces delay and enhances DoS resistance.

In Table 2.1, we summarize some of the state of the art protocols based on key disclosure for source authentication in WSNs. We evaluate these protocols in terms of security (authentication delay and resilience to DoS attacks), communication overhead, and scalability. T_{int} represents the time interval, t_{auth} is the real authentication time of the message, t_r is the receiving time of the message, W is the authentication delay, and k_i is the authentication key in time interval T_i.

Protocol	**Security Analysis (Authentication Delay)**	**Communication Overhead**	**Scalability**
μTESLA [13]	$W = \|t_{auth} - t_r\| = \mathrm{d}\times\ T_{int}$	$\|MAC\|+\|k_i\|$	low sender and receiver scalability
Multi-level μTESLA [5]	$W = \|t_{auth} - t_r\| = \mathrm{d}\times\ T_{int}$	$\|CDM_i\|+d*\|MAC\|+\|k_i\|$	low sender scalability
Merkle Tree μTESLA [119]	$W = \|t_{auth} - t_r\| = \mathrm{d}\times\ T_{int}$	$1 + \|log2(N)\|+\|MAC\| + \|k_i\|$	low receiver scalability
BABRA [122]	$(C + D)$ time units, with C is the batch period and D is the delay period	$\|h(k_i - 1)\| + \|MAC\| + \|index\|$	low sender and receiver scalability

Table 2.1 – Overview of some state of the art key disclosure schemes in WSNs.

In the following, we give an overview of μTESLA, Multi-level μTESLA, and also staggered authentication mechanism.

2.2.1 Overview of μTESLA

In this section, we detail the description of μTESLA. In fact, this protocol has three phases: i) sender setup; ii) message sending procedure executed by the broadcaster; and iii) message receiving procedure executed by the receiver. For more details, we refer the reader to the work in [13].

Sender Setup In μTESLA, the sender generates a key chain, which is a sequence of secret keys. The generation of the one-way key chain of length n is as follows. First, the sender chooses randomly the last key (denoted here by K_n), and then generates the remaining

values by applying successively a one-way function f (e.g., a cryptographic hash function as $MD5$), $K_i = f(K_{i+1})$. As f is a one-way hash function, every device can compute forward, e.g., compute K_0, K_1, ..., and K_j given the value of key K_{j+1}. However, no one could compute backward, e.g., compute K_{j+1} given only K_0, K_1, ..., and K_j. This is due to the one-wayness of the hash function f.

Message Sending Procedure Executed by the Broadcaster In μTESLA, the time is divided into time intervals and the sender (the broadcaster) associates each key of the one-way key chain with one time interval (see Figure 2.3). Let us assume that the sender wants to send a message at time interval T_i. In this case, the sender generates a packet adding the message, the message authentication code (MAC) of that packet using the authentication key K_i. We should keep in mind that this authentication key is still secret in that time interval (T_i). This key will only be disclosed after a delay d (few time intervals, which is in general superior than the round trip time between the sender and the receiver). That is why μTESLA based schemes are classified as temporal asymmetry approaches. The data packet format using μTESLA is as follows: $< M|MAC(K_i, M)|K_{i-d} >$; with the $'|'$ symbol denotes message concatenation, M generated message at time interval T_i, $MAC(K_i, M)$ is the authentication information using the key K_i.

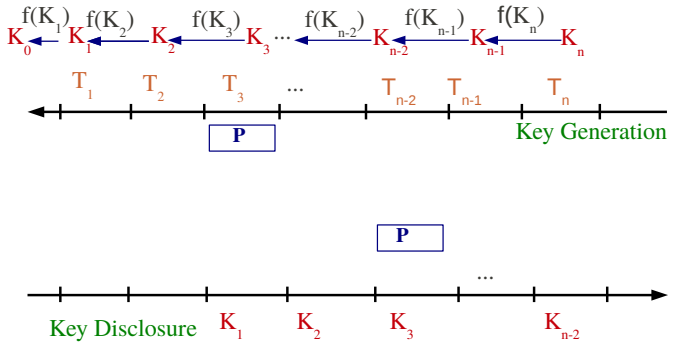


Figure 2.3. – μTESLA Scheme

Message Receiving Procedure executed by the receiver An emergent property of one-way key chain is that, once a receiver has an authenticated key of the chain, then it could reveal subsequent keys of this chain, by applying the one-way function f. This means that when the receiver has an authenticated value K_i of the key chain, it can easily authenticate K_{i+1}, by verifying $K_i = f(K_{i+1})$. Therefore, a receiver should have one authenticated key of the chain as a commitment to the entire chain. Moreover, in μTESLA based schemes,

nodes should be loosely synchronized. Initially, all nodes know the key disclosure schedule of keys of the one way key chain.

Authenticating broadcast packets When a receiver receives the packets, it needs to ensure that the packet could not be spoofed by an adversary. To verify this condition, the receiver should check for each incoming packet that the sender did not yet disclose the key which corresponds to the packet, which means that no adversary could have forged the packets. In order to verify this security condition, sender and receivers should be loosely synchronized, and that the key disclosure delay is known by all the participants in the network (sender and receivers).

When a receiver receives an incoming broadcast packet in time interval T_i, it checks the following security condition: $\frac{T_c+\delta-T_0}{T_{int}} < T_i+d$, where T_c is the local time when the packet is received, T_0 is the start time of the first time interval, T_{int} is the duration of each time interval, δ is the maximum clock difference between the sender and itself, d is the disclosure key delay. Moreover, d is represented in terms of number of time intervals. For instance, when the node receives a broadcast packet in time interval T_3 ($i = 3$), and d= 2. It has to check whether $\frac{T_c+\delta-T_0}{T_{int}} < 3+2$. In the case when the packet verifies the security condition, the receiver will store it in its buffer. The receiver could verify this packet only when receiving the corresponding key. If the security condition is violated, which means that the packet had an unusually long delay, thus the receiver drops the packet since an adversary might have altered it. When receiving a key K_i of a previous time interval, the receiver verifies its authenticity by checking whether this key matches the last authenticated stored key K_j, by verifying $K_i = f^{|j-i|}(K_j)$. If the check is successful, the new key K_i is authentic and the receiver can authenticate all packets that were sent within the time intervals T_i to T_j. The receiver also stores the key K_i instead of key K_j.

μTESLA is an extension to TESLA. The only difference between these two protocols is in their key chain commitment distribution schemes. TESLA uses asymmetric cryptography to boostrap new receivers, which is impractical for resource constrained devices due to its high computation and storage overheads. μTESLA depends on symmetric cryptography with the master key, shared between the sender and each receiver, to bootstrap the new receivers individually. In this scheme, the receiver first sends a request to the sender, and then the sender replies with a packet containing the current time T_c for time synchronization, a key K_i of one way key chain used in a past interval, the start time of the time interval T_i, the duration T_{int} of each time interval and the disclosure delay d.

2.2.2 Overview of Multi-level μTESLA: ML-μTESLA

Multi-level μTESLA [5] (ML-μTESLA) is a source authentication protocol, based on multi-level key chains. It is an extended version of μTESLA. However, in μTESLA, there is a difficulty to distribute the key chain commitments to a large number of sensor nodes. In particular, the method of bootstrapping new receivers in μTESLA does not scale to a large WSN. In [5], the authors claim that there is a mismatch between the unicast-based

distribution of key chain commitments and the authentication key of broadcast messages in μTESLA. The transmission of the initial parameters is based on unicast, however the technique is intended for broadcast authentication. The goal of Multi-level μTESLA is to enhance the scalability of μTESLA, so that it could be applied to large WSNs.

In the following, we first detail the mechanism of multi-level-key chain. Then, we discuss the steps of execution of the protocol that are: the setup phase, the sending message phase, and the receiving message phase.

Multi-level key chain The concept of Multi-level key chain has been first introduced by [5]. As μTESLA uses a unicast transmission to send the initialization parameters (for bootstrapping new receivers), this concept is not viable when dealing with broadcast transmission [13]. To mitigate this problem, the authors in [5] use multi-level key chain to distribute the commitment of one-way chain.

Setup Phase In the setup phase, the sender/broadcaster generates a multi-level key chain. This chain is a set of one-way key chain with different levels.

The low level key chains are intended for authenticating broadcast messages, while the high level key chain is used to distribute and authenticate commitments (first key) of the low-level key chains. The high level key chain uses a long enough interval to divide the time line into equal time intervals, so that it cover the lifetime of a sensor network, without having too many keys. The low level key chains have short enough intervals.

The lifetime of a sensor network is divided into n_0 (long) intervals. In Figure 2.4, $n_0 = 3$. The high level key chain has four elements K_3, K_2, K_1, and K_0, which are generated by randomly picking K_3 and computing $K_i = f(K_{i+1})$ for $i = 0, 1, .., n_0 - 1$, where f is a pseudo-random function. The key K_i is associated with each time interval T_i. The disclosure of the authentication key K_i is disclosed in time interval T_{i+1} since the high level time interval is usually very long compared to the network delay and clock discrepancies. As in μTESLA, the security condition to check whether the base station has disclosed the key K_i when a sensor node receives a message authenticated with key K_i at time T_i is as follows: $\frac{T_c+\delta-T_0}{T_{int}} < T_i+d$; where T_c is the local time when the packet is received, T_i is the i^{th} time interval, T_0 is the start time of the first time interval, T_{int} is the duration of each time interval, δ is the maximum clock difference between the sender and itself, and d is the key disclosure delay. In case of high level key chains, the disclosure delay is represented in terms of number of high level time intervals. In general, the high level key is disclosed in the next time interval, since the high level time interval is usually very long compared to the network delay and clock discrepancies ($d = 1$). For instance, let us assume a scenario where a packet is received at T_3 (the third time interval), the local time of the receiver is T_c, and having received the values of δ, T_0, T_{int}, and d. In this case, the security condition is as follows: $\frac{T_c+\delta-T_0}{T_{int}} < 3+1$.

Each time interval T_i is further divided into n_1 (short) time intervals of equal duration, denoted as $T_{i,1}$, $T_{i,2}$, ..., and $T_{i,n1}$. The base station generates a low level key chain for each time interval T_i by randomly picking $K_{i,n1}$ and computing the remaining keys by

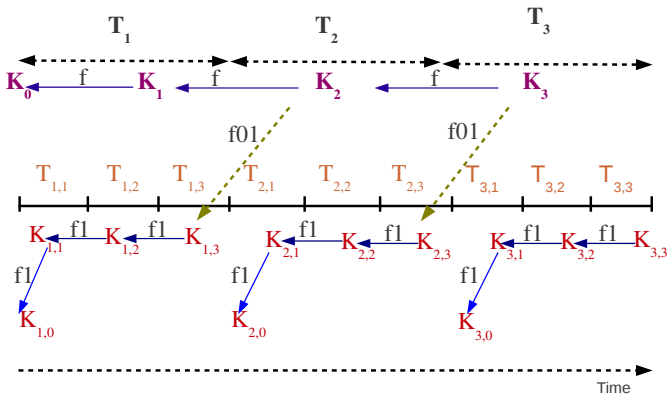


Figure 2.4. – An example of two level key chain mechanism ($n_0 = 3$, and $n_1 = 3$)

applying a pseudo-random function $f1$. The key $K_{i,j}$ is intended for authenticating messages broadcasted during the time interval $T_{i,j}$. The starting time of the key chain $< K_{i,0} >$ is predetermined at $T_{i,j}$. The disclosure delay for the low level key chains can be determined in the same way as in μTESLA. We assume that all the low-level key chains use the same disclosure delay d. In case of low level key chains, d is represented on terms of number of low level time intervals. When $d = 3$, it means that the low level key will be disclosed after three low level time intervals. When sensor nodes are initialized, their clocks are synchronized with the base station.

In addition, the base station distributes to the sensor nodes the following parameters: the starting time, the commitment K_0 for the high-level key chain, the duration T_{int} of each low-level time interval, the duration of each high level time interval, the disclosure delay for the low-level key chains, and the maximum clock discrepancy between the base station and the sensor nodes throughout the lifetime of the sensor network. In order for the sensor nodes to use a low level key chain during the time interval T_i, they must authenticate the commitment (first key of the chain) $K_{i,0}$ before the start time of T_i. To achieve this goal, the base station broadcasts a commitment distribution message, denoted as CDM_i, during each time interval T_i, with $CDM_i = i|K_{i+2,0}|MAC(K_i^{'}, i|K_{i+2,0})|K_{i-1}$ where the $'\ |'$ symbol denotes message concatenation, and $K_i^{'}$ is derived from key K_i with a pseudo random function other than f and f_1. Thus, to use a low-level key chain during T_i, the base station needs to generate the key chain during T_{i-2}, and distribute $K_{i,0}$ in CDM_{i-2}.

In particular, instead of choosing each key K_{i,n_1} randomly, each K_{i,n_1} is derived from a high-level K_{i+1} (which is used to be in the next high-level time interval) through another pseudo-random function $f01$. That is $K_{i,n_1} = f01(K_{i+1})$.

Message Sending Procedure Executed by the Broadcaster Let us consider that the base station needs to send a data packet at time interval $T_{i,j}$. The format of the data packet is as follows: $P = level\_number|index|M|MAC(K_{i,j}, M)|K_{i,j-d}$ (2) where $level\_number$ represents the level of the hash chain, $index$ is the index of the packet, M is the message generated at time interval $T_{i,j}$, $K_{i,j-d}$ represents the key corresponding to time interval $T_{i,j-d}$, and d represents the key disclosure delay for low level key chains.

Message Authenticating Procedure Executed by the Broadcaster When receiving a CDM packet, the receiver does the following operations: (i) first, it needs to be ensure that the packet could not be spoofed by an adversary. To verify this condition, the receiver should check for each CDM packet that the sender did not disclose the key; ii) second, it should verify the authenticity of the received key by comparing it to the last stored authenticated key; and iii) third, if the check is successful, the new key is authentic and the receiver can authenticate CDM packets.

When receiving a data packet, the receiver does the following operations: i) first, it checks whether the sender did not disclose the received authentication key (as in μTESLA); ii) second, it verifies the authenticity of the received (disclosed key) by comparing it to the last stored key; and iii) third, if the key is authenticated, then the new key could be used to authenticate received data packets.

2.2.3 Staggered Authentication

Staggered authentication was first proposed in [125] to deal with reducing delay associated with multicast authentication. The goal of this approach is to make the usage of receiver-side buffers more efficient, and nodes more resilient to buffer overflow denial of service attacks. The mechanism is applied to TESLA scheme, and employs several message authentication codes (MACs) that correspond to authentication keys that are staggered in time. The added MACs provide partial authentication, and the complete authentication is done when all the MACs added to the packet are correct.

To broadcast message M_j in interval T_i, the sender constructs packet P_j. In fact, a packet P_j used in TESLA has the following format:

$$P_j = M_j|MAC(K_i, M_j)|K_{i-d}$$

with M_j is a message generated at time interval T_i, K_{i-d} is a key seed disclosed, that corresponds to time interval T_{i-d}. This key will be used to authenticate packets that are sent at time interval T_{i-d}, and d is the disclosure delay.

Using a staggered authentication mechanism, the modified j^{th} data packet generated in T_i is constructed as follows:

$$\begin{aligned} P_j = & M_j|MAC(K_i, M_j)|MAC(K_{i-1}, M_j)| \\ & MAC(K_{i-2}, M_j)|...|MAC(K_{i-d-1}, M_j)|K_{i-d} \end{aligned}$$

with K_{i-d} is a key disclosed at time interval T_i. In this scheme, the sender adds d message authentication codes to the message. On receiving a packet, the node will start authenticating the packet and not to wait for d time intervals in order to confirm whether the packet is correct or not. The partial authentication allows the receiver to remove the incorrect packets as soon as possible (after verifying at each time interval whether the corresponding MAC is correct or not). In fact, adding d MACs to the packet, will allow the receiver to authenticate partially if the corresponding MAC (using the disclosed key at that time interval) is correct or not. If it is not correct, the packet will be removed from the receiver's buffer, otherwise, the packet has a high probability to be an authenticated packet. The receiver can confirm that the received packet is authenticated until the verification of all the added MACs, and when receiving the final disclosed key K_i.

2.3 System Model

In this section, we present the architectural system, the adversary model, as well as design goals.

2.3.1 Architectural System

We consider a large spatially distributed sensor network, consisting of one Base Station and a large number of sensor nodes. The sensor nodes are resource-constrained with respect to memory space, computation capability, bandwidth, and power supply. The Base Station is assumed to be more powerful than sensor nodes in terms of computation and communication capabilities. The Base Station broadcasts queries or commands through sensor nodes, and expects replies that reflect the latest information or measurements. We also assume that the WSN is loosely synchronized and that the Base Station is always trustworthy but the sensor nodes are subject to compromise.

2.3.2 Adversary Model

We focus on Denial of Service (DoS) attacks such as bogus message flooding, aiming at exhausting constrained network resources. We assume that the adversary is able to compromise a limited number of sensor nodes i.e., the adversary cannot compromise an unlimited number of sensor nodes.

2.3.3 Design Goals

Our primary security goal is that all messages broadcast by the base station are authenticated, so that the bogus ones inserted by the compromised sensor nodes can be efficiently rejected. We also focus on minimizing the overhead of the security design.

2.4 Our Advanced Schemes

Key disclosure based schemes have a major pitfall which is the authentication delay. This could be exploited by an attacker to handle various attacks, especially Denial of Service attacks in these constrained devices. Analyzing the impact of this dreadful attack on WSNs, we aim to reduce the authentication delay, and therefore reducing the forged delay in the receiver's buffer. In this section, we present our proposed schemes. The first scheme called staggered Multi-level-μTESLA consists on using the mechanism of staggered authentication in the Internet, and applying it to a protocol for Wireless Sensor Networks. The second scheme enhances our first proposal by using a Bloom filter, in order to reduce the communication overhead of all the packets.

2.4.1 Staggered Multi-level-μTESLA

The aim of our proposal is to reduce the delay of forged packets in the receiver's buffer in WSNs. In the following, we present a brief overview and a detailed description of our proposal.

Overview

Deploying key disclosure delay based schemes in the resource constrained devices as WSNs is of great importance. However, the delay induced by these schemes could be exploited by an attacker to handle various attacks, especially Denial of Service attacks, and to lead to buffer overflow in these devices. In order to mitigate against these attacks, we refer to the principle of partial authentication and apply it to the resource constrained devices. To this aim, we focus on applying this principle to Multi-level μTESLA, which is a scalable source authentication protocol in WSNs. Our proposal is called Staggered Multi-level μTESLA (SML-μTESLA). As Multi-level μTESLA, this scheme consists of splitting the transmission time into equal length intervals and generating an authentication key corresponding to packets sent in a given time interval. This authentication key is derived using a publicly available one-way function. Staggered Multi-level μTESLA is based on MACs from successive Multi-level μTESLA keys. Many adversaries will not be able to forge all the MACs constructed as opposed to the single-MAC-based scheme.

Description

In the following, we describe the set up phase, the transmission phase, as well as the verification phase of staggered Multi-level μTESLA.

Set Up Phase As Multi-level-μTESLA, the base station sends the initialization parameters to all the receivers in a secure channel after establishing the multi-level key chains: start

time, number of high level time intervals, number of low level time intervals, the first key of the high level chain, the first key of the low level chain, the disclosure delay of low level key chains, the disclosure delay of high level key chains (which is here one time interval), and the maximum clock discrepancy.

Transmission Phase When the base station needs to send a packet at time interval $T_{i,j}$, it needs to create a data packet P as follows.
$P = level\_number|index|M|MAC(K_{i,j}, M)|MAC(K_{i,j-1}, M)|\ MAC(K_{i,j-2}, M)|\ ...|$
$MAC(K_{i,j-d-1}, M)|K_{i,j-d}$
The sender adds d message authentication codes to the message. Each message authentication code will be authenticated at each time interval (when receiving the corresponding authenticated key). Let us assume that the sender sends the packet P in time interval $T_{i,j}$. In order to authenticate this packet, the sender adds d message authentication codes using the following keys $K_{i,j}$, $K_{i,j-1}$,..., and $K_{i,j-d-1}$. When receiving this packet, the receiver will buffer it. At time interval $T_{i,j+1}$, the receiver will receive the disclosed key $K_{i,j-d-1}$ from another packet that discloses this information. Using the key $K_{i,j-d-1}$, the receiver can authenticate the packet P with doing a partial authentication to verify the corresponding message authentication code ($MAC(k_{i,j-d-1}, M)$). Then, if the verification does not hold (which means the packet is not correct), the receiver will remove the received authenticated packet in that time interval. Otherwise, the packet will still in the receiver's buffer until the reception of another disclosed key that serves to authenticate the other message authentication code. The complete authentication of the packet will be done until the reception of the disclosed key $K_{i,j}$. In that time, the receiver can confirm whether the packet is correct or not. This method will reduce the number of forged packets in the resource constrained devices.

In SML-μTESLA, the receiver does not have to wait for d time intervals in order to start authenticating packets. In fact, the receiver can use any received keys to begin the authentication process and can thus promptly remove bogus packets. Hence, the number of forged packets in the receiver's buffer is decreased and the scheme is more resistant to DoS attacks.

The following Figure 2.5 represents an example of SML-μTESLA mechanism between the sender and the receiver. In fact, the receiver does not have to wait for three time intervals (in this example, $d = 3$) to start authenticating packets (packet $P_{2,3}$ in Figure 2.5). The receiver can use a received key in a time interval to start the authentication process, and can thus promptly remove $P_{2,3}$ if it is a bogus packet.

SML-μTESLA is efficient since it is based on symmetric cryptography. Thus, the additional computation and communication requirements introduced by the extra MACs will not cause significant performance degradation. When it receives a packet, the receiver puts the packet at the head of the queue, and degrades the packet to lower layers as additional keys arrive and the corresponding MACs are verified. If the verification fails, the packet is dropped from the queue. When the final key involved arrives and the corresponding MAC is verified, then complete authentication is achieved.

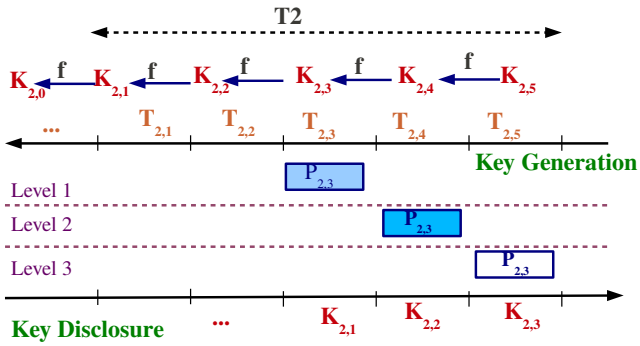


Figure 2.5. – SML-μTESLA: An example of execution

2.4.2 Bloom Filter Multi-level-μTESLA

Our second proposal consists on devising a protocol that offers low cost, by using a Bloom filter data structure to map the multiple message authentication codes.

Overview

In SML-μTESLA, a data packet generated in $T_{i,j}$ is constructed as follows:
$P = level\_number|index|M|MAC(K_{i,j}, M)|MAC(K_{i,j-1}, M)|MAC(K_{i,j-2}, M)|$
$..|MAC(K_{i,j-d-1}, M)|K_{i,j-d}$
Thus, the sender uses multiple message authentication codes in order to have a more resistant scheme to DoS attacks. In order to enhance the performance of our scheme, we use a Bloom Filter vector.

Description

Our second proposal is called BML-μTESLA, which is an extension of Staggered Multi-level μTESLA. The major goal of BML-μTESLA is to reduce the communication overhead. In fact, in time interval $T_{i,j}$, the sender generates d message authentication codes for each data packet, and constructs the set E. For each data packet, we have:

$E = < MAC(K_{i,j}, M), MAC(K_{i,j-1}, M), .., MAC(K_{i,j-d-1}, M) >$
We have to map the elements of E (each with $|MAC|$ bytes) to an m-bit vector B with $B = b_0, b_1, ..., b_m$. Therefore, we have $m < d \times |MAC|$ to reduce the filter size and

$m > k \times d$ to have a small probability of a false positive [126]. Note that k represents the number of hash functions used in the Bloom filter. These k hash functions are known by every node as well as by the base station. For the sake of clearness, let us assume that the set of d MACs is denoted as follows: $< MAC_1, MAC_2, MAC_3, ..., MAC_d >$. Initially, all the bits of the Bloom Filter B are initialized to '0'. After that, the Base station fills the Bloom Filter as follows: For each item (MAC) on E, if there exists $h_l(MAC) = i$, then the i^{th} bit of B is set to '1', ($b_i = 1$).

In our case, we used a space-efficient Bloom filter [126] which is a randomized data structure often used to represent a group of elements (d MACs) with a small false positive rate. That is, the Bloom filter saves space at the cost of representing a slightly larger group, than the original group of elements.

Choice of the Parameters of the Bloom Filter Given the number d of MACs generated and concatenated in one packet, and the storage space of m bits for a single Bloom filter, the minimum probability of a false positive [100, 126] f_{prob} that can be achieved is

$$f_{prob} = (0.6185)^{(\frac{m}{d})}$$

The probability of a false positive decreases as the fraction $\frac{m}{d}$ increases. If we take for example $m = 16$ bits, $k = 3$ and $d = 3$, then the minimum probability of a false positive is $f_{prob} = 0.077$. To send a data packet P during a time interval $T_{i,j}$, the sender generates d MACs which are then mapped to a Bloom filter vector $b_{i,j}$. The data packet corresponding to a message M in time interval $T_{i,j}$ is then constructed as follows:

$$P = level\_number|index|M|b_{i,j}|K_{i,j-d}$$

Figure 2.6 illustrates how d MACs of each data packet are mapped into a Bloom Filter vector using k hash functions.

On receiving the data packet, the receiver tries to do the following operations:

- verify if the number of '1' bits is less than or equal to $d \times k$ bits in the vector. If it is not, the packet is dropped; else the receiver computes the message authentication code MAC' with the correspondent key.
- verify if the computed message authentication code is in the Bloom filter $b_{i,j}$. In fact, for each hash function h_i (with $1 \leq$ i$\leq k$) used in the Bloom filter, it verifies if $h_i(MAC')$ is between 0 and $m-1$. If all the corresponding bits in the vector are set to one, then the packet is assumed to be partially authenticated. The packet is degraded to the lower levels of the buffer until all the correspondent MACs are verified, when receiving the corresponding keys in further packets as in SML-μTESLA. Else, the verification fails and the packet is dropped.

Our solution BML-μTESLA inherits the efficiency and security of SML-μTESLA. In particular, it inherits the resistance to packet loss. Further, the forged packets have a high

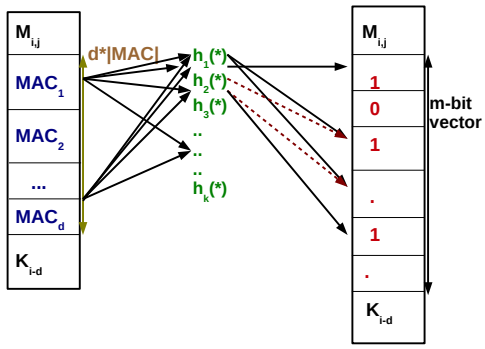


Figure 2.6. – Mapping d MACs into a Bloom filter Vector

probability to be dropped before the expiration of the authentication delay. Furthermore, our protocol reduces the communication cost and the total energy overhead by using a Bloom filter.

In the following section, we present the model checking tool AVISPA, that we use for verifying the three protocols (ML-μTESLA, SML-μTESLA, and BML-μTESLA).

2.5 Formal Validation

Formal validation tools are used in order to show the safety of a protocol. To this aim, there are many formal validation tools that are proposed in the literature [127–131]. We aim to formally verify and model ML-μTESLA, SML-μTESLA, and BML-μTESLA using the AVISPA model-checking tool and HLPSL language [97, 131]. We show that those protocols can safely be used for source authentication. We also demonstrate that those three protocols exhibit also some attack problems which are hard to eliminate. A security protocol must fulfill authentication, confidentiality, integrity, and non-repudiation, etc. Errors revealed after deployment could appear to be catastrophic. Thus, validation before implementation is crucial. Validation requires the development of many formal methods to investigate pros and cons of any protocol.

2.5.1 Overview of AVISPA

We used the AVISPA toolbox which is devoted to the verification of security protocols. In fact, AVISPA was used earlier to verify WSNs. In [132], the authors analyze the TinySec authentication protocol with AVISPA. They report a man-in-the-middle attack for pairwise-key establishment.

AVISPA provides a High-Level Formal Specification Language (HLPSL) [97] for specifying protocols and verifying their security properties. Once the model of the protocol is

specified, AVISPA translates it into an Intermediate Format (IF). This intermediate format consists on the input of many back-ends that are integrated into AVISPA: SATMC (SAT- based Model-Checker), OFMC (On the-Fly-Model-Checker), Cl-Atse (Constraint-Logic-based Attack Searcher) and TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols).

In order to specify a protocol in AVISPA, we present two kinds of roles: basic roles and composed roles. A basic role describes participant initial knowledge, the initial state, the set of transitions and events of each participant. On the other hand, composed roles describe protocol sessions. Several intruder models are implemented and each model has its specificities. The intruder is modeled by the channels over which the communication takes places. The Dolev-Yao intruder model is the strongest possible model of the capabilities of an intruder which preserves the properties of encryption. In fact, the intruder can read all transmitted messages, re-direct messages, and also block messages. If it receives an encrypted message and has knowledge of the appropriate key, it can learn the contents of the message. The Over-the-air intruder model uses Over-the- air channels which deliver the message to the intruder and to the recipient at the same time. It is possible for the intruder to send messages. However, in this communication model, the intruder cannot perform man-in-the- middle or a protocol on- the-fly attacks. We use the Dolev-Yao Intruder model since it is adequate to evaluate the correctness of our protocols.

2.5.2 Multi-level-μTESLA

We formally verify the state of the art protocol Multi-level-μTESLA, and we present how we model it using HLPSL language in AVISPA model checking tool.

Model Description

We implement a simple network consisting of three nodes: the sender, the receiver, and the intruder. The sender broadcasts data packets and CDM packets. Each node is modeled as an independent process, broadcasting or receiving according to the protocol.

We model the broadcast of messages as sending a copy of every message simultaneously on each channel. We highlight the fact that all messages in AVISPA are passed through the attacker regardless of channel definitions. It is crucial to be able to prove that ML-μTESLA protocol has the desired source authentication property. In order to model the system, we have to model its components using the high level formal language HLPSL.

For ML-μTESLA, the system is composed by the network, the adversary and by the protocol itself. In fact, the analysis starts by specifying the models used for the network, for the adversary and for ML-μTESLA protocol. Having established this formal framework, one can specify the security property source authentication required for ML-μTESLA. After establishing the objectives, the model allows the proving or the disproving of the validity of the source authentication property for ML-μTESLA. We cannot specify the exact topology

of the network. In fact, the model checker will generate all the possible combinations of source node-destination node pairs. Verification will be made over all the possible topologies that can be built with the specified number of nodes.

Given that our state space needs be finite, we set a limit on the number of packets that can be broadcast during one session. One packet is transmitted in each time interval. Obviously, this assumption does not affect the analysis since attacks do not depend on the number of broadcast packets. Moreover, due to the fact that AVISPA does not offer arithmetic semantics, we have modeled the increase of the counter by representing a function $tick$, such as $tick(0)$ represents 1, $tick(tick(0))$ represents 2 and so on.

Moreover, the current model assumes the possibility of gaps, i.e. the receiver may miss a message. We send artificial time ticks to keep the sender synchronized with the receiver. The current model does not include the possibility of messages being delayed, i.e. being received in a later time interval because AVISPA does not handle a clock. How long it takes from sending to receiving a message depends on the value of a delay and the actions the intruder performs.

In our validation model, we assume that ML-μTESLA uses some cryptographic primitives including MAC values ($hash$) and Pseudo-Random Functions (f, $f1$, $f01$). Note that the MAC is computed by a message authentication code function that takes as input a message and a secret key, whereas Pseudo-Random Functions ($PRFs$) provide commitments to keys. S and R know $PRFs$ and the message authentication code function to be used in the session.

We also assume that the participants are initially synchronized: R knows the disclosure schedule of the keys, and S sends packets at regular intervals that are agreed with R during the synchronization process. Our model of ML-μTESLA uses a role environment which includes the two agents S and R, the local variables as hash functions, the channels, the public key, and the function tick which is used in order to calculate the number of keys, time intervals, the intruder knowledge, and also combined sessions. ML-μTesla protocol involves two participants: a sender (S) and a receiver (R). In this scheme, we make the following assumptions:

- S and R communicate through an unreliable channel that is under complete control of an intruder (I).
- While specifying the environment (i.e., defining a set of local states, a set of actions, a protocol), we assume that S has all the information it needs to prepare a packet.
- Moreover, we assume that I has all the information needed to prepare well-formed packets.
- We consider a Dolev-Yao intruder who controls the channel and who is able to encrypt and decrypt messages if he or she has the appropriate key.
- We assume that the intruder sends (resends and fakes) well-formed packets only i.e., any packet contains a message body, a key, and appropriate MAC values.

- We assume that S, R, and I use shared $PRFs$ and a shared MAC function.

Modeling using HLPSL

Below we illustrate $A-B$ notation with the well-known Multi- Level-μTESLA protocol:

Step 1. $S \rightarrow R$: ($M_{i,j}$, MAC($K_{i,j}, M_{i,j}$), $K_{i,j-d}$) where $F1^j(K_{i,d})= K_{i,j-d}$, and $K_{i,j}$=$F1$ ($K_{i,j+1}$)

Step 2. $S \rightarrow R$: ($K_{i+2,0}$, MAC(K_{i+1}, $K_{i+2,0}$),K_i) where F(K_{i+1})=K_i, F01(K_{i+3})= $K_{i+2,N1}$, $K_u = F^{|u-v|}(K_v)$, and $K_{i+2,0} = F1^{N1}(K_{i+2,N1})$, where F, $F1$, and $F01$ are pseudo random functions, $i \in [1...N1]$ and $j \in [1...N]$.

Note that Step 1 corresponds to the transmission of data packets while Step 2 describes the transmission of CDM packets.

We also note that the sender S broadcasts a message ($M_{i,j}$, $MAC(K_{i,j},M_{i,j})$, $K_{i,j-d}$) with a Message Authentication Code (MAC) generated with a secret key $K_{i,j}$, which will be disclosed after a certain period of time. When a receiver receives this message, if it can ensure that the packet was sent before the key was disclosed, the receiver can buffer this packet and authenticate it when it receives the corresponding disclosed key. The sender also transmits CDM (commitment distribution messages) packets ($K_{i+2,0}, MAC(K_{i+1}, K_{i+2,0}), K_i$) in order to authenticate the first key ($K_{i+2,0}$) of each low level key chain using the high level key chain K_{i+1} .

ML-μTESLA distributes the initialization parameters (the number of high level and low level keys, the first keys of the low level key chain and the high level key chain,..) during the initialization of the sensor nodes. We model the time intervals with a series of separate intervals. We release the keys after two messages ($d = 2$) and therefore, after two time intervals, that need authentication as in ML-μTESLA.

$A-B$ notation, which shows only message exchanges is too high level to capture control-flow constructs such as if-then- else branches, looping and other features, which talk about the execution of actions by a single participant of a protocol run. That is why we need a more expressive language like HLPSL. HLPSL is a role-based language, meaning that we specify the actions of each kind of participant in a module, which is called a basic role. We instantiate these roles and we specify how the participants interact with one another by combining multiple basic roles together into a composed role. For each participant in our protocols, there will be one basic role specifying his sequence of actions. This specification can later be instantiated by one or more agents playing the given role.

In the case of our three protocols, there are two basic roles, which we call sender, and receiver. Each basic role describes what information the participant can use initially (parameters), its initial state, and ways in which the state can change (transitions). We have specified the following security property: "the receiver accepts a message if it was actually sent by the sender". We will verify this property by means of AVISPA. We have modeled this property by means of several goal predicates: $witness()$ and $request()$. The first predicate i.e., $witness$ $(S, R, sender\_datastream, M')$ can be interpreted in the following manner. The sender S thinks that he talks with receiver R and for whom he creates the value

M' with the purpose *sender_datastream*. The *request* $(R, S, Sender\_datastream, M')$ specification can be interpreted as follows: the receiver R thinks that he talks with sender S who sent him the value M' with the purpose *sender_datastream*. The protocol goals are declared in the goal section.

Note that in our model, we use state machines to present how we specified some actions and events in our three source authentication protocols. Circles represent states and lines indicate transitions or events. A transition consists of a trigger, or precondition, and an action to be performed when the trigger event occurs. This is illustrated in Figure 2.7. If the system is, for example, in state $S1$ and when some events occur, then it will be a transition to state $S2$ and some actions will be executed. Actions and events are implemented using HLPSL language.

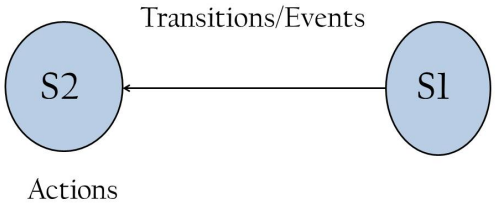


Figure 2.7. – A scheme of state machine interpretation

The initialization phase has to be secured. The sender generates the number of keys in the high level chain and the number of keys in the low level chain. Thus, when the system is in state 0 and we receive an event RCV $(start)$, then the system moves to state 1 and executes the following actions:

- update the current time, i.e. the start time,
- generate the number of keys for each key chain: N and $N1$,
- generate the first key of each key chain,
- transmit the initialization parameters.

We note that this intruder model cannot break cryptography, thus, if the intruder does not know the key, it cannot forge or modify the message. The intruder knowledge is defined in the environment role as follows.

role environment() def=
const s, r: agent,

sr, ir, sync: channel (dy),
hash_: hash_func,
f, f1, f01: hash_func,
K_S: public_key,
tick: text -> text,
t_0, loss: text
intruder_knowledge = {s, r, hash_, loss, f, f1, f01, t_0}
composition
session(s, r, sr, sync, f, f1, f01, K_S)
end role
sr, *ir*, and *sync* are channels that are dominated by a Dolev-Yao intruder model (channel (dy)). When an intruder could execute an action, it needs to add *session(i, r, ir, sync, f, f1, f01, K_S)* in the role environment.

We have used two level key chains: one high level key chain and four low level key chains. Each low level key chain has three keys and each low level key will be generated in the correspondent low level time interval. Knowing the last key of the high level key chain, we can compute the first key of this chain by applying pseudo random functions. Moreover, the multi-level key chain offers the possibility to compute the first key of the first low level key chain using pseudo-random functions

We can assume that the role of the initialization parameters is very crucial to determine the safety of our protocols. In order to transmit a data packet, the sender generates a message M using the operator $new()$, then it sends M, the message authentication code: $(hash\_(K', M'))$ and the disclosed key: $(F1(F1(K')))$.

In what follows, we describe some actions and local states indicating the reception of a new packet (data packet or CDM packet), the buffering of a verified packet, and also, the authentication of a buffered packet. The loss of packet is also modeled and therefore the update of buffering keys. In Figure 2.8, we model the receiver side when a data packet arrives (using HLPSL). In fact, the system is in state 4, and when a data packet is received $(RCV(M1'.Hash'.K\_published'))$, and the current time is inferior to N time intervals $(Time/ = N)$, then the system moves to state 5, and we assign the value of the published key in the data packet to K_{prev2}. K_{prev2} indicates the presumed published key.

In Figure 2.9, we specify the case when a data packet is lost. In fact, when the following conditions are satisfied: (the $Time$ is inferior to N and the node receives $RCV(loss)$), then the receiver updates a variable gap indicating that there is effectively a loss of packet, and the authentication of the buffered packets could be done by the reception of another packet in the future. We have to remember that all the authentication keys are generated from a key chain using a pseudo random function. That is why when a packet is lost, the scheme is still resistant to packet loss, i.e., authentication continues even when there are packet losses.

We note that the intruder knowledge has a major impact on finding out whether there is an attack or not. In our model, we insert a maximum of knowledge: the agents, the hash function, the pseudo random functions, etc.

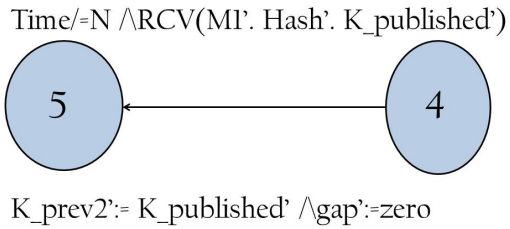


Figure 2.8. – Role receiver: Arrival of a data packet

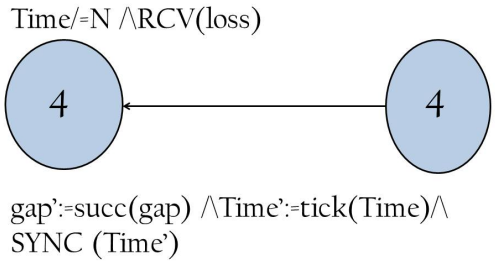


Figure 2.9. – Loss of Packets

The security properties of ML-μTESLA, as is stated in its presentation, are based on the fact that all the packets are transmitted in an authenticated manner. That is why the security property that we formally verified was the authentication of the source by the receiver node when it receives the DATA or the CDM packets that have been sent. In order to verify this property, it had to be formally specified. This authentication security goal consists out of two goal facts (witness and request) and a goal (authentication). The goal facts are used to augment the transitions of the basic roles, and the authentication goal is used to assign a meaning to them. Using OFMC and ATSE backend in order to verify possible attacks, we found that our protocol is safe when verifying the source authentication property.

2.5.3 Staggered Multi-level-μTESLA

In this scheme, the disclosed keys are released earlier than in ML-μTESLA. We used multiple message authentication codes in one packet. This number of MACs is equal to the

key disclosure delay. If for example, the key disclosure delay is three time intervals, then SML-μTESLA packet has three MACs. Our protocol is also initially synchronized then the initialization parameters are transmitted in a secure manner or pre-loaded in the sender and the receiver.

The following scenario describes the A-B Model:

Step 1. $S \rightarrow R :< M_{i,j}, MAC(K_{i,j}, M_{i,j}),$
$MAC(K_{i,j+1}, M_{i,j}), .., MAC(K_{i,j-d+1}, M_{i,j}), K_{i,j-d} >$ where $F1^d(k_{i,d}) = K_{i,j-d}$,
and $K_{i,j}$=$F1(K_{i,j+1})$.

Step 2. $S \rightarrow R :< K_{i+2,0}, MAC(K_{i+1}, K_{i+2,0}), K_i >$
where $F(K_{i+1}) = K_i$,
and $K_{i+2,0} = F1^{N1}(K_{i+2,N1})$,
$F01(K_{i+3}) = K_{i+2,N1}$, and $K_u = F^{|u-v|}(K_v)$
where F, $F1$, $F01$ are pseudo random functions, $i \in [1...N1]$ and $j \in [1...N]$.

As in ML-μTESLA, we define the environment role, the agents, the intruder model and the intruder knowledge. We adopt the same validation model as described in ML-μTESLA.

We indicate that the previous simulation scenarios done in ML-μTESLA could be applied to SML-μTESLA in HLPSL and we have obtained the same results as the previous protocol. In fact, in order to transmit a data packet, as described in Figure 2.10, the sender generates a new message M' using $new()$, then concatenates, with the message $M^{'}$, two message authentication codes (using $hash\_(K^{'}, M^{'})$,$hash\_(K\_prev^{'}, M^{'})$), and the disclosed key $F1(F1(K^{'}))$. In this scenario, we use two message authentication codes because the key disclosure delay is two time intervals. $K\_prev^{'}$ represents the previous key and if we generate a packet for transmission, then K_prev is updated by the value of the authentication key in this time interval. Thus, in the following time interval, this key will be used to generate a message authentication code. The sender uses more than just key to construct MACs during time interval T_i, such as K_prev. Thus many potential adversaries will not be able to forge the MACs constructed using K_prev, K.

Upon accepting a message as valid, the receiver executes the *request* function. This function tests that the identity of the supposed sender and the value itself are the same as the ones specified in the corresponding witness function. If not, then the attacker has managed to successfully forge a packet. The specified model of SML-μTESLA allows the proving of the validity of the source authentication property.

2.5.4 Bloom Filter Multi-level-μTESLA

Bloom filter based scheme (BML-μTESLA) consists of an enhanced version of SML-μTESLA. In fact, this latter uses multiple message authentication codes increasing the communication cost. It is possible to reduce this cost by using a Bloom filter data structure. This, in turn, will reduce the energy consumption.

We specify the Bloom filter based scheme as follows:

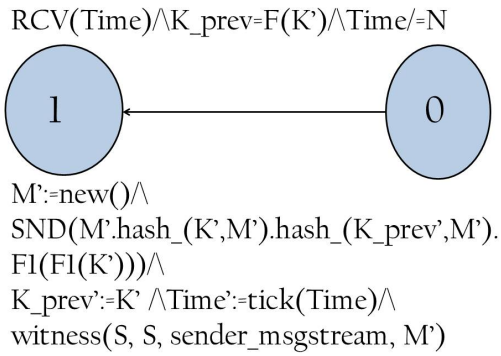


Figure 2.10. – Transmission of a data packet (SML-μTESLA)

Step 1. $S \leftarrow R$: $(M_{i,j}, b_{i,j}, K_{i,j-d})$ where $b_{i,j}$ is a Bloom filter generated in time interval $T_{i,j}$ and maps the following items that was integrated in a data packet in SML-μTESLA: $MAC(K_{i,j}, M_{i,j})$, $MAC(K_{i,j+1}, M_{i,j})$,..., $MAC(K_{i,j-d+1}, M_{i,j})$.

Step 2. $S \leftarrow R$: $(K_{i+2,0}, MAC(K_{i+1}, K_{i+2,0}), K_i)$ where $F(K_{i+1}) = K_i$, $K_{i+2,0} = F1^{N1}(K_{i+2,N1})$, $F01(K_{i+3}) = K_{i+2,N1}$, where F, $F1$, $F01$ are pseudo random functions, $i \in [1...N1]$ and $j \in [1...N]$.

The initialization parameters are transmitted from the sender to the receiver. Then, the sender computes for each packet the corresponding Bloom filter, transmits the message, the Bloom filter, and the disclosed key using the following specification

$$SND(M'.Filter'.F1(F1(K')))$$

where Filter is a set of nat (natural) representing the Bloom filter, and the disclosed key is $F1(F1(K'))$. The receiver must first verify that the disclosed key is safe by comparing it to the latest buffered key. Once this key is verified, then it will be buffered.

We represent the Bloom filter data structure as a set of "0"s and "1"s in HLPSL language. Recall that the model checking tool does not support arithmetic operations. We set the value of the key disclosure delay to two time intervals, as in ML-μTESLA. The receiver first creates two message authentication codes. These two MACs are mapped using three independent hash functions H_1, H_2, and H_3.

2.5.5 Security Analysis

We have specified our protocols using HLPSL language and AVISPA tools. In order to detect attacks like replay, parallel sessions, and DoS attacks, we used ATSE backend which detects possible attacks and gives a trace.

- **Replay Attacks.** Replay attacks consist on playing back previously transmitted messages by the intruder to sabotage a protocol session. Replay attacks take place

when the intruder redirects eavesdropped or altered messages within one or more interleaved protocol session(s). These attacks could be detected using the option session compilation in OFMC backend of AVISPA. OFMC detects the replay attack even without the second parallel session between the sender and the receiver. This is because it first simulates a run of the whole system and in a second run, it lets the intruder takes advantage of the knowledge learn in the first run. The formal analysis that we conducted confirms that our protocols are safe with regards to replay attacks.

- **Type Flaw Attacks.** A type flaw attack happens generally when the receiver of a message accepts that message as valid, but imposes a different interpretation on the bit sequence than the participant who created it. These attacks can be optionally combined with a message interception in order to prevent reception of intercepted message by its recipient such as to perform a type flaw based message replay. The intruder triggers a type flaw attack after having altered an eavesdropped message based on intruder knowledge, thus resulting in another message. Our models of the three source authentication protocols resist to this type of attack since AVISPA (OFMC and ATSE backends) does not indicate any flaw attack trace.

- **Parallel Session Attacks.** This kind of attack takes place by subsequent interleaving replays among protocol sessions, in which the intruder manipulates protocol participants in multiple roles, in order to subvert the protocol's goals. Parallel sessions can be specified in HLPSL, but they may lead to replay attacks and may be used in sequence ones. In AVISPA, we try to detect this attack by applying parallel sessions having the same parameters as the following example: $session(s, r, sr, sync, f, f1, f01, K_S)$ /\ $session(s, r, sr, sync, f, f1, f01, K_S)$.
  Where s and r are agents, sr and sync represent the channels which are dominated by a Dolev-Yao Intruder model. f,$f1$ and $f01$ are hash functions and K_S is a public key. AVISPA did not detect a parallel session attack.

- **DoS Attacks.** It takes place anytime after the occurrence of some action representing a transmission of a message taken by the intruder, or after the transmission of an encrypted message where the intruder knows the encrypted key. Based on $Iknowledge$, the intruder alters messages and transmits them. The intruder performs an integrity violation attack of the received message. In fact, we assume that the intruder concatenates some bogus data. The intruder then performs the transmission of this altered message many times representing many distinct requests. An intruder can transmit many bogus packets to an authentic receiver leading to a DoS attack. We found that ML-μTESLA can encounter DoS attacks on data packets and CDM packets.

We note that the absence of an attack with AVISPA gives a measure of confidence; however it is not a proof of protocol security since AVISPA and alike impose a bound on the role instances to get decidability.

To spoof an individual ML-μTESLA packet to a single receiver, an attacker must successfully forge the authenticator designated for that receiver in the packet and all subsequent

confirmations of validity. The probability of successfully forging a single secure MAC tag of b bits in length is 2^{-b}. In SML-μTESLA the probability of forging the authenticator which consists of d MACs will be $2^{-(d \times b)}$.

2.6 Performance Evaluation

We show that SML-μTESLA satisfies the following requirements: low memory overhead, low energy consumption, low authentication delay. We further compare SML-μTESLA and ML-μTESLA, and show that SML-μTESLA outperforms ML-μTESLA in several ways. In order to evaluate our schemes, we implement the two schemes on TinyOS using TOSSIM simulator. The experiments are done using TelosB motes.

2.6.1 Prototype Implementation

The TelosB motes [52] used in our experiments, run TinyOS operating system version 2.1 and support NesC as a programming language [98]. A TelosB mote is of the size of two AA batteries. It has an $IEEE$ 802.15.4/$ZigBee$ compliant RF transceiver, allowing radio communication in the frequency range 2.4 to 2.4835 GHz (a globally compatible ISM band), and a bit rate of 250 $kbps$ data rate. It is based on the TI MSP430 microcontroller. The MSP430 incorporates 8 MHz, 16 bit RISC CPU, 48 K bytes flash memory (ROM), and 10 K bytes RAM.

TinyOS is written in NesC, a C-based language that provides support for the TinyOS component and concurrency model. Each component can correspond to a hardware element (led, timer, ADC, etc.) and can be reused in different applications. An application is composed by a set of components linked together to achieve a fixed goal. The implementation of a component is done by defining a set of commands, events, and tasks.

Figure 2.11 depicts the components and the interfaces of our schemes. They are composed of a set of components linked together. MainC is a major component that is executed first in any TinyOS application. Basically, our implementation uses several Timer interfaces (provided by a Timer component) for handling message transmission. TimerMilliC is the standard millisecond timer abstraction. In order to enable sending and receiving messages, our scheme uses the AMSenderC, AMReceiverC, MMTESLAReceiverC, and MMTESLASenderC components that provide interfaces Packet, AMPacket, RadioSend, RadioReceive, Primitive and Buffer. This code reuse in TinyOS saves code space in ROM as well as data space in RAM because less variables and cipher contexts have to be defined. AMSender is a virtualized abstraction. That is, each AMSender can handle a single outgoing packet. Every hardware platform defines a component ActiveMessageC, which the basic packet components (AMSenderC, AMReceiverC, etc.) wire to. Generally, ActiveMessageC is just a configuration that renames a particular radio chips active message layer. In addition, our application uses a linear feed shift register random number generator (RandomLFSR) component to generate pseudo-random numbers needed by key chains. An

application can change what the default random number generator is by defining its own RandomC, which maps to a different algorithm. As block cipher, we have chosen the RC5 component to provide both encryption and authentication. RC5 as a symmetric key cipher for this system has already been implemented and tested. The LedsC component actually turns the LEDs on and off, during testing. LedsC is useful for debugging, but in deployment it is a significant energy cost and usually replaced with NoLedsC.

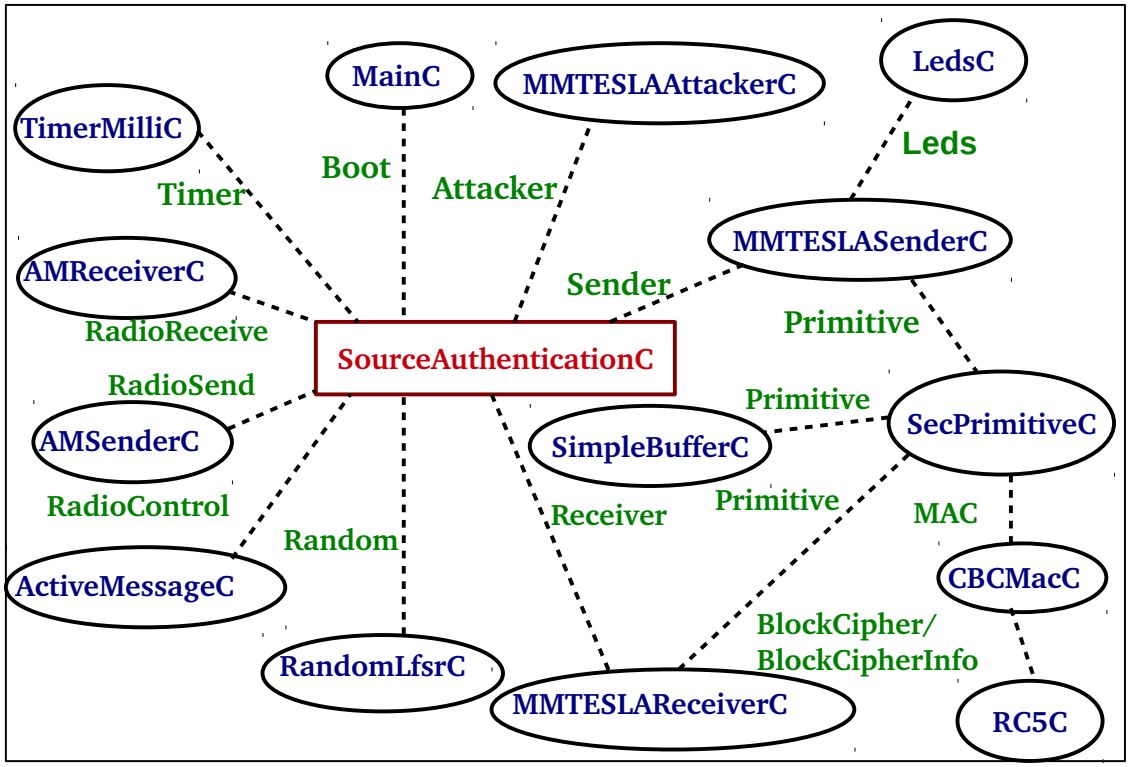


Figure 2.11. – Our Application Architecture

However, one of the major constraints on implementing the two schemes on a sensor platform is the small available payload size of packets. The standard packet payload is limited to 29 bytes under TinyOS.

In ML-μTESLA scheme, it is easy to send all the amount of data using 29 bytes. But, in SML-μTESLA scheme, it is difficult to send all the amount of data using a 29 bytes packet, then we have to modify the application Makefile using "$TOSH\_DATA\_LENGTH = 100''$ in order to have a packet with maximum 100 bytes, since in 802.15.4 the packet does not exceed 128 bytes of data length. We employed timers that fire every 100 ms in our simulations since this provides sufficient spacing for TOSSIM while still allowing for maximum channel utilization.

We notice that TinyOS has memory constraints. In fact, it allows for static memory allocation, in order to cope with the severe hardware constraints of sensor nodes. This makes

it very space and time efficient because there is no need for maintaining an additional data structure to manage the dynamic heap. In other words, this allows using the entire RAM for storing information. However, all variables and their sizes have to be known at compile time which makes working with dynamic data structures as linked lists or hash maps, impossible.

Moreover, some applications may need more memory since the amount of available RAM is not sufficient, probably just temporarily, than the node offers. In this case, EEPROM (also called flash) might be used. EEPROM is mostly larger than the RAM but reading from it and writing to it are operations needing a lot of time and energy. Thus, these memory constraints lead us to optimize the use of global variables that take a lot of memory space. We make pointers to them to reduce them, when it is necessary to use them.

We use TelosB motes in order to test and validate our protocols. However, the TelosB motes need synchronization since the implemented protocols have the requirement of node's synchronization. In order to limit this constraint, we use the flooding time synchronization protocol [51] because it is very efficient compared to other synchronized protocols and it had been tested also in TelosB motes. We had run experiments on 10 TelosB motes, with one malicious mote, and 9 honest motes.

2.6.2 Simulation Setup

We run our simulations with the TOSSIM simulator [98]. TOSSIM is actually a discrete event emulator designed specifically for TinyOS applications. We also evaluate the average node energy consumption overhead needed to authenticate the source of a transmitted packet. To achieve this goal, we used PowerTOSSIM-Z simulator [99]. It is based on TinyOS and TOSSIM. PowerTOSSIM-Z makes use of the TinyOS and TOSSIM component model to instrument hardware state transitions for the purpose of tracking power consumption. Simulated hardware components (radio, sensors, LEDs, etc.) make calls to the PowerState module, which emits power state transition messages for each component. These messages can be combined with a power model to generate detailed power consumption data or visualizations. Our evaluation is focused on the broadcast of data packets. In our experiments, we use a sender, an attacker, and a receiver component. We adopt a setting as follows. The ML-μTESLA key disclosure delay is 3 time intervals, the duration of each ML-μTESLA time interval is 100 ms, and each ML-μTESLA key chain consists of 600 keys. Thus, the duration of each ML-μTESLA instance is 60 seconds. We assume there are 200 ML-μTESLA instances, which cover up to 200 minutes in time. We also assume that each hash value, cryptographic key or MAC value is 8 bytes long. The ML-μTESLA data packet format contains a sender ID (2 bytes), a key chain index (2 bytes), a fragment index (1 byte), and one hash value (8 bytes). As a result, the packet payload size is 29 bytes.

According to the implementation we obtained, each CDM and Data message in multi-level scheme also contains 29 bytes payload. We assume 3 CDM buffers at each receiver for the two schemes. We set initially the data packet rate from the sender as 100 data packets per minute, and allocate 3 buffers for data packets at each sensor node. To investigate the authentication probability under DoS attacks and communication failures, we assume the

attacker sends 100 forged data packets per minute. We also assume the channel loss rate is 0.5. In our simulation scenarios, we define the attack rate as the number of forged packets per min. To improve the accuracy of our results, we repeated the experiments at least 50 times.

2.6.3 Results

In this section, we present the outcome of the two authentication protocols.

Authentication Probability

The authentication probability is the fraction of the authenticated packets received divided by the number of received packets. The Figure 2.12 reports the authentication probability. This is shown for the two protocols. The x-axis of Figure 2.12 indicates the buffer capacity of the receiver that is taken into consideration while the y-axis shows the corresponding authentication probability of the messages. In fact, the authentication probability increases slowly when the buffer capacity is increased. When we assume the same buffer capacity, with varying the attack rate (number of forged packets per min), the authentication probability increases when the attack rate increases.

It is interesting to note that for ML-μTESLA, the authentication probability increases when we decrease the attack rate. Moreover it also increases when increasing the buffer capacity. We can see that the two schemes always have a higher authentication rate when the buffer capacity increases. The reason is that, a sensor node is able to authenticate any buffered message once it receives a later disclosed key, since different key chains are linked together. Though in the ML-μTESLA scheme, lower-level ML-μTESLA key chains are also linked to the higher-level ones, a sensor node may have to wait for a long time to recover an authentication key from the higher-level key chain when the corresponding lower-level key chain commitment is lost due to severe DoS attacks or channel losses. During this time period, most of previous buffered data packets are already dropped.

It is straightforward to prove that ML-μTESLA scheme has to allocate a large buffer to achieve certain authentication probability when there are severe DoS attacks, while the staggered authentication can achieve higher authentication probability without any additional buffer. Figure 2.12 also shows the behaviour of SML-μTESLA corresponds to that of an ideal protocol. The reason is that in this scheme, a sensor node can verify a data packet immediately and when receiving any disclosed key on the later time interval, while in the ML-μTESLA scheme, a sensor node has to wait for a while before authenticating data messages.

Figure 2.13 shows the behavior of ML-μTESLA when we run it on TelosB motes. The x-axis indicates the duration of an attack, while the y-axis shows the corresponding authentication probability of the messages, with varying the attack rate. The authentication probability increases slowly when the duration of the attack is low.

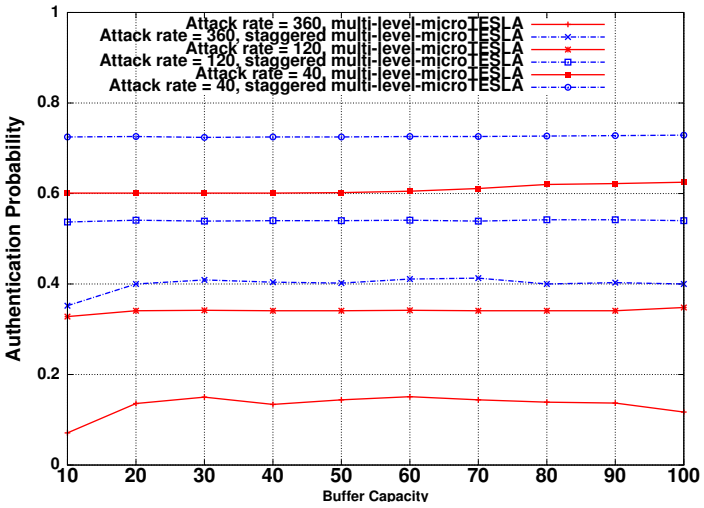


Figure 2.12. – Authentication Probability

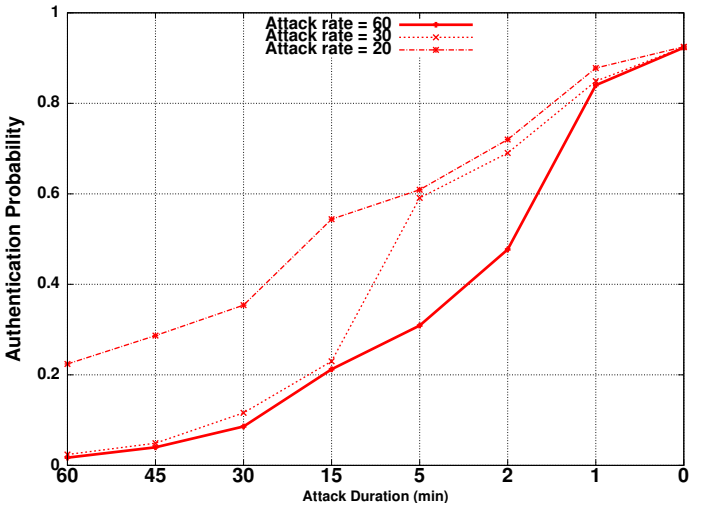


Figure 2.13. – Authentication Probability versus attack rate and duration using TelosB motes

Authentication Delay

To measure the average time for computing the authentication delay, we use SysTime, a TinyOS component that provides a 32-bit system time based on the available hardware clock. The average time to compute authentication delay (the delay of authenticated packet minus the delay of received packet) is throughly the same. This metric varies by varying the loss rate. As it shows in Figure 2.14, the authentication delay increases when the loss rate increases too. In fact, when the loss rate = 20%, then the authentication delay is 500 ms. Moreover, this delay increases for loss rate = 80%, the authentication delay is 1500 ms.

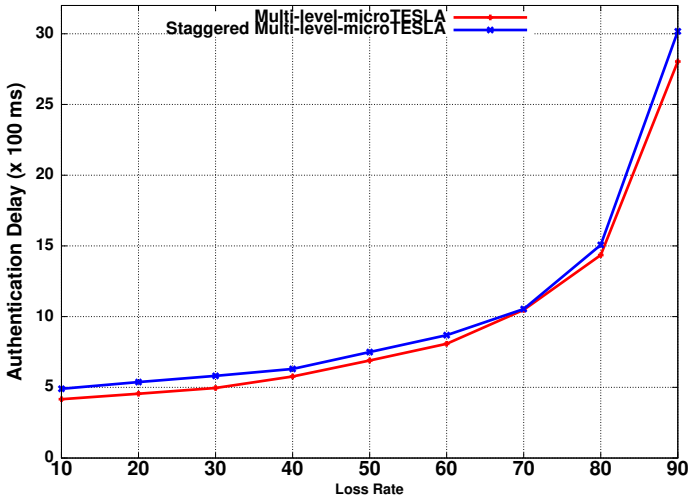


Figure 2.14. – Authentication Delay

Forged Packets

Figure 2.15 reports the average number of forged packets with varying the attack rate using experiments on TelosB motes. We could see that the average number of forged packets increased when the capabilities of the malicious node becomes stronger. When the duration of attack = 40 min, then the average number of forged packets is superior than 1200 (when attack rate = 60), however it is less than the half when attack rate = 20.

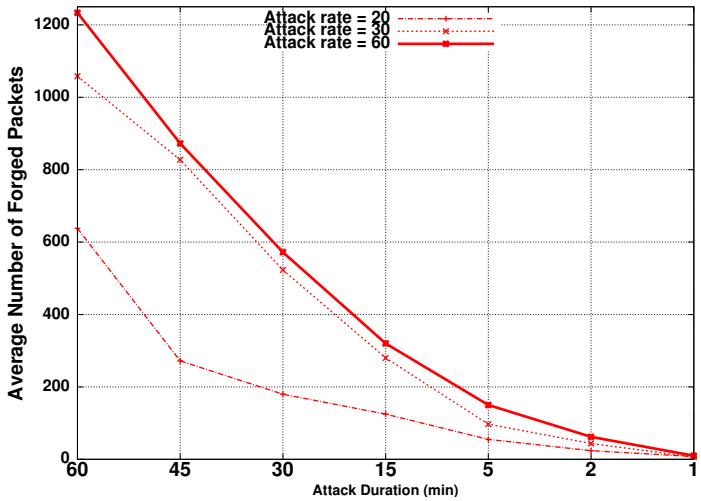


Figure 2.15. – Average Number of Forged Packets in Multi-level-μTESLA using TelosB motes

Figure 2.16 reports the average delay of forged packets with varying the loss rate. We could see that the delay of forged packets is much lower than the authentication delay. This could be explained by the fact that in staggered authentication, forged packets are quickly dropped and deleted.

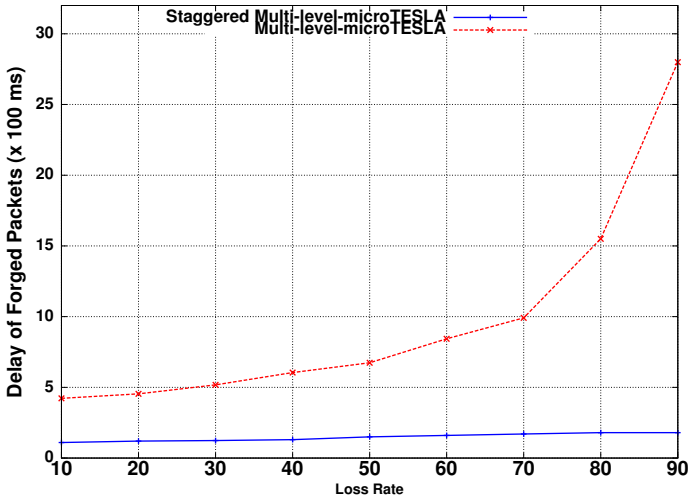


Figure 2.16. – Delay of Forged Packets in the Buffer

In ML-μTESLA, the delay of forged packets is thoroughly the same as the authentication delay. This could be explained by the fact that in staggered authentication, forged packets are quickly dropped and deleted, since received packets are only authenticated after d time intervals (in our experiments, $d = 3$). In Figure 2.17 the delay of forged packets in the receiver's buffer varies with the loss rate. Furthermore, we remark that in staggered authentication scheme, it allows a small delay of forged packets since these packets does not wait for the disclosure of one key. In this scheme, the delay of forged packets is between $(1.5 \times 100\ ms)$ and $(2 \times 100\ ms)$. This also demonstrates that forged packets in the SML-μTESLA are quickly removed from the receiver's buffer.

Figure 2.17 represents the delay of forged and authenticated packets in staggered autentication. Forged packets remain a small time in the receiver's buffer, while the authenticated packets, they are only authenticated after receiving the three authentication keys, since the disclosure authenticated key is three time intervals.

Figure 2.18 represents ML-μTESLA when varying the delay of forged and authenticated packets. In fact, these two delays are throughly the same because forged and authenticated packets have to wait for receiving the three keys. For a loss rate = 10 %, the delay is less than $(5 \times 100\ ms)$ in ML-μTESLA.

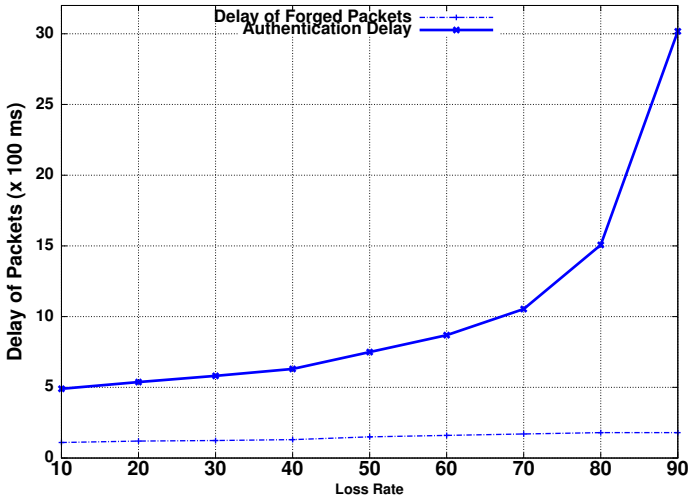


Figure 2.17. – Delay of Forged and Authenticated Packets in Staggered Multi-Level μTESLA

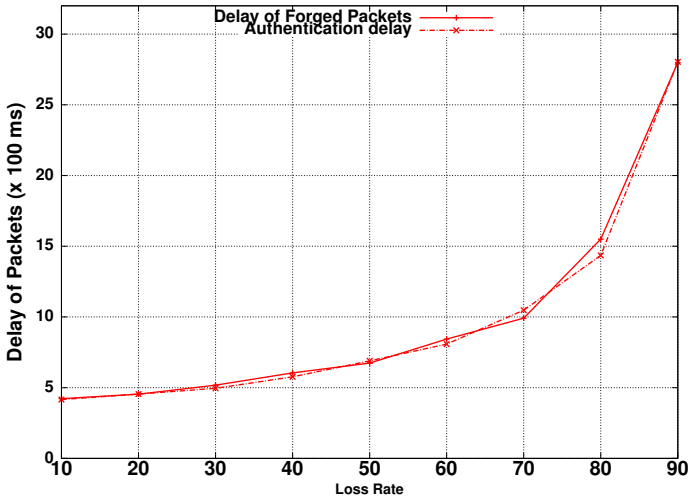


Figure 2.18. – Delay of Forged and Authenticated Packets in Multi-level μTESLA

Memory Requirements

In order to evaluate the memory space requirements among nodes, our implementation of ML-μTESLA in TelosB occupied approximately 2666 (bytes) in RAM and 24786 (bytes) in ROM, representing 25% of the available ROM and 50% of the RAM. SML-μTESLA occupied approximately 3318 (bytes) in RAM and 25962 (bytes) in ROM. The increase of the RAM in the second scheme is due to the staggered mechanism, and the access to the receiver's buffer for each time interval.

Energy Overhead

The commercially available platforms such as the MicaZ are limited to 48 KB of program memory and 10 KB of RAM [133]. They are alimented with double AA batteries that offer energy of 2850 mAh alimented by a 3 v power. Thus a sensor node provides an initial total energy of 30780 joule. The energy overhead introduced by these two applications is 248818 mj for the first scheme and 248927 mj for the second scheme, which is negligible regarding the total energy of a mote ($< 1\%$). These values correspond to a data rate of 100 data packets per minute and an attack rate of 60 data packets per minute. Therefore, the proposed source authentication solution is very efficient and suits well the severe constraints of sensor nodes. Such efficiency is necessary for any security solution in wireless sensor devices. Energy is the scarcest resource. Thus, we underline that every protocol designed for sensor nodes must be evaluated in terms of power consumption to be validated. We demonstrate with our techniques that security systems can become an integral part of practical sensor networks.

2.7 Summary

In this chapter, we present and justify a few of the basic requirements that an ideal protocol for key disclosure delay based source authentication protocol should have. We highlight that a major pitfall of these schemes is the authentication delay between the reception of the packet and its real authentication. In particular, we propose two source authentication schemes. We presented a formal approach to the security analysis of Multi-Level-μTESLA, and our two proposed schemes by means of a model checking tool namely, AVISPA. The output shows that the three protocols are safe in terms of source authentication. We also found that modeling time with AVISPA is not a trivial task since AVISPA does not handle a clock, and we must experiment many session scenarios in order to test the different attacks.

We have also done extensive simulations to show the efficiency of SML-μTESLA. Moreover, we indicate that the overhead of such protocols should be small. Performance comparisons show that SML-μTESLA exhibits better results in terms of authentication probability, energy overhead, and resilience to DoS attack. We are interested in the development of a more accurate intruder model that takes into account the particular features of wireless sensor network.

In this chapter, we focus on memory Denial of Service attacks on broadcast authentication, and in particular in μTESLA based schemes. If we consider the scenario when the receiver needs to log the received messages, in such a way that a malicious node cannot successively modify the messages, μTESLA based schemes are not candidates, to be used in such scenario. Indeed, as soon as the key is disclosed, the attacker may use that key to modify the logged messages. This kind of attack refers to the Delayed Authentication Compromise attack. The next chapter provides a secure mechanism to deal with the delayed authentication compromise attack in key disclosure based schemes in WSNs.

CHAPTER 3

Delayed Authentication Compromise

The previous chapter looked at the broadcast authentication service in key disclosure based schemes. We present that several broadcast authentication solutions have been developed for resource constrained devices, and we focus especially on a security vulnerability that could lead to a denial of service attack. We then propose several solutions to mitigate this problem. In this chapter, we demonstrate that these solutions are not effective when the received messages (and its associated receiving time) have to be stored, and used in a subsequent moment as a proof—a common scenario when receiving devices are unattended for most of the time. We first demonstrate that recently proposed schemes that also address the storage issue are vulnerable to two kinds of attacks: switch command attack (where an adversary pretends to "switch" two messages over time), and drop command attack (where an adversary just pretends to "hide" a message sent from the broadcaster). We then propose a new solution for broadcast authentication: MASS. Our analysis shows that our solution is effective in detecting both switch command and drop command attack, and—at the same time—is more efficient (in terms of both communication and computation) than the state of the art solutions.

The rest of the chapter is organized as follows. In Section 3.1, we give a brief overview of state of the art in broadcast authentication. We present the background in Section 3.2. We discuss the limitations and the weaknesses of two state of the art protocols to cope with two important attacks: switch command attack, and drop command attack in Section 3.3. In Section 3.4 we detail our solution MASS, where we explain the set-up, the transmission and the verification phase of the scheme. We analyze the security of our proposal in Section 3.5. We discuss the overhead of MASS and compare it with state of the art protocols in Section 3.6. Finally, Section 3.7 reports some concluding remarks.

3.1 Related Work

In [19, 134, 135], the authors present different ways to enhance the broadcast authentication service either by using cryptographic puzzles, or multi-level one chains to improve the scalability of μTESLA (to deal with large Wireless Sensor Networks). Let us consider the scenario when the receiver needs to log the received messages, in such a way that an attacker cannot successively modify the messages. Key disclosure based schemes are not candidates, to be used in such scenario. Indeed, as soon as the key is disclosed, the attacker may use that key to modify the logged messages. In [18], this attack process refers to Delayed Authentication Compromise (DAC) attack. In the DAC attack, the adversary may use an already disclosed key to sign for instance the command "open the valve", and save it in the memory of a compromised actuator. By using a key disclosure delay based scheme, the adversary can physically force the door, use an already disclosed key to generate a fake message requiring to open the door, and store it in the memory of the actuator. Once noticed that the valve has been opened, it will be difficult to prove with certainty who has been compromised. If the base station is trusted, we can state that the node is cheating; however if the base station itself can be compromised, then it is not possible to determine who is cheating, the node or the base station.

In order to be resilient against DAC attacks, the authors in [18] propose three algorithms called Partially Accountable Signature Scheme (PASS), Totally Accountable Signature Scheme (TASS), and Prioritized Totally Accountable Signature Scheme (PTASS). PASS works only in particular cases (so we will not focus on it), while TASS and PTASS are more widely applicable. The main idea behind these two protocols is that the time is is divided in time intervals, and we associate the authentication key both to the time interval and to the message content. The base station builds a Direct Acyclic Graph (DAG) composed by a Merkle Tree and several Merkle-Winternitz Signatures [78] (we will give a general overview of these two notions on Section 3.2). When the base station wants to send a command, it sends several nodes of this DAG, in such a way to create a one time signature of the command. The receivers can authenticate the signature, but an attacker cannot use this signature to authenticate other messages. Further details about TASS and PTASS are given in Section 3.2.2. However, it is important to highlight the difference between these two protocols. While TASS does not provide immediate authentication of the messages, PTASS does allow two levels of message priorities: low and high priority messages. The authentication of high priority messages comes with a transmission overhead, but these messages can be immediately authenticated by the receiver.

In the literature, several other protocols have been proposed for the broadcast authentication. In the following, we will review the most important one-time signature schemes that are relevant for our work. One-time signature schemes are signature schemes based on one way functions [136]. Unfortunately, using a single key pair a one-time signature scheme can be used to sign few messages only. This is due to the disclosure of information that shortly after signing a few messages can be used by an attacker. Despite this limitation, one-time signature schemes are advantageous because of their speed. Since they are based

on one way functions, their computation cost is quite low when compared with that of asymmetric primitives. In [137], a one-time signature scheme for broadcast authentication tailored towards peer-to-peer systems has been proposed. It overcomes the restrictions of traditional approaches based on signature schemes like RSA in terms of delays, signature size, and computational complexity. However, it is subject to DAC attacks. Motivated by the application of signatures to stream and broadcast authentication, Perrig proposed a one-time signature scheme called BiBa [138]. The protocol is robust to packet loss, but it suffers from DAC attacks. Hash to Obtain Random Subset (HORS) scheme is an improvement of BiBa that decreases the time needed to sign and verify messages while also reducing the key and signature sizes [139]. It is computationally efficient, requiring a single hash function evaluation to generate the signature and a few hash function evaluations for verification. Due to the large public key size (usually 10 to 20K bytes), HORS does not suite resource constrained devices [140]. Furthermore, in both BiBa and HORS, security decreases as the number of messages that are signed in a time interval increases.

In the following, we present the notation and some notions relative to the Merkle Tree and Merkle-Winternitz Signature.

3.2 Notation and Background

In this section, we present the notation used in this chapter. We also present an overview of TASS and PTASS protocols that are resistant against the delayed authentication compromise attack [18].

3.2.1 Notation

We summarize the notation used in this chapter in Table 3.1.

Symbol	Definition
t	Number of time intervals
c	Maximum number of different commands that the base station can send
m	Maximum number of commands that the base station can send in a time interval
k_{priv_j}	Private key of the time interval j
$s^j_{k,l}$	Command signature used in time interval j, with $1 \leq k \leq m$ and $1 \leq l \leq c$
$z^j_{k,l}$	Checksum node used in time interval j, with $1 \leq k \leq m$ and $1 \leq l \leq c$
F	Hash function

Table 3.1 – Delayed Authentication Compromise: Notation

3.2.2 Background

The protocols that we are going to introduce rely on Merkle Trees (also called Hash Trees) and Merkle-Winternitz Signatures.

Merkle-Tree

A Merkle Tree is a tree-based data structure generally used for authentication purposes. We will use binary trees, however our schemes can be easily adapted to use trees that have any ariety. The value of each inner node of the tree is the result of a hash function that receives in input the concatenation of the values of its children. For a leaf v of the Merkle Tree, the co-path of v is defined as the set of siblings of the vertices that lead on the path from v up to the root. We will use a cryptographic hash function such as SHA-1 for our purposes [141].

Merkle-Winternitz Signatures

Merkle-Winternitz signatures [78] rely on one-way functions to build a Direct Acyclic Graph to encode a signature. This scheme is efficient when it is used to sign low entropy messages. An edge between two nodes $v_1 \rightarrow v_2$ represents an application of the one-way function, in other words: $v_2 = F(v_1)$, where F represents the one way function. If a node has multiple incoming edges, then its value will be equal to the hash value of the concatenation of all its predecessor nodes. Figure 3.1 shows an example of a Merkle-Winternitz signature graph. The signer selects a private key k_{priv} and uses a Pseudo Random Function (PRF) to calculate s_m and z_1. Then, it uses the one-way function to calculate the other values. Finally, the public key k_{pub} is computed by hashing s_1 and z_m; that is: $k_{pub} = F(s_1 \| z_m)$. The public key is distributed to the receivers in a secure way. The left chain is called signature chain, while the simple Merkle Winternitz signature scheme to sign m bits right one is the checksum chain and it is used to prevent forgeries. In our scenario, each node of the signature chain will represent a command, or a message, that the sender may sign. To sign the command i, where $1 \leq i \leq m$, the signer sends the values s_i and z_i as a signature. To verify the signature, the receiver calculates the key $k_{pub} = F(F^i(s_i)F^{m-i}(z_i))$, and compares it with the public key of the signer: if it matches the signature is correct. Since the indices of the checksum chain run in direction opposite to the signature chain, an adversary that wants to forge a signature has to invert at least one one-way function.

The Merkle-Winternitz scheme is a secure one-time signature, but does not scale well to sign a large number of bits. However, it is worth noticing that more than two chains can be used, where each one can encode some number of bits of the signature. For example, one could encode an 8 bit number by using four chains of length 4 to encode two bits in each chain. Furthermore, the number of checksum chain can be further reduced as explained by Merkle [78]. It has to be considered that to sign 128 different commands, instead of two

chains (signature and checksum) with 128 values, we would only need one signature chain with 16 values, one signature chain with 8 values, and one checksum chain with 22 values. Note that this mechanism can be used to reduce the number of hash function evaluations that are required to authenticate a message.

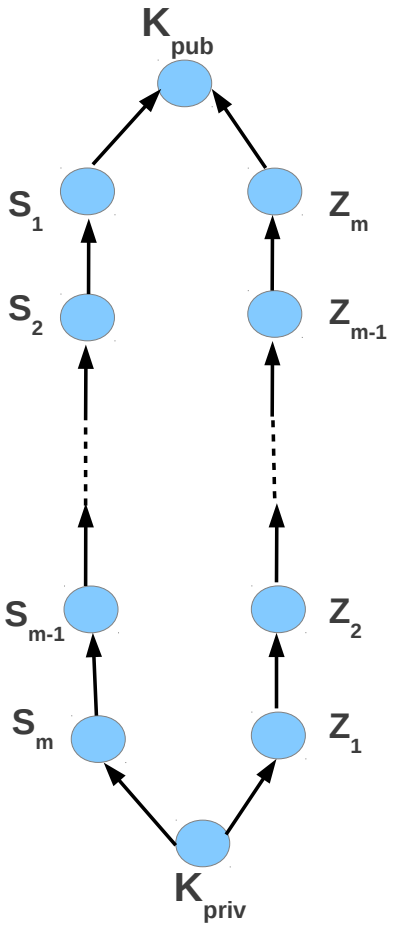


Figure 3.1. – Merkle Winternitz signature scheme to sign m bits

Totally Accountable Signature Scheme (TASS)

In order to be resilient against Delay Authentication Compromise (DAC) attacks, the authors in [18] propose TASS algorithm. The main idea in TASS is to relate the authentication key to the time interval and also to the message content. Thus, the adversary has to figure out what value of key it has to use to forge packets. The base station executes a

certain number of operations in the setup phase of the algorithm, and then builds a Direct Acyclic Graph (DAG), which is a Merkle Tree [78] and a set of hash chains. First, the base station selects randomly t private values $k_{priv\_1}, \ldots, k_{priv\_t}$, one for each time interval. For each private value k_{priv_i} in a time interval, it uses a Pseudo Random Function (PRF) to generate m values $s^i_{1,c}, \ldots, s^i_{m,c}$, and m values $z^i_{1,1}, \ldots, z^i_{m,1}$. Then, the base station starts from the values $s^i_{1,c}, \ldots, s^i_{m,c}$ and $z^i_{1,1}, \ldots, z^i_{m,1}$, to generate $2m$ hash chains. Each hash chain contains $c+1$ items. The last element of each chain represents the leading node. To compute the authentication key k_i in that time interval, the base station uses the hashed value of the $2m$ leading nodes.

$$k_i \leftarrow F(F(s^i_{1,1}) \| \ldots \| F(s^i_{m,1}) \| F(z^i_{1,c}), \ldots, F(z^i_{m,c})).$$

Then, the base station selects randomly t values of checksum nodes z_i, with $1 \leq i \leq t$. The values $k_1, \ldots, k_t$ and $z_1, \ldots, z_t$ are the leaves of the Merkle Tree. After the creation of the Merkle tree, the base station signs the root and sends it to the receivers.

For each time interval, the base station broadcasts the co-path of k_i (see Figure 3.2). Then, it will be able to send at most m authenticated messages during the time interval. When it wants to send a command, the base station sends a tuple composed by three values: the index of the command that it wants to send, one hash value selected among the nodes $s^i_{j,l}$, and one hash value selected among the nodes $z^i_{j,l}$, with $1 \leq j \leq m$ and $1 \leq l \leq c$. In particular, j indicates that it is sending the j^{th} command in that time interval, and l specifies which command the base station wants to send. At the end of the time interval, the base station sends the values of leading nodes of non used chains. The two used chains are $s^i_{j,l}$, and $z^i_{j,l}$. On the receiver side, the sensor receiving the message $< j,\ s^i_{j,l},\ z^i_{j,l} >$, will store it on its buffer until the expiration of the time interval, and the base station transmits the hash values of the first nodes of the not used chains. In that moment, it can authenticate the buffered messages by executing some operations:

- It rebuilds the hash chains up to recover the leading node of each chain, that is $F(s^i_{1,1}), \ldots \| F(s^i_{m,1}), F(z^i_{1,c}), \ldots, F(z^i_{m,c})$.
- By knowing all the values of these nodes, it can compute $k_i = F(F(s^i_{1,1}) \| \ldots \| F(s^i_{m,1}) \| F(z^i_{1,c}), \ldots, F(z^i_{m,c}))$.
- It authenticates k_i by using its co-path up to the root. If the computed root matches the one verified at the beginning of the session, then all the messages are correct and therefore authenticated.

Figure 3.2 illustrates the DAG created by TASS in the Set up phase, for four time intervals. This DAG represents a Merkle tree of eight leaves, and a set of $2m$ hash chains. For each time interval, the authentication key is computed using the values of the leading nodes.

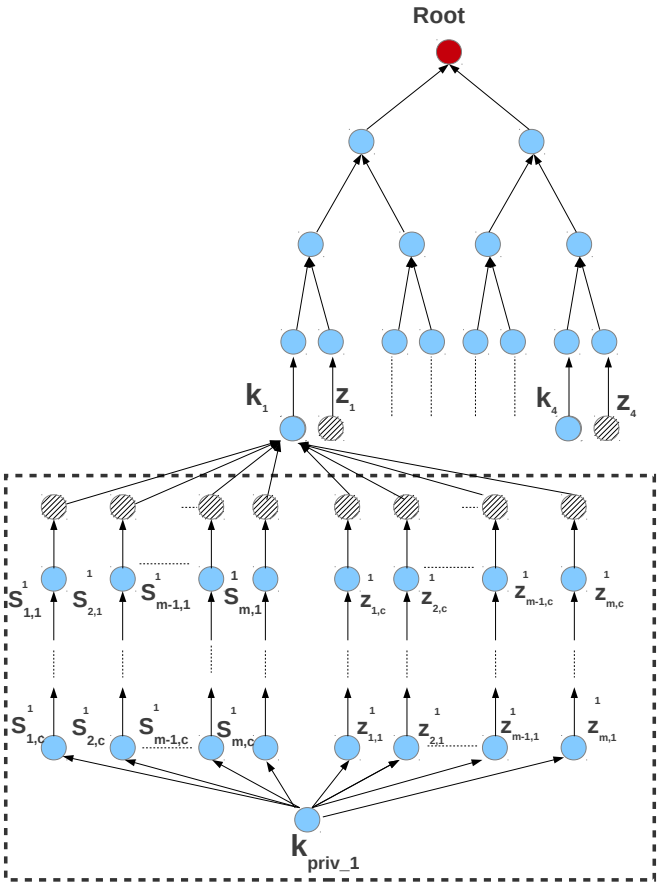


Figure 3.2. – TASS Direct Acyclic Graph ($t = 4$)

PTASS

In TASS scheme, the receiver proceeds to authenticate a message after the reception of the leading nodes of not used chains, at the end of the time interval. In order to resolve this limitation, the authors in [18] propose PTASS. The main goal of PTASS is to authenticate immediately the messages, with using different levels of priorities of messages. In particular, when the base station needs to transmit low priority messages, their authentication can be delayed until the end of the time interval as in TASS. However, when the base station transmits high priority messages, these messages should be immediately authenticated. In order to provide immediate authentication, the base station extends the DAG used by TASS, by adding one more layer. In particular, a new Merkle tree is added between the leading nodes of the hash chains and the authentication keys k_i, for $1 \leq i \leq t$ (t is the number of time intervals). It is straightforward to note that the overhead of creation of

the DAG on PTASS is much higher than the overhead needed to create the DAG of TASS. When transmitting a message, the behavior of the base station depends on the priority of the message. If the message has low priority there is no change with respect to TASS. If it has high priority, the base station will send two nodes of the Merkle Winternitz hash chains (one for the key chain and the other for the checksum chain), together with the hash values belonging to the co-path that are needed to immediately authenticate the key of that time interval. It is interesting to note, that the transmission and authentication cost added by PTASS is not negligible.

In the following section, we give the limitations and attacks for the state of the art.

3.3 Limitations and Attacks for the State of the Art

TASS and PTASS are the only protocols suitable for constrained devices such as sensors and actuators, and that are resilient against the DAC attack [18]. However, these protocols are subject to two attacks that we refer to as the *drop command* and the *switch command* attack.

Drop Command Attack In the drop command attack scenario, the adversary may drop commands that are sent from the base station. Let us assume that the base station wants to update the code of the sensor, and sends a command "reset". In TASS and PTASS, a receiver that does not receive a command, will infer that there is no command transmission by the base station. Thus, in this scenario, the sensor will not execute the command "reset", and it will never notice that a message has been dropped.

Switch Command Attack TASS and PTASS are also vulnerable to the switch command attack that aims at switching the order of the messages sent by the base station. For the sake of clarity, we describe the switch command attack by illustrating an example of commands. Let us consider the following scenario, where the base station needs to send the command "reset" followed by the command "close the valve", while the attacker wants to open the valve. In TASS and PTASS, the malicious node is able to switch the order of the commands, pushing the receiver to execute the command "close the valve", and then to execute the command "reset", that will reopen the valve.

Authentication Delay and Efficiency Limitations In TASS scheme, the receiver has to wait the end of a time interval in order to authenticate the received messages. This may lead also to DOS attacks. PTASS overcomes this drawback by providing a way to send high priority messages that are immediately authenticated. However, this feature comes with an increase of the communication and the transmission overhead. We evaluate and analyze the performances of PTASS in Section 3.6.

In the following section, we propose and detail our secure solution.

3.4 Our proposal: MASS

Our solution MASS improves both the security and the efficiency of the two state of the art schemes TASS and PTASS. In particular, it adds resiliency to the drop command attack and to the switch command attack. Furthermore, MASS provides the immediate authentication as in PTASS, while requiring the same authentication and transmission overhead of TASS. In the following, we present an overview and successively a detailed description of our proposal.

3.4.1 Overview

In MASS, the base station builds a DAG composed by a Merkle tree and several hash chains (see Figure 3.3). The base station has a finite set of c different commands that can broadcast to the receivers. The session is divided in t time intervals, and in each time interval a maximum of m messages can be sent by the base station. At the beginning of the protocol, the root of the DAG is securely transferred to the receivers. Then, when the base station needs to send a command, it also sends a particular set of nodes belonging to the DAG. The receivers use this set of nodes to compute the root of the DAG, and they authenticate the message only if the computed root is equal to the one that has been received at the beginning of the protocol. This is mainly the same approach used in TASS, but the DAG used in our protocol is slightly different. The novelty is in the way the different hash chains are tied one to each other. Indeed, the nodes highlighted in Figure 3.3 as "Verification Nodes" are used to avoid switching of the order of the hash chains. Further details are given in the following about the "Verification Nodes" and leading nodes that are tied in MASS in a novel manner, compared to TASS and PTASS.

3.4.2 Description

The protocol is composed by three main phases. The first phase is the initialization or the set up phase executed by the base station. The second phase consists on the message transmission. The third phase is the verification of the received message.

Set-up Phase

In the set-up phase, the base station builds the DAG depicted in Figure 3.3 by randomly selecting t private keys $k_{priv\_1}, \ldots, k_{priv\_t}$. At the end of this phase, the root of the DAG is securely transferred to the receivers. Note that each node with a solid ingoing arrow represented in Figure 3.3 is computed as the hash of the concatenation of its children.

The creation of the DAG works as follows. For each private value k_{priv_i}, the base station uses a Pseudo Random Function (PRF) to generate m signature values $s^i_{1,c}, \ldots, s^i_{m,c}$, and m control values $z^i_{1,1}, \ldots, z^i_{m,1}$. Each value $s^i_{k,l}$, for $1 \leq k \leq m$ and $1 \leq l \leq c$ is a

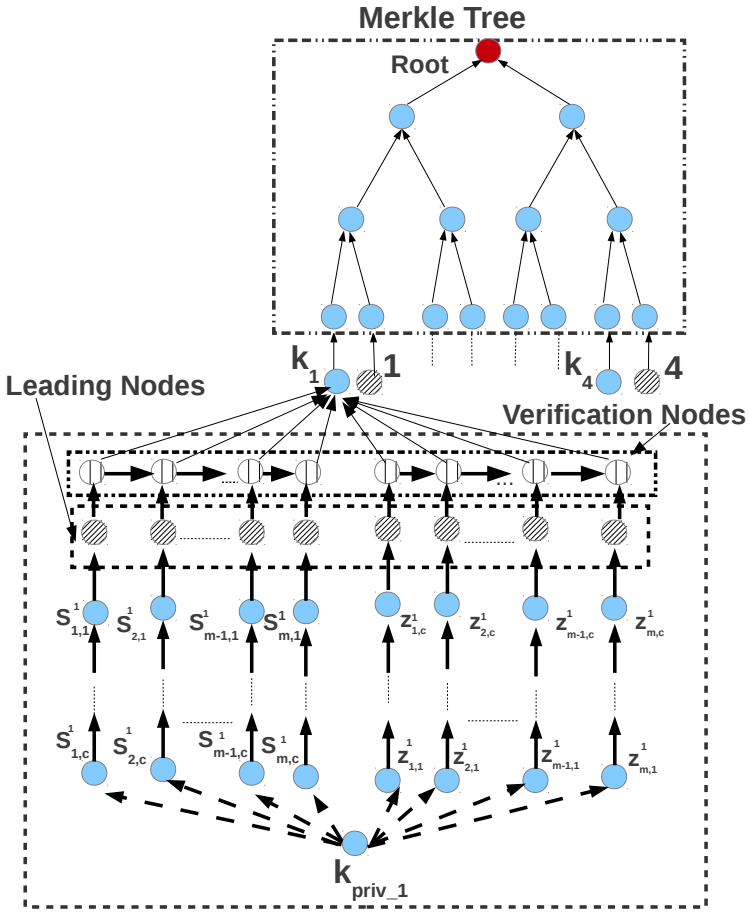


Figure 3.3. – MASS Direct Acyclic Graph ($t = 4$)

command that can be signed by the base station, while $z^i_{k,l}$ are used as checksum nodes. Starting from the values $s^i_{1,c}, \ldots, s^i_{m,c}$ and $z^i_{1,1}, \ldots, z^i_{m,1}$, the base station derives $2m$ hash chains, each one counting $c + 1$ items. We will refer to the last generated item of a hash chain as the *leading node* of the chain. From the leading nodes, the base station generates a new set of nodes called *verification nodes*. It adds a new constraint on how the verification nodes are generated (see Figure 3.3). More formally, the set of the $2m$ leading nodes is as follows:
$< F^c(s^i_{1,c}), F^c(s^i_{2,c}), \ldots, F^c(s^i_{m,c}), F^c(z^i_{1,1}), F^c(z^i_{2,1}), \ldots, F^c(z^i_{m,1}) >$.
The set of $2m$ verification nodes is as follows:
$< F(F^c(s^i_{1,c})), F(F^c(s^i_{2,c}) \| F(F^c(s^i_{1,c}))), ..., F(F^c(s^i_{m,c}) \| F(F^c(s^i_{m-1,c}))),$
$F(F^c(z^i_{1,1})), F(F^c(z^i_{2,1}) \| F(F^c(z^i_{1,1}))), ..., F(F^c(z^i_{m,1}) \| F(F^c(z^i_{m-1,1}))) >$. The authentica-

tion key k_i is then computed by hashing the concatenation of the values of the verification nodes.

By repeating this process for each private value k_{priv_i}, the base station calculates the set of authentication keys k_i, for $1 \leq i \leq t$. It then associates a counter value i to each authentication key k_i. This value is a sequential number that indicates the time interval. Finally, the base station uses the t key values and t counter values to build the Merkle tree highlighted in Figure 3.3. We consider four time intervals, and we show only hash chains regarding K_1 (see Figure 3.3). The resulting DAG (Direct Acyclic Graph) is the combination of a Merkle tree and a set of hash chains.

Transmission Phase

At the beginning of each time interval t_i, the base station sends the values of the leading nodes of that time interval and also the co-path of the authentication key k_i. When receiving these values, the receivers can calculate the verification nodes, and finally the root of the Merkle tree. They have to verify that the root is equal to the one received in the set-up phase. If so, the leading nodes are successively used to authenticate the messages received during that time interval.

When the base station wants to send a command, it sends a tuple composed by three values: the index of the command that it wants to send, one hash value selected among the nodes $s^i_{j,l}$, and one hash value selected among the nodes $z^i_{j,l}$, with $1 \leq j \leq m$ and $1 \leq l \leq c$. In particular, j indicates that it is the j^{th} command sent in that time interval, and l specifies the precise command it wants to send.

Verification Phase

MASS allows immediate authentication. Indeed, the sensor receiving the tuple $< l$, $s^i_{j,l}$, $z^i_{j,l} >$ can authenticate it soon after the reception. The receiver node executes some operations as follows:

- It generates the value of the two leading nodes (of the used chains) corresponding to the received message $< l$, $s^i_{j,l}$, $z^i_{j,l} >$.
- It checks that the two leading nodes correspond to those sent by the base station at the beginning of the time interval. If so, the message is authenticated.

3.5 Security Analysis

The security of MASS relies on the one-wayness of the hash function F. Indeed, it can be shown that an adversary that wants to forge a signature has to invert at least one one-way function. In this section, we discuss the robustness of MASS against two different attacks: the drop command attack and the switch command attack.

Figure 3.4 illustrates the set of hash chains, the leading nodes, and the verification nodes associated to the first time interval. At the beginning of the time interval, the receiver will use the leading nodes to prove if a received message is correct or not. We represent also an example of useful commands ("Close", "Open", "Check", and "Reset") as well as the special command "NULL" added by MASS.

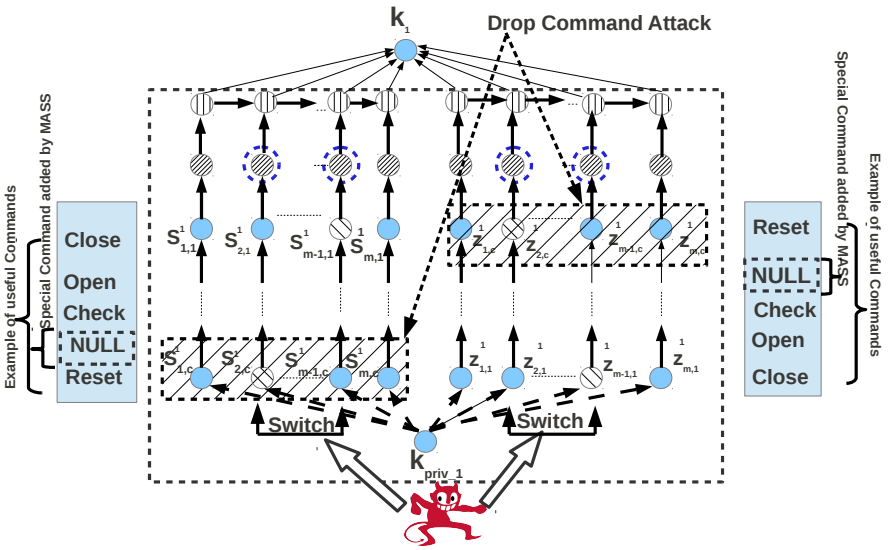


Figure 3.4. – MASS: Drop and Switch Command Attack Detection

Drop Command Attack In order to detect the drop command attack, MASS has a special command "NULL" which is used to explicitly communicate that no active command is sent (see Figure 3.4). Let us consider that the base station in time interval t_i does not need to send any command. In this case, the base station sends a "NULL" command. A node receiving this special command will infer that there is no command transmission at that point in time. If a receiver did not receive any command (i.e., not even the special command "NULL"), then it detects that a command was dropped. In brief, MASS detects the drop command attack thanks to its explicitly use of a "NULL" command.

Switch Command Attack In order to resolve the weaknesses of TASS and PTASS against the switch command attack, MASS uses the leading nodes to authenticate the messages received during a time interval, as described in the transmission and verification phase (see Section 3.4.2).

Let us consider a scenario depicted in Figure 3.4, where the base station needs to send the command "reset" followed by the command "close the valve" to all the receivers. The

aim of the adversary, by running the switch command attack, is to affect the order of execution of these two commands (see Figure 3.4), so that it wants to open the valve. In particular, when receiving the command "close the valve" from the attacker, the receiver executes the following operations.

- It computes the value of the two leading nodes (of the used chains) corresponding to the command "close the valve".
- It checks that the two leading nodes does not correspond to those sent by the base station at the beginning of the time interval.

By executing these operations, the receiver detects that the message is not authenticated.

MASS is also able to detect the switch command attack executed on different sessions. Indeed, the counter value which is a leave of the Merkle tree in MASS, indicates the time interval. A malicious node running a switch command attack on different sessions, could not succeed to achieve its goal.

3.6 Performance Comparison

In this section, we compare the performances of MASS, TASS and PTASS. Table 3.2 presents the different overheads incurred by MASS, and two state of the art protocols (TASS, and PTASS). For the sake of clarity, PTASS high refers to the transmission of high priority messages, while PTASS low refers to low priority messages. To evaluate the efficiency of our approach, we report on the DAG creation overhead, the average transmission overhead, and the average authentication overhead for MASS.

DAG Creation Overhead MASS makes use of DAG, which is made by a Merkle tree [78] and a set of hash chains, while PTASS uses a DAG with two levels of Merkle trees (each leaf of the first Merkle tree is the root of a second Merkle tree) and a set of hash chains. We note that these Merkle trees are precomputed by the base station, and this operation can be executed off-line. In each session composed of t time intervals, the base station computes the root of the DAG and signs it using a public key cryptography, and then sends it to the receivers. It is worth noting that the DAG creation overhead of MASS is related mainly to the computation of the Merkle tree, and the set of hash chains. In fact, for a session with t time intervals, it needs to build a tree with $2t$ leaves. Therefore, the base station needs to use $4t - 1$ hash functions to build the tree. Then, for each time interval, the base station builds $2m$ hash chains, and each chain has $c + 2$ elements. For each time interval, it needs $2m(c+2)$ hash functions to build the hash chains. Moreover, the base station computes the authentication key, by using one hash function evaluation. To generate the two leaves of the tree, it needs two hash functions evaluations. Then, the total number of hash functions evaluations for one time interval is $2m(c + 2) + 3$. If we consider t time intervals to build

Protocol	**Priority**	**Off-line DAG Creation Overhead (Number of Hash Functions)**	**Average Transmission Overhead for an Authenticated Message (Number of Hash Values)**	**Average Authentication Overhead for a Message (Number of Hash Functions)**
TASS	n/a	$t(2m(c+1)+7)-1$	$\frac{\log(2t)}{m}$	$\frac{\log(2t)+3}{m}+c+1$
PTASS	Low	$2mt(c+3)+5t-1$	$\frac{\log(2t)}{m}$	$\frac{\log(2t)+3}{m}+c+1+\log(m)$
	High	$2mt(c+3)+5t-1$	$\frac{\log(2t)}{m}+\log(m)$	$\frac{\log(2t)+3}{m}+c+1+\log(m)$
MASS	**n/a**	$\mathbf{t(2m(c+2)+7)-1}$	$\mathbf{\frac{\log(2t)}{m}+2}$	$\mathbf{\frac{\log(2t)+3}{m}+c+3}$

Table 3.2 – Overheads of TASS, PTASS, and MASS

the tree, then the base station needs $t(2m(c+2)+3)+4t-1$ hash functions. In total, the creation of DAG on our proposal needs $t(2m(c+2)+7)-1$ hash functions evaluations.

Average Transmission Overhead The transmission overhead is an important metric that should be taken into account when dealing with resource constrained devices. This overhead depends on the DAG that is used, and it represents the number of hash values that the base station adds within the message. Indeed, in each time interval, the transmission overhead to authenticate the key of a time interval is equal to $\log(2t)$. Before message transmission, the base station sends the $2m$ leading nodes. In total, the average transmission overhead for MASS is equal to $\frac{\log(2t)}{m}+2$ hash values.

Figure 3.5 reports the average transmission overhead for an authenticated message, while varying the number of messages m and time intervals t. When the number of messages increases, MASS achieves better performances compared to PTASS high. It is worth noting that the number of hash values are smaller than PTASS high. In particular, MASS performs better than PTASS when m is higher than 4 messages per interval.

Average Authentication Overhead To authenticate a message, the receiver has to compute several hash functions. Considering that m messages at most can be sent in each time interval, the receiver in our proposal has to compute the number of hash function evaluations to authenticate a message. Indeed, at the beginning of the time interval, the receiver needs to compute the root of the Merkle tree by performing $\log(2t)$ number of hash functions evaluations. The number of hash functions to compute the verification nodes from the leading nodes is $2m$. The computation of the key based on the verification nodes, requires one hash function evaluation. Moreover, the computation of the two leaves of the Merkle tree needs two hash functions evaluations. Thus, at the beginning of the time interval, the receiver node has to compute $2m+3$ hash functions. Moreover, for each received message, the receiver has to compute the value of the leading nodes of the used chains. This computation requires $c+1$ hash functions. In total, MASS requires $\frac{\log(2t)+3}{m}+c+3$ hash functions evaluations.

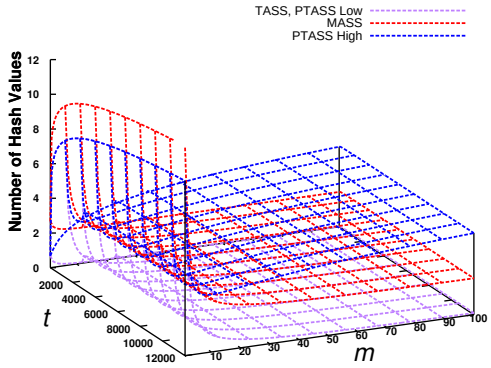
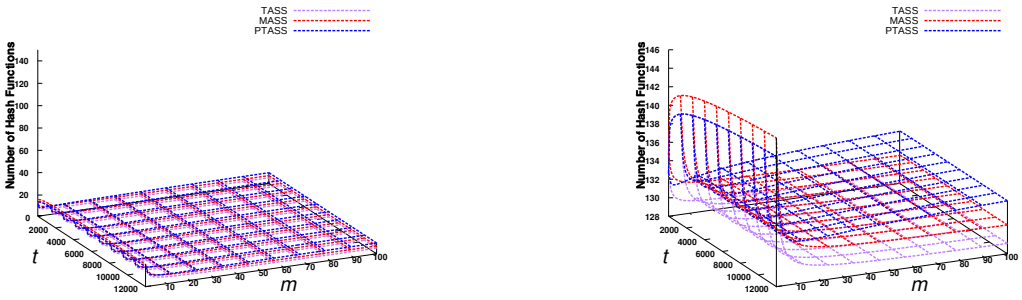


Figure 3.5. – Average Transmission Overhead for an Authenticated Message

In order to assess the average authentication overhead for a message, we varied the number of messages and commands. In Figure 3.6(a), we report this overhead while varying the number of messages m, and fixing the number of commands $c = 5$. It is straightforward to note that the number of hash functions increases slightly when m increases. MASS requires less number of hash functions evaluations compared to PTASS. Moreover, in Figure 3.6(b), we report the average authentication overhead when $c = 128$. It is interesting to note that for PTASS, the average authentication overhead increases. However, MASS has lower overhead compared to PTASS, even when increasing the number of commands and messages.



(a) c=5 (b) c=128

Figure 3.6. – Average Authentication Overhead for a Message

3.7 Conclusion

We presented in this chapter how current broadcast authentication protocols, that are resilient to Delay Compromise Attack, introduce vulnerabilities. An adversary can actually exploit these vulnerabilities to perform the drop command attack, as well as the switch command attack. We come up with a solution MASS, that allows a receiver to detect these two attacks, with an efficient manner. Moreover, the design of our proposal provides immediate message authentication. We analytically compared MASS with the state-of-the-art solutions (TASS, and PTASS) and proved that the overhead introduced by MASS is low, even for a high rate of messages and commands. The analysis of MASS shows its feasibility in the resource constrained devices compared to PTASS. Moreover, the features of MASS make it efficient for real time applications.

As observed in this chapter, one of the most vexing problem for broadcast authentication in WSNs is the delayed authentication compromise attack. In order to provide source authentication in key disclosure based schemes, the concept of key management is temporal asymmetry. Many researches focused on key management and distribution [142, 143] in order to establish authentication keys: pairwise, cluster, and groups keys. It has been proven that group key schemes [25] are more energy efficient than individual schemes due to the low communication overhead. Group key based schemes are used in order to be aware of the overall situation, troops in combat missions report their status and share the observed data. To this aim, in the next chapter, we will focus on group communications, in order to build lightweight source authentication mechanisms in Wireless Sensor Networks. We propose a key management mechanism for the efficient, and rapid adoption of a new authentication key, by a group of one-hop communicating sensor nodes.

CHAPTER 4

Secure Group Communications

In the previous chapters of this thesis, we consider two attacks on key disclosure based schemes: memory denial of service attack, and delayed authentication compromise attack. In this chapter, we present novel symmetric-key-based authentication schemes which exhibit low computation and communication authentication overhead. Our schemes are built upon the integration of a reputation mechanism, a Bloom filter, and a key binary tree for the distribution and updating of the authentication keys. Analytical evaluation of the proposed authentication schemes shows that the estimated average number of concatenated message authentication code in a packet from time 0 till time t is $4pt$, with p is the probability that a key is corrupted.

Our contribution in this chapter is threefold.

1. We propose a key management mechanism for the efficient, and rapid adoption of a new authentication key, by a group of one-hop communicating sensor nodes. The member nodes are arranged according to a binary tree overlay topology, whose root is the communication source (see Figure 4.1). The efficiency of the proposed mechanism is due to the fact that it provides low communication overhead compared to communicating with individual keys. Furthermore, our proposed scheme is rapid since the update of the authentication group key is done by adopting the key binary tree.

2. The group key is vulnerable against many malicious attacks. In order to alleviate against these attacks and reduce their impact, we first focus on proposing a reputation generator in each node. This generator evaluates the reputation value of the nodes. Once a malicious node (which has a low reputation value) is detected, it will be discarded and isolated from the communication process.

3. End to End authentication is ensured via a series of a hop by hop authentication on the path from the source to group members. At each hop, communication between a node and its immediate neighbors is authenticated, with a selected group key from a set arranged according to a binary tree (see Figure 4.1).

This chapter presents a family of source authentication schemes in wireless sensor networks. In the first scheme, we present a key binary Tree based Source Authentication

scheme called TSA, which updates keys based on the reputation of nodes, and it prevents the compromised nodes from understanding the communications between non-compromised nodes. Furthermore, we present a second scheme called BTSA that is based on a Bloom filter data structure to reduce communication overhead. Extensive analysis are conducted to evaluate the proposed schemes, and the results show that these schemes can achieve a good level of security, and significantly improve the effectiveness of source authentication service.

4.1 System Model

We present in the following our architecture system, our design goal and the reputation model.

4.1.1 Architectural System

We consider a wireless sensor network which consists of a large number of sensors (see Figure 4.1). These nodes are limited in power supply, bandwidth, and computational capability.

We refer the reader to figure 4.1 that describes different steps of the communication process using multi-hop communication. At each hop (or at each step of the communication), a communicating source transmits messages to their direct neighbors. In step (1), we consider that node (S_1) is the communicating source, and in this case, (S_1) creates its own key tree (KT_1). S_1 uses the different pairwise keys between $S1$ and its n_1 direct neighbors (S_2, S_3, .., S_{n1}) to create KT_1. At first, S_1 authenticates its messages using the group key $kgr_{S1,1}$. If the reputation mechanism detects that a node or a key is suspicious, then the group key will be updated, it will be $kgr_{S1,1-1}$ and $kgr_{S1,1-2}$. In fact $kgr_{S1,1-1}$ is used to authenticate messages transmitted from S_1 to the set of nodes (S_2, S_3, .., $S_{\frac{n1}{2}}$), and $kgr_{S1,1-2}$ is used to authenticate messages transmitted from S_1 to the set of nodes ($S_{\frac{n1}{2}+1}$, .., .., S_{n1}). Our reputation generator will be further described in section 4.1.3. In step (i), we consider that node (S_i) is the communicating source. S_i creates its own key tree (KT_i) to select its authentication key. KT_i uses the different pairwise keys between S_i and its n_i direct neighbors (S_{i1}, S_{i2},.., S_{in_i}). At first, S_i authenticates its messages using the group key $kgr_{S_i,1}$. If a suspicious node is detected, then the group key will be $kgr_{S_i,1-1}$ and $kgr_{S_i,1-2}$.

4.1.2 Design Goal

We focus on evaluating the energy consumption of our proposed two protocols TSA and BTSA. Moreover, we evaluate the communication, the computation and the memory overhead of the different schemes. We analyze the security of BTSA against bit and random attack.

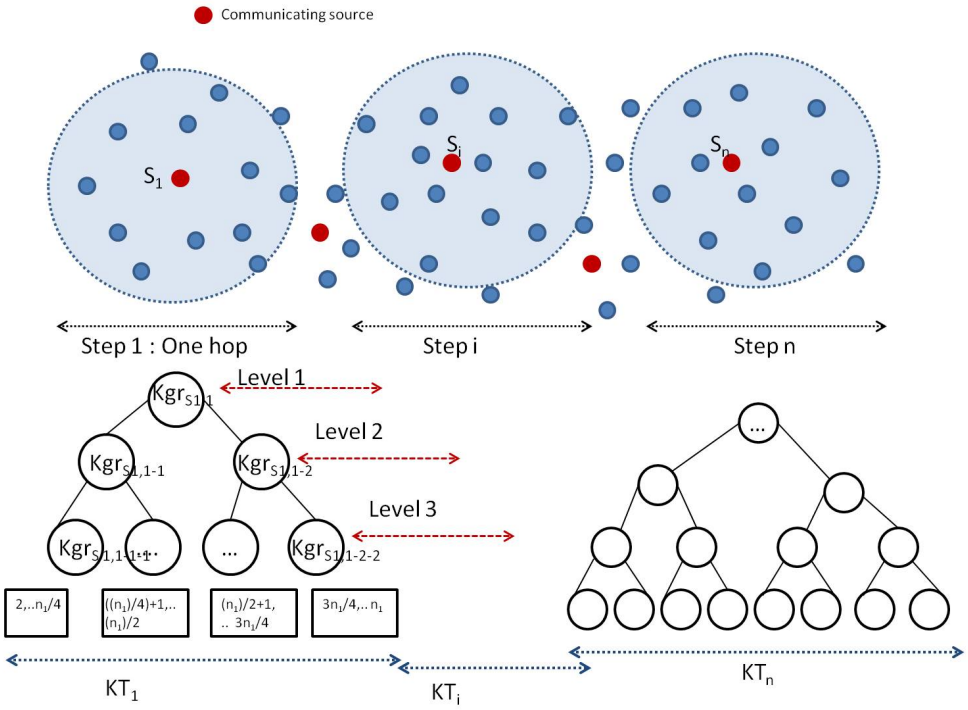


Figure 4.1. – Architectural Model

4.1.3 Reputation Model

The reputation model provides a mechanism to detect malicious nodes, and offers some degree of prevention mechanism. In order to compute the confidence of nodes, the reputation model differentiates nodes as malicious or honest.

State of the art. Detection of Malicious Nodes in WSNs

A misbehavior is related to the detection of false messages, which means that transmitted packets are not authenticated. The way for detecting this kind of misbehavior is using either reputation-based mechanisms or credit-based mechanisms. However, credit based mechanisms are incentive mechanisms which encourage nodes to forward packets, and to not be selfish ([144], [145], [146]). Our concern is to detect misbehaving nodes that authenticate falsely the packet. In fact, the key could be compromised, or the sender/receiver could also be compromised. Thus, reputation based mechanisms seem to be more attractive to our problem. The reputation concept refers to the perception that a node has of another's intention and norms. Reputation is a tool that allows nodes to cooperate and to do a certain service. In other words, a node could be assigned a reputation value determined by its neighbors. Based on the reputation value, a node could be chosen for a service or not (for example it can forward or transmit packets). A node which has a low reputation

value risks to be discarded and isolated from the network. A relevant corpus of research in reputation based mechanisms has been presented in [147], [148]. CONFIDANT [147] uses a reputation based system in order to isolate misbehaving nodes. CONFIDANT and our reputation mechanism have the same objectives, however they differ in some points. Our metrics for computing a node reputation, take into account the node historical reputation which helps to be more fair in calculating a node reputation. Moreover, our metrics take into account the recommendation of all intermediate honest nodes in order to compute the reputation value between two nodes that they don't have a direct communication between them during a time window. We use the recommendations not only from direct neighbors, but from two-hop neighbors, in order to be resilient against collaborative malicious nodes.

A New Reputation-based Mechanism to Detect Malicious Nodes in WSNs

In order to detect suspicious nodes, we used a reputation based mechanism. Each node in the network has a reputation generator. This generator regroups a monitoring module that detects misbehavior, and a reputation module that collects and evaluates the reputation value of a given node. In order to make the proposed scheme resilient against collaborative attacks, we refer to two-hop neighbors in order to use their reputation values. The reputation value between two nodes A and B is represented by $rep(A, B)$, which means node A evaluates the reputation of node B. We assume that $0 <= rep(A, B) <= 1$. In the initialization step, all nodes have the same reputation value equal to 1 (1 is the maximum value). The reputation value concerns the number of authenticated messages divided by the number of received messages.

$$rep(A, B) \quad = \quad \frac{\#authenticated}{\#received}. \tag{4.1}$$

Moreover, each node stores a reputation table having this structure:
$< Node\_id, Reputation\_value, list\_neighbors >$, where $Node\_id$ is the direct neighbor node, $Reputation\_value$ is the reputation value corresponding to the node $Node\_id$, $list\_neighbors$ is the list of neighbors of $Node\_id$. $list\_neighbors$ has the following structure $< Node\_id, Reputation\_value >$.

We suppose that there are two nodes A and B. We consider two scenarios to compute $rep(A, B)$. In the first scenario, we suppose that there is a direct communication between the two nodes. Thus $rep(A, B)$ is computed as follows (Equation 4.2).

$$rep(A, B) \quad = (1 - \alpha) \times rep_{table(A,B)} + \quad \alpha \times rep_{cur(A,B)}. \tag{4.2}$$

where α is a value between 0 and 1. The computed value of $rep(A, B)$ has two parts: the first part designs the previous stored table of $rep(A, B)$ in the stored table of A. The second part denotes the current reputation value of node B. We take into account the historical reputation value of a node. In the second scenario, we suppose that there is no direct communication between the nodes A and B, then node A will check the different

intermediate nodes in its stored table to evaluate the reputation of node B. Assume that nodes C, D, and E have a direct communication between A and B. Only C and D have a reputation value that it is superior than a threshold T. In this case, the reputation value between A and B, is the sum of the different reputation values collected by the nodes C and D. In general, this reputation value is computed using the following Equation 4.3, where $Ninter\_nodes$ is the number of nodes that their reputation value exceeds a certain Threshold T.

$$rep(A,B) \quad = \quad \frac{\sum_{i \in Ninter\_nodes} rep(A,Ni) \times rep(Ni,B)}{Ninter\_nodes}. \tag{4.3}$$

If $rep(A,B)$ is superior than T, we could confirm that B is not suspicious, else B is a malicious node.

4.2 Related Work

Many security-critical applications depend on key management processes to operate. When a node is compromised, these applications demand a high level of fault tolerance. This is a challenging problem since there are many stringent requirements for key management, and the resources available to implement such processes are very constrained in wireless sensor networks. Key management and key distribution are primordial steps to provide source authentication service. The research community has revealed a relevant corpus of work on the key management and source authentication in WSNs [23, 69, 75, 76]. Group key based schemes are energy efficient due to the low communication overhead [25, 149]. However, when a node in the group is compromised, the communication with other nodes will be eavesdropped, and then the group key is compromised.

A primary challenge in providing source authentication service in WSNs, consists on managing the trade-off, between providing acceptable levels of security, and conserving scarce resources [48, 75, 76], in particular energy, needed for network operations. Several solutions have thereafter been proposed to address key management, and key distribution in wireless sensor networks [75], [76]. In particular, relevant research efforts have addressed the group key technique in order to localize computation, and also to decrease communication overhead in wireless sensor networks. Key tree structure is considered in previous literature to secure group communications. Several secure group key management techniques have been proposed to support secure multicast [21, 150]. We use grouping to tackle with the process of combining a set of sensor nodes having similar properties as it was considered in [25], for example in situational awareness, border protection, asset tracking, and digital battlefield. In a typical group key management scheme, there is a trusted third party, known as Key Distribution Center (KDC). This single trusted entity is responsible for generating and distributing keys securely to the group members. In [21], authors propose a group key management scheme called Topological Key Hierarchy (TKH). It generates a key tree by using the sensor network topology with consideration of subtree-based key tree separation

and wireless multicast advantage. In [24, 25], they propose a t degree polynomial with the constant term of the polynomial being the secret key. It computes t distinct shares of the polynomial and stores them at the users. To compute the group key, $(t + 1)$ shares of the polynomial are required and this $(t + 1)$th share is sent as an activating share by the KDC. In [26], the authors conceived a scheme using a key tree to manage group members. Their group key technique reposes on a centralized group rekeying scheme based on logical key tree for WSNs. Their scheme is based on a group key management, that is not practical in sensor networks since it uses complex operations. Furthermore, the authors in [151] proposed a group rekeying scheme for filtering false data in sensor networks.

Several works presented the problem of group key management in secure group communications, the rekeying cost when an element (a member of a group) leaves or joins a group. Nevertheless, none of them have studied the detection of a malicious member, and its impact on the group key mechanism. We present a simple key tree based scheme to transmit authenticated messages, and enforce it with a reputation based mechanism in order to detect malicious nodes. Thus, our mechanism discard/isolate the suspicious nodes (or suspicious keys) from the communication process.

4.3 Proposed Source Authentication Protocols

We assume the existence of a key establishment protocol in wireless sensor network [54, 152]. This protocol can help every pair of sensor nodes to setup a secret key to protect their communication. Moreover, we assume that each local source does a mapping between the physical address for their one hop neighbors, and a logical address assignement. In fact, the communication source assigns logical addresses in an ascending order (see Figure 4.2). Furthermore, our proposed reputation model presented in Section 4.1.3 is used to evaluate and classify the reputation of nodes. This reputation evaluation has a role to warn the different local sources, for a possible compromise of authentication keys. Along this chapter, a node x neighborhood refers to the nodes that are only one hop away from node x. We use the following notation:

- S and R are principals, i.e. the communicating nodes.
- ID_S denotes the sensor identifier of node S.
- M_K is the encryption of message M with key K.
- m is the bloom filter's size.
- $Hash$ is a hash function.
- $MAC(M, K)$ denotes the computation of the message authentication code of message M with key K.
- K_{S-R} denotes the secret pairwise key shared between S and R.

- n_neigh is the number of neighbors.
- e is the number of nodes in the low level group.
- g is the number of low level groups.
- L denotes the height of the binary tree.
- $l(k)$ designs the level of the key K in the binary tree.

4.3.1 Key Tree based Source Authentication Scheme: TSA

In this section, we present our basic key tree scheme called TSA. In fact, a simple and effective way to authenticate broadcast messages is to use group keys. We assume that each source can add multiple Message Authentication Codes (MACs) into a broadcast message. TSA consists of three steps: initialization (key tree generation), message transmission, and group key updating.

Initialization Phase

Each source node S creates a key binary tree (see figure 4.2), based on the number of its neighbors and on the size of each low level group. A low level group refers to a leaf in the key binary tree. Let e be the number of elements in a low level group, and n_{neigh} represents the number of neighbors of the source. There are $g = \frac{n_{neigh}}{|e|}$ low-level groups. There is a trade off between security and communication when choosing the value of g or e. The group keys are organized in a binary tree for height $\log_2(g)$. If we have a tree of height L, S first transmits the first key (the root of the tree) K_{gr}. At each step of the protocol, the source refers to this key binary tree. In fact, in order to authenticate a broadcast message, S should use a group key shared among it and its neighbor nodes. As a result, an adversary will not be able to forge messages without compromising the group key.

Figure 4.2 presents an example of a key tree generated by a source S, which has neighbors presented by 1 to 16. These neighbors are divided into four groups. Each group contains 4 nodes. K_{gr} is the first level key, and it is used to authenticate messages for neighbors 1 to 16. K_{gr1} is a second level key used to authenticate packets transmitted to neighbors 1 to 8. In this initialization phase, the local source generates the first key. In fact, it chooses a random value of q (Algorithm 1, line 2). Furthermore, the source S generates the first level authentication key (K_{gr}) using q random pairwise keys (Algorithm 1, line 3). In the next step, the group key is distributed to group members through unicast (line 4 and line 5). A receiver R_i decrypts the received message using its pairwise key.

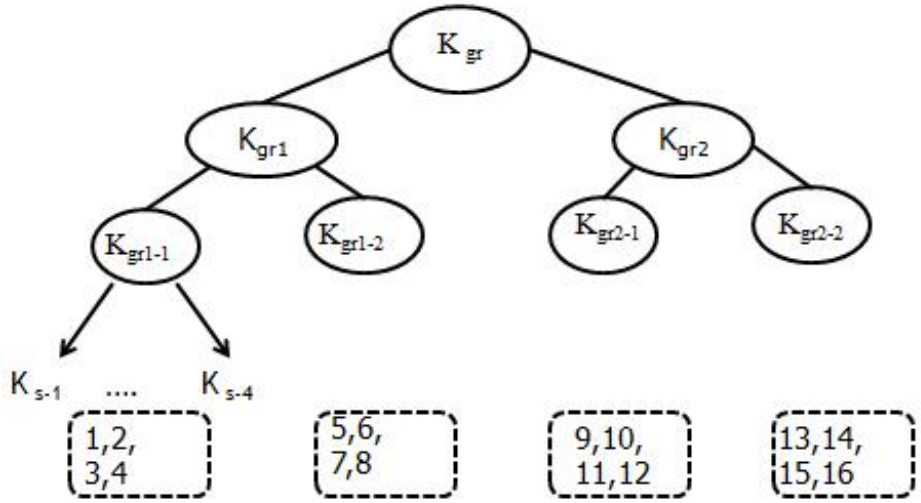


Figure 4.2. – Key Tree Generation by a source S

Algorithm 1: Initialization phase executed by a source S

1 Input: ID_S: Identity of a source S,
$|neigh_S|$: the neighbors of S,
$Hash$: hash function,
R_i: neighbor of source S with $i \in [1...|neigh_S|]$,
K_{S,R_i}: the pairwise key between S and R_i ;
2 S chooses a random value q with $q \in [1...|neigh_S|]$;
3 $K_{gr} \leftarrow Hash(K_{S,R1}||K_{S,R2}||....||K_{S,R_q}||ID_S)$;
4 for all i $\in [1...|neigh_S|]$;
5 transmit $< (K_{gr})_{K_{S-R_i}} >$;
6 EndFor ;

Transmission Phase

In this phase, we present the sending message procedure as well as the receiving message procedure. The process at the source side is shown in Algorithm 2. The source uses the current authentication group keys to generate the corresponding message (Algorithm 2, line 2 and line 3), then sends the message. Each receiver performs the verification steps as follows: First, when receiving the packet, the receiver computes the corresponding message authentication code, compares the computed value to the retrieved one, if both are equal then the packet is declared authentic. Otherwise, it is considered as not authentic. In the sending message procedure, the source S uses its current authentication keys. For example, we could consider the scenario that the authentication group key is the first key which is denoted in figure 4.2 as K_{gr}. Then, the transmitted packet will be $< M, MAC(M, K_{gr}) >$.

The receiver verifies the current message. If the verification succeeds, then the message will be transmitted, else it will be dropped.

Algorithm 2: Sending message procedure executed by a source S

1 Input: M: the transmitted message,
K_{gr1},....K_{gr2},...K_{grn}: n current authentication group keys;
MAC: message authentication code algorithm;
K_{S,R_i}: pairwise key between the source S and its neighbor R_i;
2 For each current authentication group key K_{gri} ;
3 Generate $MAC(M, K_{gri})$;
4 EndFor
transmit < M, $MAC(M, K_{gr1})$,$MAC(M, K_{gr2})$, ..$MAC(M, K_{grn})$ >;

Updating Authentication Keys Phase

Source authentication needs the participation of confident nodes. In our scheme, we assume that after computing the reputation value of its neighbors, the node mentions its confident nodes, and suspicious nodes. Algorithm 3 depicts the updating process executed by a source S. If $rep(S, R_i) < T$ (T is a threshold), then it means that the member R_i has a low reputation value, and that the authentication group key K is suspicious. S proceeds to add it in its suspicious nodes, and updates the group key without using the pairwise key between S and R_i. Moreover, if $rep(R_i, S) < T$, and that n_1 nodes in the neighborhood of S confirm this state, they will send a report to S that they will not receive any message from it, and that the key K could be suspicious. Thus, S will update its group key. If the key K is suspicious, S checks if the level number of the key $l(K) < L$ (Algorithm 3, line 2) then it will use the two keys corresponding to the two child nodes of key K (line 5, line 6) in order to replace key K. S splits the level $l(K)$ group defined by K into two smaller groups to isolate compromised neighbors. We could consider the following scenario when the key K_{gr} (see figure 4.2) is detected as compromised; in this case $L = 3$ and $(l(K_{gr}) = 1) < 3$, S uses the two second level keys K_{gr1} and K_{gr2}. We suppose that key K_{gr1} is computed using q pairwise keys among the set of pairwise keys $\{K_{S,1}, ..., K_{S,8}\}$. Then, S hashes these q keys together with ID_S using the hash function $Hash$. In Section 5, we will evaluate the value of the parameter q. The resulting value is K_{gr} which is the authentication group key for neighbors 1 to 8. The source proceeds to the same operations to compute the second key K_{gr2}, which is computed using q pairwise keys among the set of pairwise keys $\{K_{S,9}, ..., K_{S,16}\}$. Then, S hashes these q keys together with ID_S using the hash function $Hash$. The resulting value K_{gr2} is the authentication group key for neighbors 9 to 16. If one of these keys is detected as compromised, the source will use its two children keys. Then, S transmits the new authentication keys to the presumed receivers and authenticates its messages as follows: $< M, MAC(K_{gr1}, M), MAC(K_{gr2}, M) >$. The receiver will authenticate the received message and if it succeeds, it will be transmitted, else it will be rejected. The process of rekeying continues until we achieve the last level of

the tree. If the last level key group is also suspicious, S uses the pairwise keys of the nodes. For example, in figure 4.2, if we consider that the key K_{gr2-1} is suspicious, in this case the source will update it by using its pairwise keys K_{S-9}, K_{S-10}, K_{S-11}, K_{S-12}.

Algorithm 3: Updating authentication group key executed by a local source S

Input: M_1, *received_msg*: the received data,
$K_{S,R}$: the pairwise key between the source S and the receiver R;
$K_{S,R_{i1}}$, $K_{S,R_{i2}}$, $K_{S,R_{j1}}$, $K_{S,R_{j2}}$: pairwise keys;
K_{gr_i}: the current authentication group key to authenticate messages for receiver R;
MAC: message authentication code algorithm;
Received_report_message ();
if $l(K) < L$ **then**
    K_1, $K_2 \leftarrow$ *use_two_child_nodes_key*(K);
    Choose q random*_pairwise_keys*;
    with (q <number of nodes in each current group) ;
    $K_1 \leftarrow Hash(K_{S,R_{i1}}, , ..., K_{S,R_{j1}})$;
    $K_2 \leftarrow Hash(K_{S,R_{i2}}, , ..., K_{S,R_{j2}})$;
    transmit_encrypted_keys(K_1, K_2);
else
    use_pairwise_keys_nodes_group();

4.3.2 Bloom Filter based Source Authentication Scheme: BTSA

We remark that when the number of compromised nodes increases, the number of used authentication group keys increases too. Thus, the communication overhead rises up. In order to reduce this overhead, we refer to a Bloom filter data structure. In fact, in this section, we present the Bloom filter based scheme which consists on combining the basic key tree scheme and a Bloom filter structure. The major variations of bloom filters include compressed bloom filters, counter bloom filters, and space-code bloom filters [126]. Compressed bloom filters can improve performance in terms of bandwidth saving, when bloom filters are passed on as messages, however it consumes more energy than the standard one. Counter bloom filters deal mainly with the element detection operation of bloom filters. Space-code bloom filters and their variations are suitable for representing static sets, whose size can be estimated before design and deployment.

In our case, we used a space-efficient bloom filter which is a randomized data structure often used to represent a group of elements (in our case n MACs) with a small false positive rate. That is, the Bloom filter saves space at the cost of representing a slightly larger group, than the original group of elements. It uses d hash functions. The message size exchanged in sensor networks should be minimized, in order to be resilient against wireless channel errors and to save energies. The size of a Bloom filter is therefore limited, resulting in high false positive rates. Message authentication codes $\{MAC_1, MAC_2, ...MAC_n\}$ are represented in the Bloom filter b_i. Their values are hashed d times using d different hash functions, and the resulting values are mapped into d bit positions of an m-bit Bloom filter. Note that it is possible for a hashed value to be mapped into an already marked bit position. The

Bloom filter maintains this collision as "1" without change to reduce space. The sender side transmits the following packet: $< M_i, b_i >$ where b_i is the correspondent Bloom filter of the message M_i. On receiving the data packet, the receiver tries to do the following operations:

- verify if the number of "1" bits is less than or equal to $n \times d$ bits in the vector. If it is not, the packet is dropped otherwise, the receiver computes the message authentication code MAC with the correspondent key.
- verify if the computed message authentication code is in the Bloom filter b_i. In fact, for each hash function h_i (with $1 < i < d$) used in the Bloom filter, it verifies if h_i (MAC') is between 0 and $m - 1$. If all the corresponding bits in the vector are set to "1", then the packet is assumed to be authenticated. Else, the verification fails and the packet is dropped.

An element (message authentication code value) which is not a legitimate element of the set could falsely succeed in the Bloom filter verification and therefore a false positive of an element occurs.

4.4 Security Analysis

In this section, we analyze the security of our proposed schemes against some attacks: the bit random attack, and the random attack.

4.4.1 Bit Random Attack Scenario

In this attack scenario, we consider the case that an adversary generates a false Bloom filter of m-bit size (all initialized to "0") in Bloom filter based scheme. Then, the adversary can choose randomly $n \times d$ bit positions in the vector and mark them as "1". This probability is computed as follows

$$\binom{m}{nd}\left(\frac{1}{2}\right)^{nd}\left(\frac{1}{2}\right)^{m-nd} = \binom{m}{nd}\left(\frac{1}{2}\right)^{m}. \tag{4.4}$$

Therefore the probability that a Bloom filter is falsely authenticated after using d hash functions will be

$$fpr_{RbBF} = \left(\binom{m}{nd}\left(\frac{1}{2}\right)^{m}\right)^{d}. \tag{4.5}$$

4.4.2 Random Attack Scenario

This scenario refers to an adversary that randomly generates a false Bloom filter vector of m-bit size. In fact, it randomly marks "1" in $n \times d$ bit positions. As it is known by

the source and its neighbors, a Bloom filter cannot exceed $n \times d$ "1"s. Then, the malicious node marks as many bits as possible up to $n \times d$ bits. Thus, in this case, at a certain bit position, the probability that a bit is marked as "1" is $\frac{n \times d}{m}$. Therefore, the probability that this Bloom filter is falsely authenticated even after verification using d hash functions is fpr_{RBF}.

$$fpr_{RBF} = \left(\frac{nd}{m}\right)^d. \tag{4.6}$$

4.5 Performance Analysis

We evaluate our proposed schemes using metrics: average number of MACs, the communication overhead, and the computation overhead. Both security and performance factors have to be considered when using our proposed source authentication schemes.

4.5.1 Average Number of MACs

In this section, we study the key tree based authentication protocol under the following assumption: each key is corrupted with probability $p > 0$ (and thus not corrupted with probability $q = 1 - p$). We assume also that L, the desired height of the tree is given. For any $t > 0$, we denote by X_t the number of concatenated MACs from time 0 till time t. Then, X_t is a random variable, and it is easy to see that $1 \leq X_t \leq 2^L$. However, one can do a probabilistic analysis of this random variable to obtain its average value. Indeed, we have the following lemma :

Lemma 4.5.1 *If we denote by $\mathbb{E}(X_t)$ the expected value of the random variable X_t, then :*

$$\mathbb{E}(X_t) = 4pt. \tag{4.7}$$

Proof For any $t > 1$, assuming known the value of X_t, one can compute the probability for X_{t+1} to be equal to a given value using the following relation : $\forall i, j > 0$,

$$\mathbb{P}r(X_{t+1} = i \mid X_t = j) = \begin{cases} q^2 & \text{if } i = j \\ 2pq & \text{if } i = j + 2 \\ p^2 & \text{if } j = i + 4 \\ 0 & \text{otherwise.} \end{cases}$$

Hence, we derive the expected value of X_{t+1} given the value of X_t :

$$\begin{aligned} \mathbb{E}(X_{t+1} \mid X_t) &= q^2 X_t + 2pq(X_t + 2) + p^2(X_t + 4) \\ &= X_t + 4pq + 4p^2 \\ &= X_t + 4p. \end{aligned} \tag{4.8}$$

Now, we define the new random variable Y_t as: $Y_t = X_t - 4pt$ for any $t > 0$. Then :

$$\begin{aligned} \mathbb{E}\left(Y_{t+1} \mid X_t\right) &= \mathbb{E}\left(X_{t+1} \mid X_t\right) - 4p\left(t+1\right) \\ &= X_t - 4pt \\ &= Y_t. \end{aligned} \tag{4.9}$$

Thus, the sequence Y_t is martingale with respect to the sequence X_t and hence :

$$\begin{aligned} \mathbb{E}\left(Y_{t+1}\right) &= \mathbb{E}\left(\mathbb{E}\left(Y_{t+1} \mid X_t\right)\right) \\ &= \mathbb{E}\left(Y_t\right) = \mathbb{E}\left(Y_t\right) = 0. \end{aligned} \tag{4.10}$$

Hence $\mathbb{E}\left(X_t\right) = \mathbb{E}\left(Y_t\right) + 4pt = 4pt$. Which ends the proof. ■

Fixing the probability p, it is clearly that $E(X_t)$ is a linear function.

4.5.2 Communication Overhead

In this section, we present the communication overhead related to the two schemes. A chipcon CC1000 radio used in Crossbow MICA2DOT motes consumes 28.6 and 59.2 μJ to receive and transmit one byte, respectively, at an effective data rate of $12.4Kb/s$ [111]. Moreover, we assume a packet size of 36 bytes, 29 bytes for the payload and 7 bytes for the preamble as suggested in TinyOS [153].

If we use the key tree approach without a Bloom filter, the data packet includes data (8 bytes), n MACs with each MAC (8 bytes). Message is then $(8+8\times n)$ bytes. Therefore, there should be $(7\times 1\times x+8+8\times n)$ bytes for transmission (including 7-byte preamble per packet, x is the number of packets to transmit $8+8\times n$ bytes). Hence, the hop-wise energy consumption of message transmission is $(7 \times 1 \times x + 8 + 8 \times n) \times 59.2 \times 10^{-6} mJ$. Then, the energy consumption of message reception is $(7 \times 1 \times x + 8 + 8 \times n) \times 28.6 \times 10^{-6} mJ$. When we used a Bloom filter, message (1) is $(8 + m)$ bytes. Therefore, there should be $(7 \times 1 \times y + 8 + m)$ bytes for transmission (y is the number of packets to transmit $8 + m$ bytes). Hence, the hop-wise energy consumption of message transmission is $(7\times 1\times y+8+ m)\times 59.2\times 10^{-6} mJ$. Then, the energy consumption of message reception is $(7\times 1\times y+8+ m)\times 28.6\times 10^{-6} mJ$. For each broadcast message, every sensor node should retransmit the message once and receive w_0 times the same message where w_0 denotes the node density of sensor nodes within an area equal to the transmission range of sensor nodes. We note W as the network size. Also, we note E_1 the total energy consumption in communication for the Bloom filter based scheme. We have

$$\begin{aligned} E_1 =& W[(7 \times 1 \times y + 8 + m) \times 59.2 \times 10^{-6} + \\ & (7 \times 1 \times y + 8 + m) \times 28.610^{-6} w_0]. \end{aligned}$$

When we don't use Bloom filter, the energy consumption in communication E_2 will be

$$E_2 = W[(7 \times 1 \times x + 8 + 8 \times n) \times 59.210^{-6} + (7 \times 1 \times x + 8 + 8 \times n) \times 28.6 \times 10^{-6} w_0].$$

In the two scenarios, we have varied the number of n and m that should satisfy the equation related to the minimum false positive probability which is defined in [126] by:

$$f = 0,6185^{\frac{m}{n}}.$$

f should be as low as possible, and in our experiments, we have chosen $f = 10^{-6}$. In the first scenario, we have considered a worst case ($n = 16$ MACs attached to the data packet), in the second one $n = 8$ MACs. Figure 4.3 illustrates the energy consumption in communication as a function of W with $w_0 = 10$ of the two schemes in the two different scenarios. Clearly, the Bloom filter based scheme consumes a much lower energy. For example, when $W = 1000$, the Bloom filter based scheme always costs 27 J, while the scheme without Bloom filter costs more than twice that value (59 J) in the first scenario. We observe also that the energy saving increases with network size.

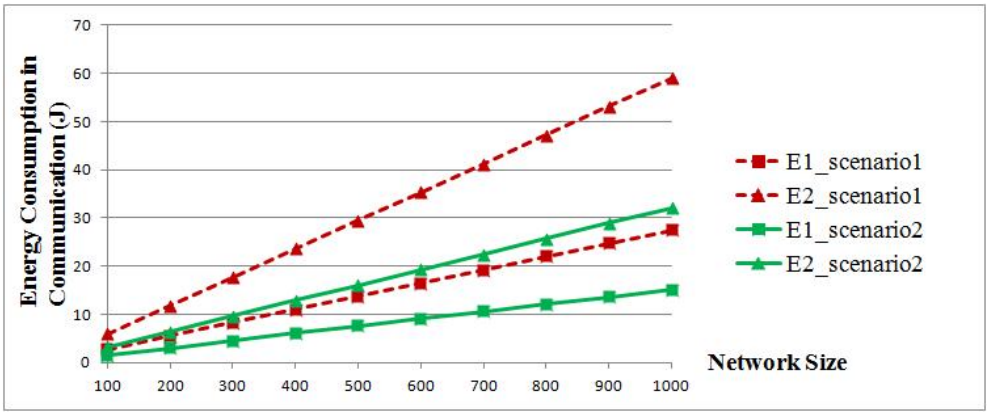


Figure 4.3. – Energy Consumption in Communication

4.5.3 Lifetime of a Key

The authentication group key will be updated once one neighbor of a source S is detected as non-confident. In order to evaluate the lifetime, it needs to evaluate the time between the actual decision and the last update. Let T_i be the random variable that denotes how much time will the key K_i remains for the authentication operation. Let p be the probability

that the key is corrupted, t is a time parameter, and ϵ is a parameter used to control the probability that the key is renewed.

$$Pr\left[T_{ki} <= t\right] \quad = \quad 1 - q^t = 1 - (1-p)^t \tag{4.11}$$

$$\Pr\left[T_i <= t\right] = 1 - q^t = 1 - (1-p)^t \tag{4.12}$$

If we want to compute the probability that T_i do not exceed t

$$\Pr\left[T_i < t\right] <= \epsilon. \tag{4.13}$$

$$\Pr\left[T_i < t\right] = 1 - (1-p)^t \tag{4.14}$$

This implies that

$$(1-\epsilon) <= (1-p)^t.$$

When applying the Logarithm, we have

$$t >= \frac{ln\,(1-\epsilon)}{ln\,(1-p)} \tag{4.15}$$

$$Pr\left[T_i < \frac{ln\,(1-\epsilon)}{ln\,(1-p)}\right] <= \epsilon \tag{4.16}$$

4.6 Summary

Source authentication is in its infancy in Wireless Sensor Networks which are featured with resource constraints and deployed in strategic areas. To address this problem, we have proposed a family of source authentication schemes which rely on a key tree to update the authentication group key in order to prevent the compromised nodes from understanding the communications between non compromised ones. The performances of the schemes have been analyzed mathematically, confirming its effectiveness.

In the previous three chapters, we studied the security and in particular source authentication service in Wireless Sensor Networks. In the next chapter, we will focus on securing alert message applications in vehicular communications. Our aim is to devise secure solutions for fast multi-hop schemes. We analyze the security threats for fast broadcast solutions, and then we come up with a secure solution to mitigate the impact of some attacks on a vehicular safety application.

CHAPTER 5

Fast and Secure Multi-hop Broadcast for Inter-Vehicular Communication

In the previous three chapters, we focus on security issues in Wireless Sensor Networks, and provide efficient solutions for authentication service in these constrained networks. As vehicular communications have attracted a lot of attention in the recent decades due to the need of a safe driving or entertainment, we focus on this chapter on security issues in a vehicular safety application. Inter-Vehicular Communication (IVC) is amongst the most promising and challenging applications of Vehicular Ad-hoc Networks (VANETs) [27, 28]. Many applications are possible in this context, yet local danger warning systems remain the most prominent ones. Clearly, the effectiveness of safety related application is based on the reliability of broadcast information. Therefore, secure communication in this context is a crucial aspect that must not be overlooked.

To discuss attacks to IVC based safety applications, we consider a state of the art protocol which is representative of this class of applications: the Fast Multi-hop Broadcast Algorithm (FMBA) [30]. Through FMBA algorithm, we analyze the security threats to state of the art IVC based safety applications also proposing countermeasures for these threats. We hence propose a solution which is both fast and secure in broadcasting safety related messages: Fast and Secure Multi-hop Broadcast Algorithm (FS-MBA).

This chapter is organized as follows. The next section reviews main work and background related to the security issues of IVC. Section 5.2 discusses the functioning of the case study algorithm (FMBA). Sections 5.3, 5.4, and 5.5 detail possible attacks, whereas respective solutions are presented in Sections 5.6, 5.7, and 5.8. Section 5.9 wraps up the solutions in a single algorithm (FS-MBA). Section 5.10 discusses some performance issues. Finally, in Section 5.11 conclusions are drawn.

5.1 Background on Fast Multi-hop Broadcast Solutions for IVC

Inter-Vehicular Communication (IVC) is an important component of the intelligent transportation system [27, 154–157]. It enables a driver (or her/his vehicle) to communicate in a multi-hop fashion with other drivers located out of the radio range. As a result, information gathered from IVC can foster road safety and transportation efficiency. Benefiting from the large capacities (in terms of both space and power) of vehicles, the nodes of these networks can have long transmission ranges and unlimited lifetimes. Main IVC applications can be categorized into three classes [158], which are as follows:

- Information and warning functions: Dissemination of road information (including accidents, road congestion, etc.) to remote vehicles.
- Communication based longitudinal control: Exploiting the “look through" capability of IVC to help avoiding accidents.
- Co-operative assistance system: Coordinating vehicles at critical points such as blind crossing (a crossing without light control) and highway entries.

Most of the safety related applications, including state of the art ones, share properties that put them into the same class of solutions: IVC based vehicular safety applications [30, 31, 158–161]. These common properties are as follows:

1. Communication is generally vehicle-to-vehicle (V2V), without infrastructure.
2. Vehicles exchange messages containing their position, direction, speed and possible dangers.
3. Broadcast messages have to be propagated as quickly as possible within a certain area of interest, even through multi-hop forwarding.
4. Specific algorithms are employed to choose as few forwarders as possible over the multi-hop path to fasten the propagation of alert messages.
5. Vehicles' information such as position, direction, speed and transmission range is used to feed the forwarder selection algorithm.

For a clearer understanding of the subject, in the remaining part of this section we discuss background information about general medium access control (Section 5.1.1), and general routing in vehicular communication (Section 5.1.2), fast broadcast solutions in safety related IVC applications in Section 5.1.3, and security issues in recent research trends related to vehicular networks (Section 5.1.4).

5.1.1 Medium Access Control in Vehicular Communication

A wide range of project activities have initiated around the world in order to improve vehicular communication networks. IEEE 802.11 task group *p* developed an amendment to

the 802.11 standard in order to enhance the 802.11 to support vehicular ad hoc networks (VANETs). This standard is known as 802.11 *p*, it defines physical and medium access control layers of VANETs. In addition, the IEEE 1609 working group defined IEEE 1609 protocol family which developed higher layer specification based on 802.11 *p*. This protocol consists of four documents: IEEE 1609.1, IEEE 1609.2, IEEE 1609.3, and IEEE 1609.4. IEEE 1609 protocol family and 802.11 *p* together are called Wireless Access for Vehicular Environments (WAVE) standard. This system architecture is used for automotive wireless communications [162]. Due to the characteristics of VANET and limited bandwidth, periodic broadcast messages can consume the entire available bandwidth. Furthermore; the emergency messages need to be disseminated quickly and efficiently. Consequently, there is a need to prioritise important and time-critical messages and use quality of services.

The IEEE 802.11 *p* medium access control layer implements a priority scheme in a similar way to IEEE 802.11 *e* EDCA (Enhanced Distributed Channel Access) function. The IEEE 802.11 *p* medium access control layer is based on multichannel operation of WAVE architecture and 802.11 *e* EDCA. EDCA mechanism defines four different access categories (AC) for each channel. The access categories are indicated by AC0-AC3, and each of them has an independent queue [163]. The EDCA mechanism provides prioritization by assigning different contention parameters to each access category. AC3 has the highest priority to access medium, whereas AC0 has the lowest priority. Each frame is categorized into different access categories, depending on the importance of the message. In IEEE 802.11 *p* MAC layer, there are six service channels and one control channel and each of them has four different access categories.

IEEE 802.11 *p* uses a medium access control protocol that is based on a Carrier Sense Multiple Access protocol with Collision Avoidance (CSMA/CA). Due to the fact that beacons in VANETs may be sent several times per second [164] and vehicular density can vary greatly (and become large during traffic jams), it is expected that the channel may become congested [165], resulting in a deterioration of the cooperative awareness and hence the performance of the Intelligent Transportation Systems (ITS) applications. When a node tries to access the medium and finds the channel busy, it chooses a random backoff time from the interval $[0, CW]$ and delays the medium access for the duration of backoff. CW represents the contention window size. If no acknowledgement is received (e.g. a collision occurs) the CW size is doubled and the process starts over.

Several works evaluate the impact of different CW settings on the performance of 802.11 *p*. CW is initialised to $CWmin$ and doubled after a failed transmission (e.g. the sender has not received an acknowledgement within a certain timeout) up to $CWmax$. As a result, a transmission can be attempted multiple times up to a retry limit, after which the transmitting node gives up. In [163,166], the authors study the performance of 802.11 *p* unicast transmissions for the AC defined by 802.11 *p*'s Enhanced Distributed Channel Access (EDCA). In [166], the authors study two CW adaptation mechanisms in a vehicular to infrastructure setting after identifying that static CW settings do not perform optimal under the dynamically changing vehicular environment.

5.1.2 Routing in Vehicular Networks

To enhance the safety of drivers and provide a comfortable driving environment, messages with different purposes need to be exchanged among vehicles through IVC. However, due to high mobility, efficient routing represents a crucial technical challenge in vehicular communications, thus attracting the attention of researchers all around the world proposing innovative solutions [167–169].

The design of efficient routing protocols for vehicular communications requires particular attention for its highly dynamic topology. In general, topology routing protocols use link's state within the network to transmit the packet from source to destination, whereas this approach would fail in presence of highly variable connectivity among nodes due to their mobility. As vehicular communication can deal with a large number of vehicles but also with interest for local information, geographic routing may embody an efficient approach [167]. Routing based on geographic location exploits nodes' knowledge about their position and their neighbors' position, obtained through services like global positioning system (GPS). In this kind of routing, the vehicle does not maintain any routing table or exchange any link state information with neighbor nodes. In fact, geographic routing protocols take forwarding decisions based on the geographical position of the destination of a packet. A forwarder vehicle must know only its own position, the position of the destination, and the positions of its one-hop neighbors. Geographic routing protocols are not required to maintain explicit routes, thus scaling well even with dynamic networks and representing an interesting approach for vehicular communications.

5.1.3 Fast Broadcast in Safety Applications

In IVC systems, several applications require multi-hop broadcast to inform vehicles (and drivers) about road data, delivery announcements, traffic congestion, proximity with other vehicles, accidents and even entertainment related information for passengers [30, 31, 158–161, 170]. The simplest broadcasting mechanism is flooding, where messages are re-broadcast by each node that receives them. Although very simple, yet this technique may lead to high message collision probability and data redundancy, thus resulting rather inefficient in terms of radio resource usage and message delivery time.

When a message is disseminated to receivers beyond the transmission range, the multi-hopping could be used. However, multi-hop broadcast can consume a significant amount of wireless resources for unnecessary retransmissions. These facts are important motivations for many works focused on efficient multi-hop message broadcast in VANETs; as high mobility and high number of nodes make multi-hop broadcast significantly more challenging in a VANET environment.

The broadcast delivery time represents one of the main issues of IVC. Indeed, it has been proven that this characteristic is strictly related to both the number of relays of the messages (hops) and the network congestion [30, 159, 161, 171–176]. In [171], the demand-driven transmission (DDT) protocol adjusts the timing of rebroadcast packets such that

the vehicle farthest away from the source node retransmits earlier than the other nodes. Ad hoc multi-hop broadcast and urban multi-hop broadcast are proposed in [161] for vehicular networks. These protocols are designed to address the broadcast storm, hidden node, and reliability problems in multi-hop broadcast. Sender nodes try to select the farthest node in the broadcast direction, to assign the function of forwarding and acknowledging the packet without any a prior topology information. That is, senders select the farthest node without knowing the identification (ID) or position of their neighbors. FMBA [30] aims at reducing the number of hops traversed by a message, in order to minimize the propagation delay of a message. Vehicles in a car platoon dynamically estimate their transmission range and exploit this information to efficiently propagate a broadcast message with as few transmissions as possible. In essence, the farthest vehicle in the transmission range of a message sender or forwarder will be statistically privileged in becoming the next (and only) forwarder. In [159], authors have enhanced the fast broadcast algorithm using heterogeneous transmission range. Unlike [30], the authors select the forwarder of the message as the vehicle which transmission spans farther, not the farthest vehicle in the transmission range of the sender.

In summary, several multi-hop broadcast algorithms have been proposed. These algorithms generally share a set of properties, thus falling into a single class of solutions. Unfortunately, they have all been developed without security in mind, whereas security is a fundamental problem in this context which should not be overlooked [29]. Indeed, attackers might run malicious actions to inject false information or alarm, thus rendering ineffective the safety application [35–41].

More in detail, the authors in [39] classify the attacks on vehicular communications into:

- **Bogus information.** One or several legitimate members of the network send out false information to misguide other vehicles about traffic conditions. In order to cope with such misbehavior, the received data from a given source should be verified by correlating and comparing them with those received from other sources.
- **Cheating on positioning information.** Injection of a false position by a malicious vehicle pretending to be at a claimed position.
- **ID disclosure of other vehicles.** This is to track their location. A global entity can monitor trajectories of targeted vehicles and use this data for many purposes, we could take the example of some car rental companies that track their own cars.
- **Denial of Service.** The attacker may want to bring down the IVC or even cause an accident. Example of attacks include channel jamming and aggressive injection of dummy messages.
- **Masquerade.** The attacker claims to be another vehicle by using false identities.

In this work, we analyze the security of a representative algorithm for state of the art IVC based safety applications, and propose countermeasures to handle the security threats. In particular, we focus on one of the main threats to safety application: the possibility to attack the protocol to impede its useful service. For ease of exposition, but without loss of

generality, we focus especially on FMBA as it embodies both a state of the art solution and a representative example of the IVC based vehicular safety applications class possessing all the five properties mentioned in Section I. Indeed, problems and possible countermeasures identified for FMBA can be adapted also to other protocols/algorithms, belonging to the same general class of applications sharing the aforementioned set of properties.

5.1.4 Fast and Secure IVC in Future Trends of Vehicular Networks

For completeness, we present in this subsection a brief discussion on fast and secure message transmission considering research trends in vehicular networks.

Vehicles are an important source of computing and sensing resources for drivers. These resources are increasingly under-utilized. The idea of vehicular clouds comes in handy to solve this problem [177, 178]. In fact, the aim of this technology is to let vehicles share resources such as computational power, storage and Internet connectivity. Security issues encountered in vehicular clouds are very specific [178]; the high mobility and position information of vehicles make the problem very novel and challenging. In addition, the attackers are physically moving from place to place as vehicles are mobile nodes. Compared to a static network, it is much harder to locate the attackers. Moreover, in vehicular cloud, attackers and their targets may be physically colocated on one machine. For instance, an attacker can obtain confidential information and tampering with the integrity of information and the availability of resources.

Named Data Networking (NDN) is a newly proposed architecture for the future Internet that replaces IP end-to-end communication model with a request/reply model to retrieve data by application data names directly; distributed caching among participating nodes is exploited to this aim. This technology has been recently proposed also for vehicular networks [179]. Even if few works have recently shed light on significant privacy issues for NDN [180], and shown the feasibility of DoS attacks that may render caching and routing inoperative [181, 182], a comprehensive analysis of security issues and related solutions for vehicular networks is still lacking. Furthermore, we also highlight the possibility of leveraging FM radio channels for (malicious) IVC communication, as recently shown in [183], for Android car radio devices.

Finally, security offered in infrastructure based multicast or broadcast, such as in WiMax and other similar wireless technologies (e.g., [184]), could be considered for employment even in vehicular networks. Furthermore, several solutions are focused on broadcast based on static network with pre-established shared secrets between nodes and the base station, thus introducing delays. Unfortunately, the peculiar size, mobility, and connectivity of vehicular networks make the aforementioned solutions not suitable for this context. Furthermore, road safety related applications are strictly based on geographic locality and on real time response, thus achieving higher efficiency levels when based on Vehicle-to-Vehicle communication, rather than resorting on centralized approaches.

5.2 Preliminaries and Notation

In this section we present the notation (summarized in Table 5.1) and assumptions used in this chapter. Furthermore, we describe in details FMBA, the case study chosen to represent IVC based vehicular safety applications.

Symbol	Definition
$CMBR$	Current Maximum Back Range
$CMFR$	Current Maximum Front Range
$LMBR$	Latest-Turn Maximum Back Range
$LMFR$	Latest-Turn Maximum Front Range
$MaxRange$	How far the transmission is expected to go backward before the signal becomes too weak to be intelligible
d	Distance between two vehicles
CW	Contention Window
$CWMax$	Maximum Contention Window
$CWMin$	Minimum Contention Window
$Hello$	$Hello$ message transmitted by a vehicle in the estimation phase to update the transmission range
drm	Declared transmission range in the $Hello$ message
P	The prover vehicle
V	The verifier vehicle
R	The geographical region

Table 5.1 – Fast and Secure Multi-hop Broadcast for IVC: Notation

It is worth noting that FMBA is designed to speed up multi-hop broadcast both frontward and backward. However, for the sake of clarity, in this work we refer only to the case where alert messages have to be sent only backward, with respect to the vehicular traveling direction (the frontward case is just specular).

5.2.1 Model assumptions

To simplify the discussion we have made the following assumptions about the general model we are considering:

- We suppose that at most one malicious vehicle is on the network since, we want to evaluate the impact of one malicious vehicle on degrading the performances of the network.
- There are no obstacles and no buildings in the road.

- The hearing communication range is symmetric. It means that if a vehicle V hears a vehicle P, then we assume that P can also hear V.
- We suppose that there are N vehicles arranged in the platoon. A platoon can be looked at as a collection of nodes/vehicles connected by a wireless local area network (LAN), and are engaged in following each other longitudinally.
- A vehicle V does not know its transmission range.
- The verifier node V communicates directly with the verified node P.
- Each vehicle knows its own location, for instance, using GPS that provides accurate information about time and position.
- All the vehicles belong to a Public Key Infrastructure [185, 186]; i.e., each vehicle has a public/private pair of keys and a unique identity certified by a Certification Authority. We assume that the certification authority corresponds to the government agency responsible to assign license plates: a vehicle can be used only if it is provided with a unique license plate, a PKI certificate associated to its plate ID, and the public key of the Certification Authority. We assume that certificate revocation lists are updated at given time interval (e.g., daily) by the vehicle and stored in a local memory[1].
- The power and computational resources are supposed largely adequate for our application's requirements.
- The network is loosely time synchronized.

5.2.2 Fast Multi-Hop Broadcast Algorithm (FMBA)

The aim of the Fast Multi-Hop Broadcast Algorithm (FMBA) is to reduce the time required by a message to propagate from the source to the farthest vehicle in a certain area of interest [30]. To achieve this goal, FMBA exploits a distributed mechanism for the estimation of the communication range of vehicles. These communication range estimations are obtained by exchanging a number of $Hello$ messages among the vehicles, and are then used to reduce the number of hops an alert message has to traverse to cover a certain area of interest. This leads to a decrease in the number of transmissions as well as the time required by a broadcast message to reach all the cars following the sender within a certain distance.

This scheme is composed by two phases: the estimation phase, and the broadcast phase. The former is continuously active and is meant to provide each vehicle with an up-to-date estimation of its transmission range. Instead, the latter one is performed only when a message has to be broadcast to all vehicles in the sender's area of interest.

[1]Please note that the list of revoked ID for *all* passenger vehicles in USA could fit in some 1GB storage.

During the estimation phase, each vehicle estimates its transmission range (frontward and backward) by the means of $Hello$ messages. The aim of this protocol is to have all the vehicles aware of their transmission range, and thus it could exploit this in the broadcast phase. In order to continuously refresh the estimation of the transmission range of the different vehicles, time is divided into turns and the information collected during a certain turn are kept also for the whole next one, and then discarded. Information for the current turn is represented by Current-turn Maximum Front Range ($CMFR$) and Current-turn Maximum Back Range ($CMBR$). In fact, $CMFR$ represents the estimation of the maximum frontward distance from which another vehicle along the area of the interest can be heard by the considered vehicle. The value of $CMBR$ is used to estimate the maximum backward distance at which the considered vehicle can be heard. The values of $CMBR$ and $CMFR$ are updated based on the received $Hello$ messages until the current time expires.

The priorities of vehicles to forward the broadcast message are determined by assigning different waiting times from their reception of the message to the time at which they try to forward it. This waiting time is randomly computed based on a contention window value. If, while waiting, some of the following cars already transmitted the message, preceding ones abort their forwarding process as the message has already been propagated. In fact, the larger the contention window, the more likely somebody else will be faster in forwarding the broadcast message.

In particular, different contention windows are here assigned to each vehicle. The contention window of each vehicle in the platoon is varied between a minimum value (CWMin) and a maximum one (CWMax). This value depends on the distance from the sending/forwarding vehicle and on the estimated transmission range ($MaxRange$) declared in the broadcast message. In other words, the contention window represents the maximum number of time slots a vehicle waits, before taking upon itself the task of forwarding the broadcast message: each vehicle randomly selects a waiting time within its contention windows. In order to forward a packet, each receiver has to compute its waiting time before attempting to forward the message. This waiting time is expressed through a contention window (CW) computed using Equation 5.1.

$$CW = \left| \frac{(MaxRange - d)}{MaxRange} \times (CWMax - CWMin) + CWMin \right|. \tag{5.1}$$

When a car has to send or forward a broadcast message it computes the $MaxRange$ value in the broadcast message as the maximum between $LMBR$ and $CMBR$ values. To avoid unnecessary transmissions, all vehicles between the original sender and the current forwarder abort their attempt to forward the message; whereas all vehicles behind the current forwarder compute a new CW based on last forward parameters to participate in the election for the forwarder on the next hop.

The use of Equation 5.1 is to determine which vehicle will propagate the broadcast message on the next hop. For instance, upon receiving a broadcast message from the front, the considered vehicle will determine its contention window based on Equation 5.1, and then

computes a random waiting time based on it. If, while waiting, the same message has been heard again coming from behind, the message has already propagated from the considered vehicle that can hence stop trying to forward it: somebody else already did it. Conversely, if the same broadcast message is heard from frontward, this means that a preceding vehicle has already forwarded it. The procedure has to be restarted with the new parameters included in the newly heard broadcast message. If the waiting time expires without having heard any other vehicle forwarding the same message, then considered vehicle broadcasts it including the estimated transmission range. If the broadcast fail, a backoff mechanism is utilized to compute the next transmission time.

In Figure 5.1, we present the CW of a vehicle V versus the position of different vehicles. Vehicles compute their CW through Equation 5.1. The farther a vehicle is from the source of a broadcast message, the smaller its CW results.

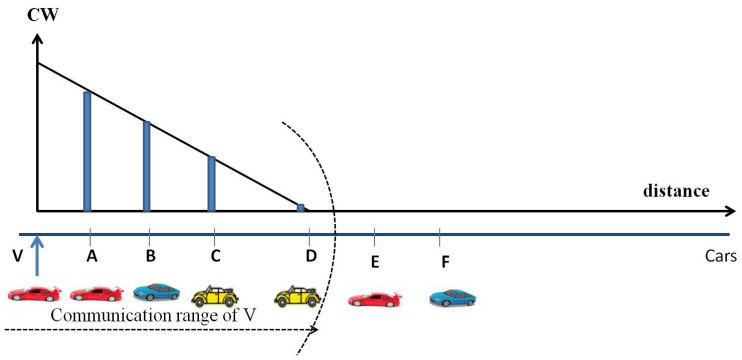


Figure 5.1. – Contention window versus distance

The waiting time is a value computed randomly within CW. Thus, as presented in Figure 5.1, if we assume distances among vehicles as $d(D,V) \geq d(C,V) \geq d(B,V) \geq d(A,V)$, then the expectation of vehicles' CWs generated by FMBA results $CW(D) \leq CW(C) \leq CW(B) \leq CW(A)$. Therefore, in the considered example D has the highest probability to become the next forwarder of the message transmitted by vehicle V, since its waiting time is randomly chosen within the smallest CW among those assigned by FMBA to vehicles A, B, C, and D. This leads to a general reduction of the number of hops and time needed by a broadcast message to traverse its area of interest.

5.2.3 Example of FMBA

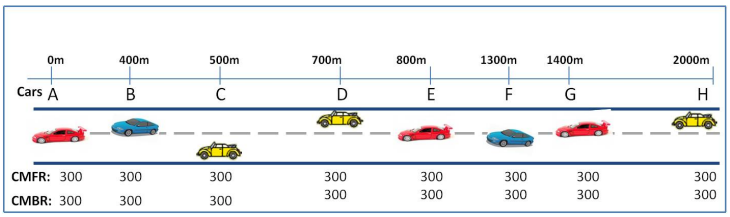
Since our analysis is based on the FMBA case study, in this section we present an example of execution of this algorithm. To help the readers, the example is presented in Figure 3.

We first suppose to have an initial state (Figure 5.2(a)), where all cars have initial values $CMFR = CMBR = 300m$; whereas the actual transmission range of each vehicle is $1000m$. We consider a $2000m$ portion of road and we suppose that car F sends the

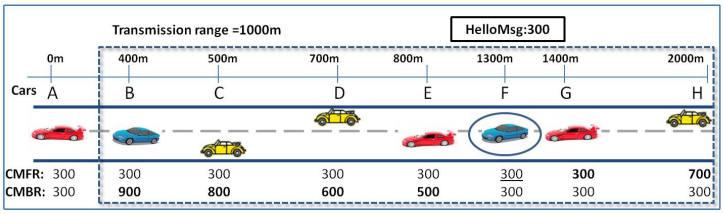
first $Hello$ message (Figure 5.2(b)) broadcasting its $CMFR$ (i.e., $300m$). This value is used by cars in front of car F to update their $CMBR$ values and by cars following car F to update their $CMFR$ values. The value of $CMBR$ (respectively $CMFR$) of each vehicle receiving the broadcast $Hello$ message is updated with the maximum among: i) the broadcast $CMFR$ value; ii) the previous value of $CMBR$ (respectively $CMFR$), and iii) the distance from car F. The rationale of this is that $CMFR$ represents how far a message was heard coming from the front of the car. Thus, when received in a broadcast message, this value is used to compute how far following vehicles are able to hear a message coming from their front. The value of $CMFR$ can hence be one of the parameters to determine the $CMBR$, which corresponds to the backward maximum transmission range (that will be declared when transmitting an alert message). Similar but reversed considerations could be made if we considered the frontward direction for alert message propagation.

In Figure 5.2(c), car D sends a second $Hello$ message broadcasting a $CMFR$ of $300m$ so that vehicles in D's transmission range can update their $CMBR$ and $CMFR$ as explained. Then, car G sends the third $Hello$ message (Figure 5.2(d)).

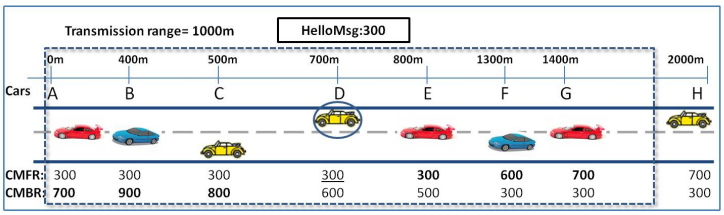
In the next step (Figure 5.2(e)), car C has to broadcast an alert message. We can remark that the algorithm has modified C's $CMFR$ and $CMBR$. The broadcast message issued by car C includes in its $MaxRange$ field the latest $CMBR$ value. We remark that the maximum transmission range, estimated by car C, after only three $Hello$ messages is $900m$ over an actual one of $1000m$. Cars following C and hearing the broadcast message can then compute their CW through Equation 5.1 so as to have a forwarder, possibly close to the end of C's backward transmission range.



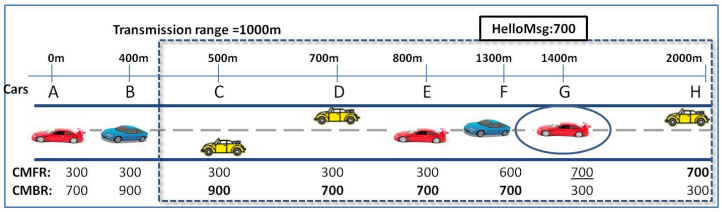
(a) Initial state



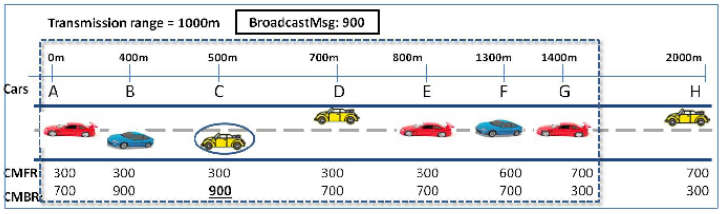
(b) First Hello message



(c) Second Hello message



(d) Third Hello message



(e) Broadcast message

Figure 5.2. – Example of a fast broadcast algorithm

5.3 Attack #1: Position-Cheating Attack

In this section, we present a position cheating attack. This attack is mainly linked to properties 2), 4), and 5) mentioned in Section I and its goal is to induce delay by increasing the CW of honest vehicles.

5.3.1 Overview

A malicious node could announce in a *Hello* message a false position being more distant than the real one. Then, honest nodes eventually receiving an alert broadcast message will compute unnecessarily large CWs, thus slowing down the forwarding process. For ease of presentation, Figure 5.3 depicts the impact of this attack reporting the CWs of some vehicles depending on their distance from the original sender/forwarder (vehicle V) of the alert message.

In particular, as the CW of each vehicle is computed through Equation 5.1, without the malicious vehicle the CW function should vary as shown by the continuous line in the Figure 5.3 (from its maximum in correspondence of vehicle V to its minimum at the end of the transmission range which is assumed to be close to vehicle D). Instead, if during the estimation phase, a malicious vehicle within V's transmission range sent a *Hello* message to declare a fake position corresponding to M' in the Figure 5.3, the transmission range estimation of vehicle V would be wrongly computed as the distance from V to M', instead of the distance from V to D. This leads vehicles A, B, C and D to wrongly compute their CWs with higher values. In fact, those nodes will consider the minimum CW in correspondence of position of M', as shown by the dotted line in Figure 5.3.

This simple, yet effective attack, modifies the computation of CW, increasing in average the contention period of each node before any forwarding transmission can take place, hence slowing down the transmission of the alert message.

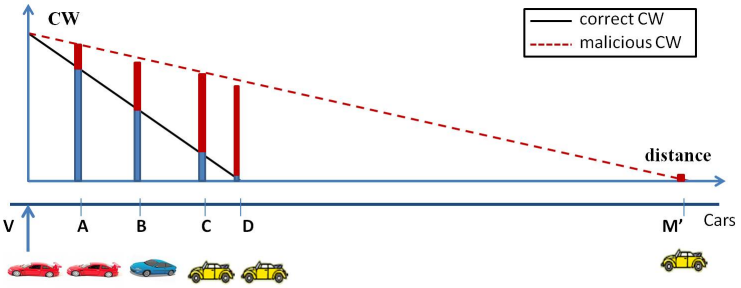


Figure 5.3. – Impact of distance cheating on the waiting time

5.3.2 Description

In this section, we present Algorithm 4 executed by a malicious vehicle M. In fact, it cheats about its claimed position, declaring a false position (Algorithm 4, line 2). Then, M broadcasts its *Hello* message (Algorithm 4, line 3) indicating its claimed position.

Algorithm 4: Position-Cheating attack executed by a malicious vehicle M

1 Input: *real_position*: (Real position of M);
claimed_position: (Claimed position of M);
vehicle_ID: ID of the vehicle M;
drm: declared max range of M;
$Hello$ msg: $Hello$ message generated by M;
2 $claimed\_position > real\_position$;
3 $M \rightarrow *$: Hello msg = $< vehicle\_ID, claimed\_position, drm >$;

5.4 Attack #2: Replay Broadcast Message Attack

In this section, we present the replay broadcast message attack which has the aim to enforce honest nodes to not collaborate and not forward the packets. This attack is mainly linked to properties 1), 2), and 3) mentioned in Section I. In particular, the adversary repeats the transmission of the same message, enforcing other vehicles to not forward packets.

5.4.1 Overview

In this scenario, we consider an honest node which broadcasts a message to all the receivers in its transmission range. We suppose that the adversary intercepts the broadcast message and rebroadcasts it without waiting. First, we remark that all the nodes that receive this same broadcast message from the front, the attacker push them to restart the broadcast procedure (as explained in the forwarding procedure of a broadcast message). Second, all the nodes that receive this message from the back, stop trying to forward this message. In fact, according to the FMBA, the messages has been already propagated over the considered vehicles, and these vehicles will exit the forwarding procedure. The adversary could repeat broadcasting the same message, pushing the nodes to not forward the packet, by just restarting at each time the broadcast process.

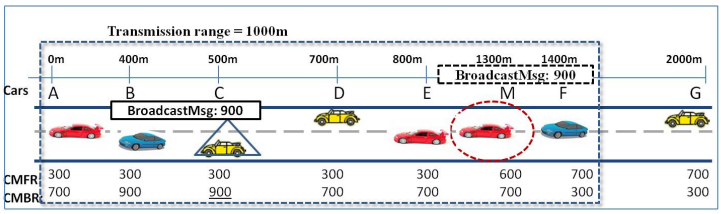
In more details, let us consider the scenario depicted in Figure 5.4(a). The honest car (C) forwards its messages. In this attack scenario, we suppose that the vehicle (M) is malicious and does not wait for the expiration of its time interval; it sends the message immediately. When receiving the message, vehicles which are behind M (car F in Figure 5.4(b)) restart the broadcast process, whereas nodes which are in front of M (cars D and E in Figure 5.4(b)) will exit the forwarding process.

To summarize this attack, a malicious node M could do some operations:

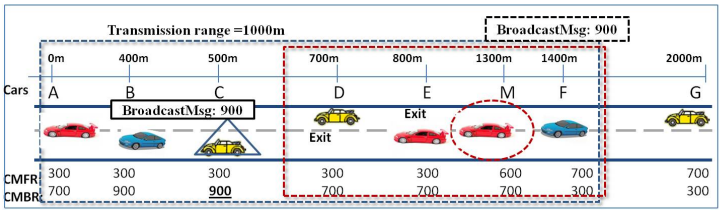
1. M does not modify the message but just broadcasts it.

   - Nodes which are behind M restart the forwarding process, thus wasting time.
   - Nodes which are in front of M exit the forwarding process.

   No one could forward the packet if the adversary repeats every message sent by the forwarder or the sender vehicle.



(a) Step 1



(b) Step 2

Figure 5.4. – Example of enforcing non cooperation attack

2. M modifies the broadcast message and forwards it with a high $MaxRange$ so as to generate slow forwarding hops with vehicles employing unnecessarily high CW values.
3. M forwards the message with a low $MaxRange$ in order to increase the probability that more than one vehicle will simultaneously attempt to forward the message, thus resulting in transmission collision and hence in forwarding delay.

5.4.2 Description

In this section, we present the Algorithm 5 executed by the malicious vehicle M. In fact, a source S broadcasts a message (line 2) and the malicious vehicle M retransmits it without waiting (line 4).

Algorithm 5: Enforcing non cooperation attack executed by the malicious vehicle M

1 Input S: the sender of the message;
broadcast msg: the broadcast message of S;
2 $S \rightarrow$*: *broadcast* msg;
3 M intercepts the *broadcast* msg and does not wait for the expiration of its waiting time;
4 $M \rightarrow$ *: *broadcast* msg;

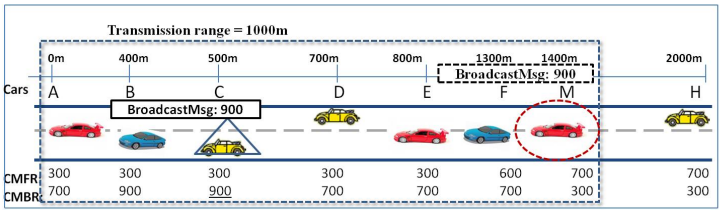
5.5 Attack #3: Interrupting Forwarding Attack

The goal of this attack is to degrade the network performances by impeding alert message relaying. This attack is mainly linked to properties 1), 3), and 4) mentioned previously.

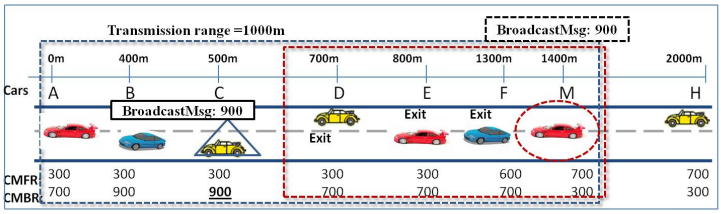
5.5.1 Overview

In this attack, the forwarder vehicle is malicious and tries to broadcast a message frontward but not backward. To do so, the malicious node has to be located at the end of the transmission range and be endowed with a directional antenna. By forwarding the alert message only frontward, vehicles in front of the malicious node will abort their forwarding procedure, as the message has already been sent farther than their position. On the other hand, vehicles behind the malicious node will simply not receive any message, thus interrupting the forwarding procedure.

Let us consider the attack scenario depicted in Figure 5.5(a) and Figure 5.5(b). The honest vehicle C broadcasts a message, while the chosen forwarder vehicle is M (using the forwarding procedure in FMBA algorithm). Assume that M is a malicious vehicle (Figure 5.5(a)) that forwards the message on front (not backward) by adjusting its transmission range. Vehicles (D, E, and F) will exit the forwarding process (Figure 5.5(b)) because they have received the same message from the back. Thus, the forwarded message will never be forwarded to the other cars. The impact of this attack in Figure 5.5(b) is that it avoids the transmission of messages. This attack could damage the network, especially when there are many malicious vehicles (see Figure 5.6) that collaborate together, in order to not forward or even limit the propagation of the alert message. These attacks can incur great security threats to vehicular communications. We observe (Figure 5.6) that n malicious nodes might collaborate together, in order to block the transmission of messages and not forward them in n zones.



(a) Step 1



(b) Step 2

Figure 5.5. – Impact of a malicious forwarder node

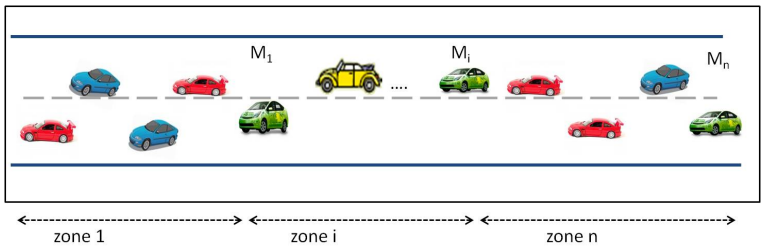


Figure 5.6. – Degrading performance attack executed by n malicious vehicles

5.5.2 Description

The malicious forwarder executes the following algorithm in order to stop the propagation of the alert message in the network. A broadcasts a message (Algorithm 6, line 2). The malicious vehicle intercepts it and transmits it to the vehicles in front of it (Algorithm 6, line 4). This attack disrupts the packet transmission process and blocks the transmission of the broadcast message.

Algorithm 6: Degrading performance attack executed by a malicious forwarder vehicle M

```
1 Input: broadcast msg : the broadcast message;
  A: the sender of the broadcast message;
2 A → ∗: broadcast msg;
3 // M is the forwarder of the message;
4 M → front vehicles of M: broadcast msg;
```

5.6 Solution to Attack #1: False Position Detection

Position dissemination is crucial for the fast broadcast algorithm [30]. Thus, forged position information has severe impact regarding the performance and security of the algorithm. We propose a detection mechanism that is able of recognizing nodes cheating about their position. Unlike other proposals described in the literature ([79], [80], [81]), our detection mechanism does not rely on additional hardware. Instead, our solution uses collaborative neighbors. We present an overview and a detailed description of our false position detection mechanism.

5.6.1 State of the Art Solutions for False Position Detection

Accurate information on position is crucial for IVC based vehicular safety applications. To this aim, detection mechanisms have been proposed in this context to recognize nodes

cheating about their location [79–84]. Position verification approaches can be grouped into two main categories [79, 85]: infrastructure based and infrastructure-less based approaches.

Infrastructure based Solutions

This approach uses special hardware dedicated infrastructure to verify the position of other vehicles. The solutions in [79] use multiple sensors to monitor and calculate trust values for position information. There are two classes of position verification sensors: autonomous and cooperating sensors. In fact, autonomous sensors work autonomously on each node and contribute their results to the overall trust ratings of neighbors. Cooperating sensors use the information exchange between the neighbors to verify positions.

In [80], the authors propose a specific use of infrastructure called "verifiable multilateration", employing three or more Road Side Units (RSUs). These RSUs send a synchronized challenge-response message to a vehicle. Then, the verification of its position is based on the consistency in response time calculation, to verify an announced location of a vehicle. Authors in [80] also exploit an history of movement of vehicles to detect forged data.

The solution in [81] uses verifiers at special locations. These verifiers define an acceptable distance between each others. The verification procedure then works as follows. First, the prover P sends a beacon containing its position. Then a verifier V replies with a challenge (a message) via radio. After receiving the challenge, P has to answer via ultrasound. The verifier computes the time required to receive the response (according to the defined acceptable distance for V). Then, P is approved to be within the region R of the verifier. This solution needs specific infrastructure: the verifiers. More specifically, these verifiers attempt to verify location claims for region R that are "near" V. Clearly, verifier nodes are assumed to be trusted and using secure communication among themselves.

An approach in [83] achieves position verification based on reception of beacons. First, the verifier nodes are divided in acceptors and rejectors. The acceptor nodes are distributed in the region R which is to be controlled. Verifier nodes could decide whether a vehicle is in a region R or not. In this approach, verifier nodes (acceptors and rejectors) are placed on distinct places, and are temporally synchronized among each other. If a vehicle P wants to prove its position (P wants to verify if it is in region R or not), it sends a beacon. As verifiers are placed in a region R, then the first verifier which receives the beacon could decide whether the position of P is acceptable or not. In fact, if the signal of P first reaches a rejector, then P is not in region R. Otherwise, if the first reached verifier is an acceptor, then P is approved to be in region R.

In [84], the proposed solution depends on two directional antennas. Each vehicle periodically sends a message containing its location together with its own two lists of front and back neighbors. A vehicle will decide on the relative positions of its one-hop neighbors based on the messages it receives.

Infrastructure-less Approaches

Infrastructure-less approaches could be classified on parameter based and model based approaches [79].

- **Parameter based approaches.** Vehicles check whether a claimed node's position is within a degree of accuracy from the actual one. This check is based on acceptable values of some network and traffic parameters, such as i) packet's timestamps consistency with current time; ii) acceptance range which assumes that no neighbor is further than the maximum transmission range. This approach assumes that the transmission range is fixed. The solution in [187] proposes distance bounding of vehicle's positions also involving verifiers and provers. This approach mainly detects distance enlarging fabrication. The verifier chooses the best common neighbor between the sender and him. This neighbor will give an estimated location of the prover. If that estimated location is not within a certain error distance of both verifier and neighbor, the verifier considers the node as malicious.
- **Model based approaches.** These solutions compare the regular behavior of the system and current actions to identify anomalies that could indicate malicious behaviors [188]. Each node periodically broadcasts its database containing information about observed nodes. When a broadcast is received, the contents are merged into the receiving node's database. Each node periodically examines events in its database searching for the scenario with the least number of malicious nodes. The disadvantage of this category of solutions is that a big search space of possible scenarios is needed to ensure efficacy.

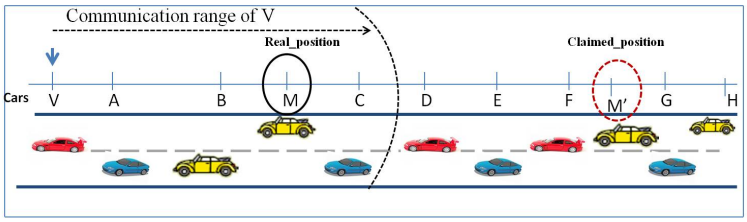
5.6.2 Overview of the Proposed Cheating Position Detection

For the sake of clarity, we present in this section some false position attacks, distinguishing among them depending on the claimed position and the real position of the verified vehicle. The main participants are the verifier vehicle V, the prover vehicle M and the other vehicles in the road. M claims a position in the $Hello$ message. The verifier vehicle V uses information, collected by collaborative vehicles, to decide whether M is a cheater or not. We elaborated three scenarios to deal with the false position attacks. The first scenario considers that there is at most one malicious vehicle in the road. The second scenario assumes that there is one malicious vehicle (either the prover vehicle M or one of the reporting vehicles). The third scenario assumes that the overhearing capabilities of the vehicles are asymmetric.

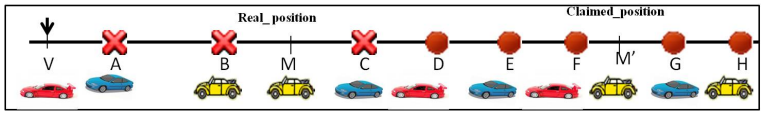
Scenario 1

In this attack scenario, we consider that there is at most one malicious vehicle. M could be either a malicious vehicle or an honest one. We distinguish three cases based on the real position and the claimed position of M.

A report is a message sent by a vehicle, to indicate that it does hear another vehicle message or not. In the first case, as presented in Figure 5.7(a), M claimed a position M' that is not included in the actual transmission range of the verifier V. Vehicle V collects information from neighbor vehicles in order to decide whether the claimed position of M is fake or not. The notation used to indicate whether a vehicle is hearing or not message M is reported in Figure 5.7(c). This legend is also applied to the other following figures. In Figure 5.7(b), the reports of the vehicles demonstrate that only A, B and C have heard the node M. Based on this reported information, the verifier V can determine that M is cheating about its position.



(a) Case 1: Claimed position is not under the transmission range of the verifier



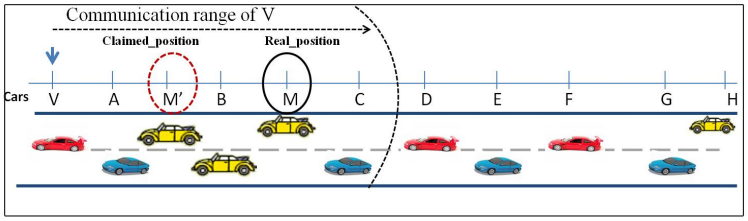
(b) Results of vehicles' reports

(c) Legend

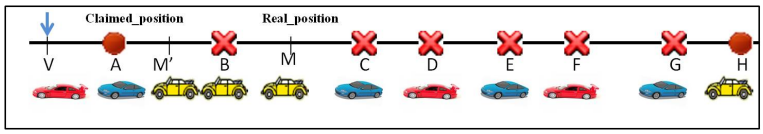
Figure 5.7. – Scenario 1 - Case 1

In Figure 5.8(a), we discuss the second case. In this case, the claimed position of M (position M') is in the communication range of the verifier V. Vehicles A, B, and C report their information about their neighbors (Figure 5.8(b)). These reports confirm that the considered vehicles (A, B, and C) received M's message, whereas vehicles D, E, F, G, and H did not hear it. Based on the collected information, the verifier node decides that the received information is consistent.

Considering the third case, as depicted in Figure 5.9(a), the verifier V is located between the real position and the claimed position of M. The verifier, in the third case, could confirm that the reported information is consistent. Thus, M might be successfully cheating; yet,



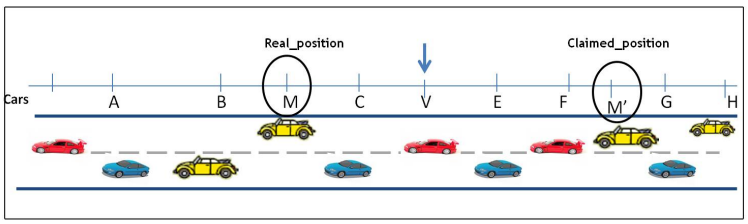
(a) Case 2: Claimed position and real position are within the transmission range of the verifier



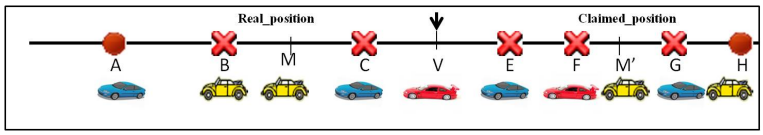
(b) Results of vehicles' reports

Figure 5.8. – Scenario 1 - Case 2

the impact of this false location does not lead to successfully modify the CW of V as the claimed position is between the positions of F and G, thus not affecting the computation of the maximum transmission range (Figure 5.9(b)).



(a) Real position and claimed position are on the transmission range of V



(b) Results of vehicles' reports

Figure 5.9. – Scenario 1 - Case 3

We could summarize these cases in Figure 5.10, which represents V's CW as a function of the distance of the different neighbors. In fact, the CW is divided into three regions based on the different collected reports and position of vehicles. Let us consider the three regions denoted by ($R1$), ($R2$), and ($R3$). Region ($R1$) represents the regular CW values (represented by the continuous line) of the vehicle V without an attack. Furthermore, if M's claimed position exceeds C's position (for example the position M'_{R_2}), this information could have an effect on the waiting time of other vehicles (A, B, C); until the claimed

position of the prover reaches the position of D (represented by the dotted line). If the claimed position of M exceeds D, in this situation the attack is detected. Region ($R2$) is limited by the positions of C and D, then a claimed position of M in this Region ($R2$) has a small effect on V's CW. Region ($R3$) represents the position of vehicles superior than the position of D. Vehicles D, E, F, G, and H are in Region ($R3$). If a malicious vehicle claims to be in Region ($R3$) (for example M'_{R_3} in Figure 5.10), it will be detected by V because most of the vehicles in this region report the non overhearing of M. Thus, the verifier V will detect M as a cheater.

To summarize, if the vehicle M claims a position in Region ($R1$), the verifier V found that: (i) the reported information is consistent, (ii) M could be cheating, and (iii) even if it cheats, there is no effect on V. In the second case, we consider that vehicle M claims a position in the Region ($R2$). Thus, the verifier V proceeds to the following assumptions: (i) the reports of the observed nodes are consistent, (ii) V could not detect whether M is cheating or not, and (iii) the claimed position of M, situated between the positions of C and D, changes the diagram and has an effect on V. In Region (R3) in Figure 5.10, the prover M claims a position located in this region. In this case, V decides that M is cheating based on the received reports of vehicles.

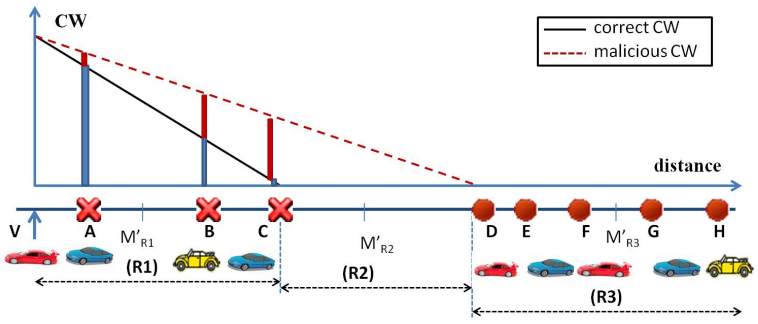


Figure 5.10. – Scenario 1: Verifier waiting time versus distance

Scenario 2

In this attack scenario, we consider that the malicious vehicle could be at most either M or one of the reporting vehicles. Let us focus on Figure 5.11. We split the regions into four parts: ($R1$), ($R2$), ($R3$) and ($R4$). ($R1$) includes the vehicles A, B, and C that they hear the message, and could be in the transmission range of the verifier V. $R1$ represents the regular CW values of the vehicle V without an attack (represented by the continuous line). The region ($R2$) represents the region between the last vehicle that heard the message and the first vehicle that did not hear the message. Region ($R3$) is the area including the first vehicle that did not hear the message, and the second vehicle that did not hear also the message. Region ($R4$) is the area including the vehicles that did not hear the message.

We consider that one of the reporting vehicles (B) is in Region ($R1$). The reporter (B) claims that it does not hear M, however it should normally heard the message. Thus, the verifier V could detect B as the only cheating vehicle because we assumed that there is one malicious node. A vehicle located in Region ($R2$) (for example the vehicle D) declares that it does not hear M. In this case, neither M nor the reporting vehicle (D) are detected as malicious. Vehicle V could not determine whether one of them is cheating or not. If the reporting vehicle is located in Region ($R3$), it does not hear the vehicle M. Thus, the verifier V considered that one of the two vehicles is cheating either M or the reporting vehicle. In another situation, let us consider that the reporting vehicle (G) is located in Region ($R4$), it transmits a report indicating that it does not hear M. Thus, the verifier V considers that M is the malicious vehicle.

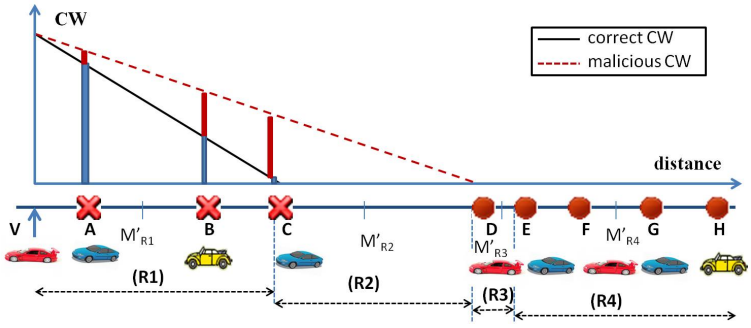
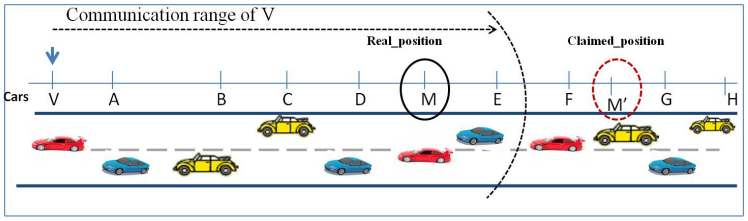


Figure 5.11. – Scenario 2: Verifier waiting time versus distance

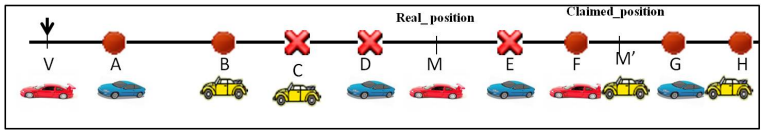
Scenario 3

We consider in this scenario asymmetric overhearing capabilities: even if a vehicle V can receive messages from M, the reverse could not be true. In this case, we consider Figure 5.12(a) in which V hears M, and M does not hear V. In Figure 5.12(b), we represent the reports of vehicles about their ability to receive messages from M. In fact, A, B, F, G, and H do not hear M. Moreover C, D, and E have heard the message of the prover M. The hearing capabilities of the vehicles are not the same. M does not hear V and also A and B do not hear M's message.

Vehicle V decides whether M is a cheater or not based on the received claimed position in the three regions (Figure 5.13). In fact, if the prover M claims to be in Region ($R3$), its claimed position (M'_{R_3}) will be detected as false. If M claims to be in Region ($R2$), the announcement of this location (M'_{R_2}) has an effect on the diagram of the verifier. However, M could not be detected as malicious. Finally we also consider the case where M claims to be in Region ($R1$) (for example in position M'_{R_1}). Thus, V could not detect M as cheating node, but the announcement of the location of M has no negative impact on the waiting time of V.



(a) Asymmetric hearing capabilities



(b) Results of vehicles' reports

Figure 5.12. – Scenario 3: Verifier waiting time versus distance

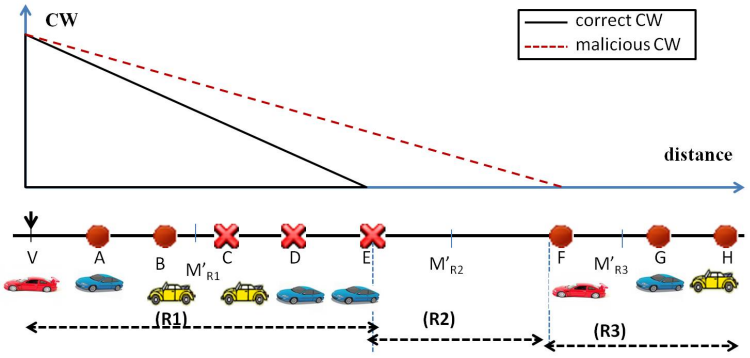


Figure 5.13. – Verifier waiting time versus distance

5.6.3 Description of the Proposed Cheating Position Detection

In this section, we describe our proposed position verification scheme. Our solution requires no infrastructure but only distributed messages exchanged between nodes to detect the malicious nodes. In fact, we discuss how the information of the vehicles could be propagated to other vehicles, in order to have a complete and a local observation. First, we discuss the structure of the transmitted message, and the timing of forwarding or transmitting the information. Second, these data could be collected from direct neighbors (in case nodes communicate directly) or from multi-hop neighbors (in case of indirect or asymmetric communication) between vehicles. Yet, we should limit the multi-hop propagation of this information within a limited region. Collected information should be recent and limited to the participating nodes. Third, in order to guarantee the authenticity of messages, nodes proceed to an authentication mechanism. To do this, we propose to transmit the information of vehicles in a modified *Hello* message. We present the sending and receiving procedure of *Hello* message. After receiving the different reports (the modified *Hello* messages) from

vehicles, each verifier executes locally a position verification procedure. Then, the verifier vehicle could detect whether the claimed position of a vehicle M is false or not.

Structure of the Modified *Hello* Message

In order to have the position of the vehicles and their neighbors, we propose to include these data into the *Hello* message instead of using additional structures. The first reason supporting this choice is that the *Hello* message is transmitted by a vehicle in each time interval (between 100 ms and 300 ms). A second reason is that the sender of the *Hello* message is chosen randomly. A third reason is to reduce the communication overhead and not generate another structure or another type of message. The modified *Hello* message includes the *vehicle_id*, the *vehicle_position*, *declared_max_range*, *timestamp*, *list_neighbors* which is the list of two hop neighbors, and a signature generated by the sender private key. We refer the reader to Figure 5.14, in which we present an element (neighbor) in the list of neighbors. This list is constructed by a set of elements of type Neighbor. Then, we clarify in each element of type Neighbor its *vehicle_id*, *vehicle_position*, and a list of indirect neighbors. In fact, a list of indirect neighbors is composed by a set of elements of type Indirect Neighbor. Each Indirect Neighbor (see Figure 5.15) is a structure which includes its *vehicle_id*, *vehicle_position*, *declared_max_range*, and *timestamp*. Moreover, the packet in Figure 5.16 contains a signature of this information by the sender private key. The Figure 5.16 illustrates the structure of the modified *Hello* message.

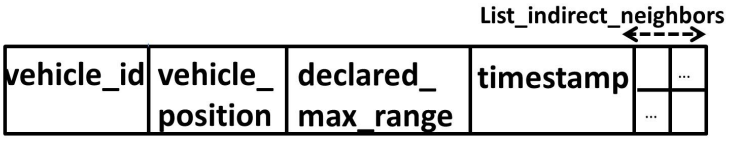


Figure 5.14. – Neighbor structure

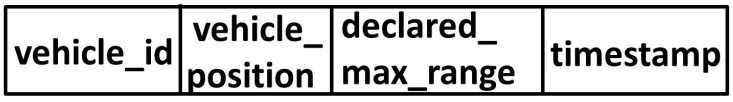


Figure 5.15. – Indirect neighbors structure

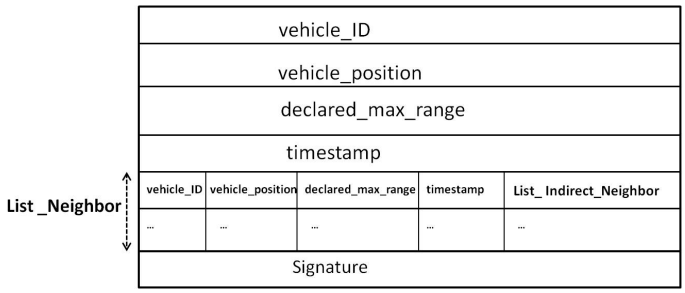


Figure 5.16. – A modified structure of *Hello* message

Sending *Hello* Message Procedure

We focus on the *Hello* message sending procedure (Algorithm 7). In every turn, each vehicle determines a random waiting time (lines 1 and 2). After this waiting time, if neither other transmission is heard nor collision happened (line 6), it proceeds with transmitting a *Hello* message. This *Hello* message includes *vehicle_ID* (line 7), the *timestamp* (line 9), the *vehicle_position* (line 8), the *declared_max_range* (line 10), the list of neighbors and their two hop neighbors: *list_neigh_S* (line 11). Furthermore, the sender uses its private key to generate a signature to the message (line 12), then it transmits the message (line 13).

Algorithm 7: Sending *Hello* message algorithm (executed by vehicle X)

1 Input: *list_neighbors*: list of neighbors of V,
vehicle_X: the identity of the sender X,
position_X: the sender position,
current_Time_X: current time of the sender,
$LMFR$, $CMFR$, *list_neigh_X*: the list of neighbors of X,
$K^X_{private}$: private key of the sender S,
H: hash function;
2 Output: *Hello* message ;
3 For each turn ;
4 *sending_time* := *random*(*turn_size*);
5 wait (*sending_time*);
6 **if** *not (heard_Hello_msg() or heard_collision())* **then**
7 *Hello_msg.vehicle_ID*:= *vehicle_X*;
8 *Hello_msg.vehicle_position*:= *position_X*;
9 *Hello_msg.timestamp*:= *current_Time_X*;
10 *Hello_msg.declared_max_range*:= $max(LMFR, CMFR)$;
11 *Hello_msg.list_neighbor*:= *list_neigh_X*;
12 *Hello_msg.signature*:= $K^X_{private}$
(H(*Hello_msg.vehicle_ID*, *Hello_msg.vehicle_position*, *Hello_msg.timestamp*,
Hello_msg.declared_max_range, *Hello_msg.list_neighbor*)) ;
13 transmit (*Hello_msg*);
14 EndFor

Receiving $Hello$ Message Procedure

The $Hello$ message receiving procedure is depicted in Algorithm 8. In particular, a vehicle receiving a $Hello$ message in line 2, generates the public key (line 3) using $f(sender\_id\_X)$ where f is a hash function. In line 4 of Algorithm 8, the receiver verifies the signature of $Hello$ message. Then, it checks for the freshness of message (line 6). Indeed, it could be an old message transmitted to the vehicles. This check is performed verifying the coherence between the time inserted in the message by the claiming vehicle (the sender of the $Hello$ message) and the current time of the receiver. Then, for each message that passes the previous checks, the receiver vehicle extracts the information ($sender\_id\_X$). The receiver checks whether this is the first received $Hello$ message carrying the $sender\_id\_X$. Then, the receiver simply stores ($sender\_id\_X, sender\_position\_X, declared\_max\_range$, $timestamp, list\_neighbor\_X$) to its list of neighbors (algorithm 8, line 9). Then, the receiver determines its own position (line 12), extracts from the $Hello$ message the $sender\_position$ (line 12), and the included estimation of the maximum transmission range (line 11), and determines the distance between itself and the sender (line 13). If the $Hello$ message is received from ahead, the value of $CMFR$ is updated (lines 14 and 15), otherwise $CMBR$ is updated (lines 16 and 17). In both cases, the new value is obtained as the maximum among the old one, the distance between the considered vehicle and the $Hello$ message sender, and the sender's transmission range estimation provided by the $Hello$ message.

Position Verification Procedure

After receiving reports ($Hello$ messages) from other vehicles, a vehicle V could decide whether the claimed position announced by a vehicle is correct or not. In Algorithm 9, we present the position verification algorithm executed by a vehicle V. In fact, V collects N reports (Algorithm 9, line 3). For each received report, it checks whether the claimed vehicle M is in the list of neighbors of these vehicles. Based on this information, V could decide if the claimed position is in a certain Region. If the claimed position is in Region ($R1$) (line 9), then the vehicle could be a malicious one, but this has no negative effect on V. Otherwise, if the claimed position is in Region ($R2$) (line 12), then the position of M has an effect on V, and M is classified as suspicious. Finally, if the claimed position is in Region ($R3$) (line 15), M is detected as a cheater. If there is at most one malicious node, we refer to the case 1 of scenario 1. If there is one malicious vehicle in the road (M or one of the reporting vehicle), then we refer to the scenario 2.

The Utility of the Timestamp and the Vehicle Position in the Modified $Hello$ message

In this section, we present the utility of some received transmitted information and their role in preventing the propagation of the adversary's message. We consider the scenario

Algorithm 8: Receiving $Hello$ message algorithm (executed by vehicle V)

```
1  Input: list_neighbors: list of neighbors of V,
   current_Time_V: the current time of V,
   sender_id_X: the identity of the sender,
   sender_position_X: the field corresponding to sender position,
   currentTime_X: current time of the sender included in the message,
   drm_X: the declared maximum range received, list_neigh_X: the list of neighbors in the received
   message,
   signedHelloMsg_X: the received signature ;
2  < sender_id_X, sender_position_X, currentTime_X, drm_X,
   list_neigh_X, signedHelloMsg_X > ;
3  K_X^Pub ← f(sender_id_X);
4  if H(sender_id_X, sender_position_X, currentTime_X,
   drm_X, list_neigh_X) ≠ K_X^Pub(signedHelloMsg_X) then
5      handle this exception ;
6  if IsNotCoherent (current_time_X, current_time_V) then
7      handle this exception ;
8  if IsNotPresent (list_neighbors, sender_id_X) then
9      add(list_neighbors, < sender_id_X, sender_position_X,
       currentTime_X, drm_X, list_neigh_X, signedHelloMsg_X >) ;
10 mp := my_position() ;
11 sp := sender_position ;
12 drm := declared_max_range ;
13 d := distance(mp, sp) ;
14 if (received_from_front(Hello_msg)) then
15     CMFR := max(CMFR, d, drm) ;
16 else
17     CMBR := max(CMBR, d, drm) ;
```

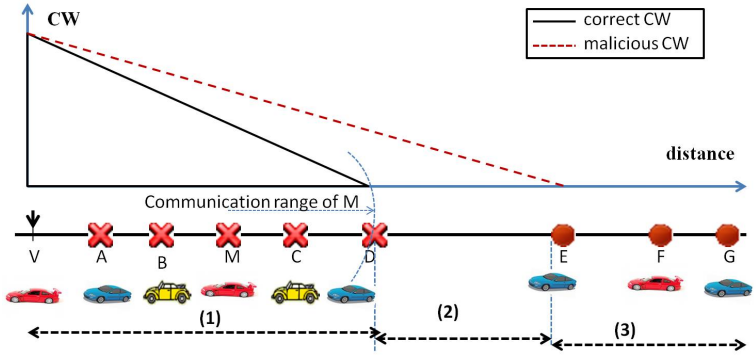
where two honest vehicles A and G are distant from each other. We consider that we have the following scenario depicted in Figure 5.17(a). If M is a malicious vehicle and has a communication range as presented in Figure 5.17(a), then it has no effect in modifying the list of neighbors of other vehicles. But, if the communication range of M is as presented in Figure 5.17(b) then E, F, and G will hear M. Thus, the diagram of V changes, and E, F, and G indicate that M is their neighbor. We could also consider the case where there are two malicious nodes collaborating together (Figure 5.17(c)). Vehicle M_1 is a malicious vehicle located near A, whereas M_2 is a malicious vehicle located near G. A sends a $Hello$ message signed by its private key. M_1 takes the same message and forwards it to M_2. Then, M_2 forwards it to G. When receiving this message, G will hear the message of A and think that A is its neighbor. Then, E and F suppose that M_1 is their neighbor, and thus the diagram of V changes. This scenario could be used by malicious nodes collaborating together, in order to influence vehicles that they are neighbors of one malicious vehicle.

In order to detect this attack or to limit its impact, a vehicle can check the sending time of the message. When receiving the message, if it was sent at a late time with respect to the receiving time, then the vehicle could detect it. Even if the message of the malicious node has propagated to some vehicles, the verification of the current time and the time

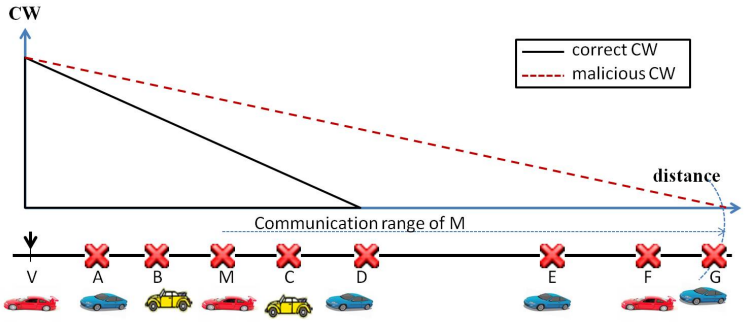
Algorithm 9: Position verification algorithm (executed by vehicle V)

```
1  Input: V: the verifier node executes the verification algorithm
   M: the vehicle for which V wants to verify its claimed position
   M1,...MN: N messages collected by V
   Claimed_position of M
   Mi =< sender_i, sender_position_i, timestamp_X,
   drm_X, list_neigh_i >;
2  Output: State of M is malicious or suspicious;
3  //i is the sender of the report Mi
   For all i ∈ 1, ...N do
   extract (list_neigh_i) from the report Mi;
4  if M ∈ list_neigh_i then
5  |  i hears M ;
6  else
7  |  i does not hear M ;
8  End For.
   V sets its CW with respect to the different information;
9  if claimed_position ∈ Region (R1) of V then
10 |  M is not detected as malicious and the claiming distance has no effect on V;
11 else
12 |  if claimed_position ∈ Region (R2) of V then
13 |  |  M is not detected and has an effect on V ;
14 |  else
15 |  |  if claimed_position ∈ Region (R3) of V then
16 |  |  |  M is detected as malicious;
```

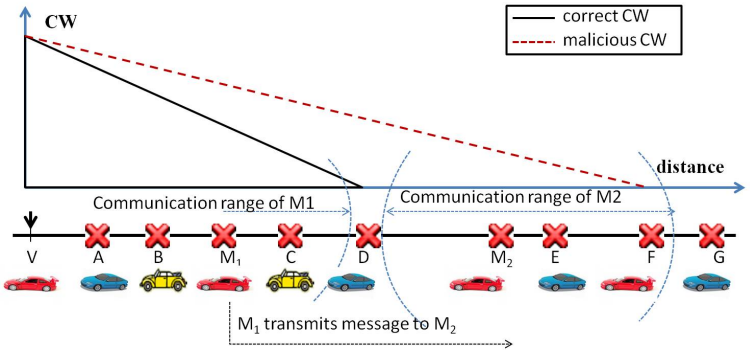
included in the message could prevent the forwarding of this message. In fact, based on time, the vehicle can limit the propagation of the message (the malicious vehicle tries to forward it, in order to convince other vehicles that they are neighbors of V). At a certain time, a vehicle can detect that it is a late message (because forwarding takes time). The vehicle could use also the position information of the sender of the message. Thus, if the position of the sender is too distant from the receiver, then the received information is not consistent and the verifier can detect the malicious vehicle.



(a) Case 1: a malicious node has a small communication range



(b) Case 2: a malicious node has a large communication range



(c) Case 3: two malicious nodes collaborating together

Figure 5.17. – Falsing list neighbors

5.7 Solution to Attack #2: Anti-Replay Protection

In this section, we present how an honest vehicle could detect a non cooperation attack or a replay message attack presented in section V. Our proposed solution is based on appending a *timestamp* to the broadcast message, and storing a table in each vehicle in order to protect against this attack. This storing table contains a list of items. Each item contains the identity of the sender or the forwarder of the broadcast message, the timestamp, and the broadcast message.

5.7.1 State of the Art Solutions of Anti-Replay Protection

Anti-replay protection allows a receiving node to identify replayed messages and discard them to guarantee of weak or sequential freshness. IPSec, for example, uses an incrementing counter included with each packet to ensure sequential freshness [189]. IPSec's sliding window allows for out-of-order packet arrival; as long as packets arrive in order, or within the sliding window, they are accepted regardless of the time interval between packets. A more strict guarantee is strong freshness, which ensures that a message was sent within a certain, usually short, preceding time period. Strong freshness can be provided in two ways: either via a challenge or response sequence, usually involving a nonce value, or using a timestamp that is added to the packet during transmission and compared by the receiver with the global clock. All of these anti-replay techniques require traffic authentication so that counter values and timestamps cannot be modified without detection.

One of the proposed anti-replay mechanism is included in the SPINS protocol for wireless sensor networks [16]. In this mechanism, weak freshness is a byproduct of CTR-mode encryption. A monotonically increasing counter is used as a nonce value by encrypting outgoing packets, and this counter is also used as an anti-replay counter.

Timestamps are a practical alternative for anti-replay support in VANETs. Timestamps require that nodes maintain fine grained synchronization, and that all nodes synchronize with a global clock. Synchronization in vehicular communication is not a problem as vehicles could use the GPS clock information. Thus, in our approach to prevent this kind of replay attacks, we use timestamps.

5.7.2 Overview of our Proposed Solution

Let us assume the malicious node received a broadcast message, and then transmitted this message, without waiting for the expiration of any contention avoidance delay. In this case, the nodes receiving this broadcast message from their front restart the broadcast procedure. At the same time, nodes receiving the message from their back abort their message forwarding procedure, since the message has been already propagated over them along the direction of propagation.

In order to detect this malicious behavior, each sender of the broadcast message should indicate in the transmitted broadcast message a *timestamp*. Then, the broadcast message includes a *vehicle_id*, a $MaxRange$, a *timestamp* (indicating the current forwarding time). Furthermore, each vehicle maintains a stored table in which it indicates in each line of the table the following fields: *vehicle_id*, a *timestamp*, and a *broadcast* message. Using these stored information, the vehicle could detect if it received the same message from its front vehicles, in a short period. The message will not be taken into consideration as it is a replayed message.

5.7.3 Description

In this section, we provide a detailed description of our proposed solution to detect malicious vehicles based on message exchange. To this aim, we discuss the structure of the broadcast message and the need for a stored table to detect replayed messages. Furthermore, in order to guarantee the authenticity of messages, nodes proceed to an authentication mechanism. To do this, we propose to transmit the timestamp in the broadcast message, and to store in each vehicle a table containing the last previous transmitted broadcast messages. We present the sending and the receiving procedure of a broadcast message. After receiving the broadcast message from a forwarder vehicle, a receiver of the message executes locally a verification procedure. Then, this vehicle could detect whether the transmitted message is a replayed message or not. The verifier vehicle uses the information stored in its table and the forwarding time to determine if it is a replayed broadcast message or not.

Broadcast Message and Stored Table

Figure 5.18 and Figure 5.19 describe the structure of the broadcast message, as well as the stored table in each vehicle.

When receiving a broadcast message, the vehicle should verify whether it is a fresh message or not. In the latter case, it drops the message. Therefore, the structure of the modified broadcast message includes a *timestamp* added by the sender.

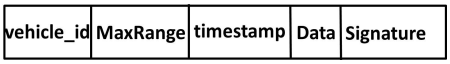


Figure 5.18. – Modified broadcast message

vehicle_id	timestamp	broadcast message
M	t_M	m_M
...	...	...

Figure 5.19. – Stored Table

Sending Broadcast Message Procedure

A vehicle sending the broadcast message proceeds as follows. With the help of Algorithm 10, we explain our scheme's behavior during the procedure for sending broadcast messages. Each vehicle proceeds with transmitting a broadcast message; this broadcast message includes the *vehicle_ID* (line 2), the $MaxRange$ (line 3), the *timestamp* (line 4), and the *data* (line 5). Furthermore, the sender uses its private key to generate a signature for the message (line 6), before broadcasting it (line 7).

Algorithm 10: Sending broadcast Message algorithm (executed by a vehicle X)

1 Input: *broadcast msg* : the broadcast message;
$vehicle\_Id\_X$: vehicle ID of the sender X;
$MaxRange\_X$: MaxRange of vehicle X;
$timestamp\_X$: timestamp or current time of vehicle X;
$data\_X$: generated data;
$Kpriv\_X$: private key of sender X ;
2 $vehicle\_ID \leftarrow vehicle\_Id\_X$;
3 $MaxRange \leftarrow MaxRange\_X$;
4 $timestamp \leftarrow timestamp\_X$;
5 $data \leftarrow data\_X$;
6 $signature \leftarrow$
$\{vehicle\_Id\_X, MaxRange\_X, timestamp\_X, data\_X\}$ $\_Kpriv\_X$;
7 *broadcast msg* $\leftarrow$
$\langle vehicle\_ID, MaxRange, timestamp, data, signature \rangle$;

Receiving Broadcast Message Procedure

The broadcast message receiving procedure is presented in Algorithm 11. In particular, a vehicle receiving a broadcast message (line 2), checks for the freshness of message (line 4). This check is performed by verifying the coherence between the timestamp, included in the message, by the claiming vehicle (the sender of the broadcast message) and the receiver's current time. In line 7 of Algorithm 11, the receiver verifies the signature of the broadcast message. Then, it checks whether the received broadcast message had already been transmitted before, using its stored table ($verified\_stored\_table()$). If the message is received for the first time, then the vehicle simply stores ($vehicle\_id\_X$, $timestamp\_X$, *broadcast msg*) in its stored table which is the role of $add\_message\_to\_stored\_table()$ (line 10). Otherwise, the *broadcast msg* is a replayed message, and thus it will be dropped (line 12).

Algorithm 11: Receiving broadcast message algorithm (executed by a vehicle V)

```
1  Input: broadcast msg: the received broadcast message;
   f: hash function;
2  broadcast msg =
   ⟨vehicle_Id_X, _, timestamp_X, Data_X, Signature_X⟩;
3  K_pub_X = f(vehicle_Id_X) ;
4  if not (verified_timestamp(current_time, timestamp_X)) then
5  |  V drops the message;
6  else
7  if verified_signature() then
8  |  if verified_stored_table() then
9  |  |  accept_message();
10 |  |  add_message_to_stored_table() ;
11 |  else
12 |  |  drop packet;
13 |  |  handle exception();
14 else
15 |  drop packet;
16 |  handle exception();
```

5.8 Solution to Attack #3: Interrupting Forwarding Attack Detection

To detect malicious vehicles attempting to jeopardize FMBA by impeding the propagation of an alert message, we propose a mechanism that detects when vehicles along the road have not received the message. Our solution requires a vehicle acting as a verifier (as described in Section 5.8.2 and 5.8.3). Each forwarder vehicle should give a proof of relaying the broadcast message. Please note that we do not need any specific mechanism to elect the verifier, since any node can act as such (in fact, messages for the verifier are not intended for any specific receiver). We can just assume that any node in the area will also act as verifier. Of course a verifier which is a malicious node would not report any problem, even if an attack is detected. However, to detect the attack and take further actions (e.g., rebroadcast the blocked message) it is enough that at least one non compromised verifier detects the attack. In the following, we present an overview of the security scheme as well as a description of it.

5.8.1 State of the Art Solutions of Detecting Misbehaving Forwarders

Malicious vehicles are compromised vehicles that are willing to put an effort to introduce some damage. Hence malicious nodes are different from selfish ones, since selfish just do not want to use their resources for the sake of protocol's success. Many studies enhance the cooperation between vehicles and incentive them to forward messages. Major contributions in incentive cooperation in multi-hop communications use either reputation based schemes ([147], [190], and [191]) or credit based schemes [192], or hybrid schemes [193]. We underline that the type of attack that we described, in section VI, deals with malicious behavior of

nodes, not selfishness. In fact, the malicious node tries to transmit packets only frontward and not backward, in order to stop the propagation of the message. Many efforts ([147], [190], and [191]) have been done to detect misbehaving nodes that do not forward the packets. We classify these works on reputation based schemes and credit based schemes.

Reputation based Mechanisms

The reputation of a node increases when it carries out correctly the task of forwarding the packets. The main problem of the reputation based mechanisms is that they work in the inefficient promiscuous mode [194]. Once a node's reputation degrades to a threshold, the node is identified as dishonest or selfish. In [195], a watchdog and a path-rater are implemented in each node. The watchdog overhears the medium to detect whether the next-hop node forwards the packets or not. The path-rater module chooses the path that avoids selfish nodes based on the watchdog's notifications. The major disadvantage of this scheme is that it incurs extra loads on the honest nodes. In [196], it is shown that the reputation based mechanisms punish the selfish nodes at the expense of decreasing the throughput of cooperative ones.

Credit based Mechanisms

In the credit based mechanisms, the cooperative nodes earn credits in order to relay packets generated from other nodes, and spend them to relay packets. In Nuglets [197], a tamper proof device is installed on each node, thus it allows nodes to store their credits and secure their operations. However, this is not secure and realistic because they assume that the device can not be tampered. In [144], the authors propose SIP, in which the destination node sends a payment receipt to the sender which issues a REWARD packet. The REWARD packet increments the credit counters stored in the intermediate nodes. The disadvantage of this scheme is that each packet needs three trips between the source and destination nodes.

The credit based schemes proposed in the literature, to detect misbehaving or selfish nodes could not be applied to our specific problem, since malicious nodes could forward only on one direction (frontward) and not backward. The goal of the compromised nodes is to stop the propagation of the alert message, thus it could induce damages and accidents. The approach we propose, is based on the fact that each forwarder node has to sent a proof of receipt to a verifier node. The receipt message includes information that helps the verifier node to detect the misbehavior of the node.

5.8.2 Overview

The kind of attacks that we described, in section VIII, deals with malicious behavior of nodes, not selfishness. In fact, the malicious node tries to transmit packets only frontward

and not backward (by adjusting its transmission range). The aim of this malicious vehicle is to stop the propagation of the message. To prevent this attack, neighbor nodes have to collaborate together to confirm whether this node is misleading or not. Therefore, we have to define a strategy able to confirm that backward neighbors have not received the message. To reclaim this, each forwarder vehicle should transmit a receipt message to a verifier node, indicating that its message was relayed. The verification will be done by a verifier node able to use its knowledge of the vehicles on the road, to decide whether there are some malicious vehicles, that are trying to impede the correct propagation of the broadcast message.

In the following, we discuss the structure of the receipt message, as well as the transmission of this message by the forwarder nodes, and the verification operation done by the verifier node. The attack is more specific to $FMBA$, and different from what was studied in the literature [192], [193].

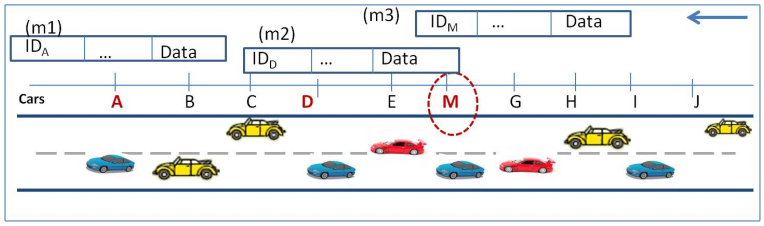
To simplify our presentation, we refer the reader to Figure 5.20(a) in which we have the vehicle A sending a broadcast message (represented by $m1$), which forwards this message to the forwarder vehicle D. Then, D transmits the broadcast message ($m2$) to the receivers in its communication range. As a consequence, M is chosen as a forwarder when applying the forwarding procedure in FMBA. Unfortunately, M is a malicious vehicle and thanks to a directional antenna tries to send messages only frontward (broadcast message ($m3$)). The goal of M is to stop the propagation of the broadcast message, by blocking vehicles in the front from forwarding the alert message, while backward vehicles (G, H, I, and J) will never receive the alert message (see Figure 5.20(b)).

5.8.3 Description

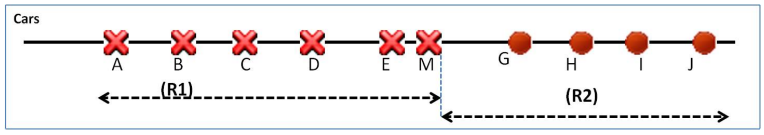
In this section, we detail the transmission of a receipt message by a forwarder vehicle, as well as the verification performed by the verifier node.

Receipt Message

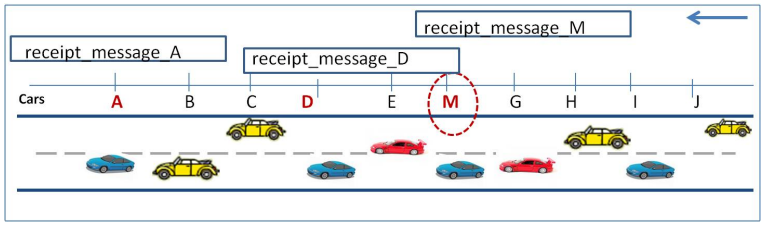
In order to send a proof of packet relay, the forwarder vehicle has to create a receipt message. In fact, forwarders contact the verifier node at least once during each time interval to send their receipts. After forwarding a broadcast message, the forwarder sends to the verifier node a receipt message containing the *vehicle_id*, the *vehicle_position*, the *timestamp*, and a *signature* generated by the forwarder private key. We refer the reader to Figure 5.20(c), in which we present an example of receipt messages generated by a forwarder. The verifier node collects authenticated receipts (containing the *vehicleidentity*, *timestamp*, and the *receiptmessage*), and performs some verifications to detect possible malicious forwarders trying to stop the propagation of the broadcast message.



(a) Step 1: Transmitting Broadcast messages



(b) Step 2: Results of overhearing messages



(c) Step 3: Sending receipt messages

Figure 5.20. – Example of non propagation of broadcast message detection

Sending Receipt Message Procedure Executed by a Forwarder Vehicle

A vehicle sending the receipt message proceeds as described by Algorithm 12. Focusing on the receipt message sending procedure, each forwarder transmits a receipt message after relaying a broadcast message. This receipt message includes the *vehicle_ID* (line 5) and the *timestamp* (line 6). Furthermore, the sender uses its private key to generate a signature to the message (line 9) before transmitting it (line 10).

Algorithm 12: Sending receipt message algorithm (executed by a vehicle X)

1 Input: *receipt msg* : the receipt message;
  $A \rightarrow *$: *broadcast* msg;
2 *vehicle_id_X*: the vehicle Id of X;
3 *timestamp_X*: the timestamp of X;
4 *K_priv_X*: the private key of X;
5 $vehicle\_ID \leftarrow vehicle\_id\_X$;
6 $timestamp \leftarrow timestamp\_X$;
7 $signature \leftarrow \{vehicle\_ID, timestamp\}\ \_Kpriv\_X$;
8 //generating the receipt message
9 $receipt\_message \leftarrow \langle vehicle\_ID, timestamp, signature \rangle$;
10 *transmit* $(vehicle\_ID, timestamp, signature)$;

Receiving Receipt Message Procedure Executed by a Verifier Node

The receipt message receiving procedure executed by a verifier node is depicted in Algorithm 13. In particular, a vehicle receiving a broadcast message (line 2) generates the public key (line 6) using $f(vehicle\_id\_X)$ with f is a hash function. In line 10 of Algorithm 12, the receiver verifies the signature of the receipt message; then, for each message that passes the previous checks, the receiver vehicle extracts the information ($vehicle\_id\_X$) and ($timestamp\_X$), and stores ($vehicle\_id\_X$, $timestamp\_X$, $signature\_X$) (line 13). At this point, the verifier node can verify using the different collected receipts whether there is a propagation of the broadcast message or not, since it knows the position of the different vehicles. If the information carried by the last forwarder vehicle is not consistent with the dispersion of vehicles on the road, this means there are neighbors that have not received the broadcast message. In this case, the verifier can detect that one of the forwarder vehicles is malicious, as it does not forward packets backward.

Algorithm 13: Receiving receipt message algorithm (executed by a vehicle V)

```
1  Input: receipt msg: the receipt message of vehicle X;
   vehicle_id_X: the received vehicle Id ;
2  timestamp_X: the received timestamp;
3  signature_X: the received signature ;
4  K_pub: public key;
5  receipt_message ← ⟨vehicle_ID_X, timestamp_X, signature_X⟩ ;
6  K_pub ← f(vehicle_id_X) ;
7  if not (verified_timestamp(current_time, timestamp_X)) then
8  |  V drops the message;
9  else
10 if verified_signature() then
11 |  if verified_consistency then
12 |  |  accept_message();
13 |  |  Add_message_to_stored_table() ;
14 |  else
15 |  |  handle exception;
16 else
17 |  drop packet;
```

5.9 FS-MBA

In this section, we present FS-MBA, i.e., a global view of the solutions in the previous sections. In the remainder of this section, we provide a security overhead analysis for the FS-MBA algorithm (Section 5.9.1).

To make a global view of our secure solution, Algorithm 14 illustrates the merge of the algorithms presented before (to tackle different type of attacks).

In particular, upon receiving a message (Algorithm 14, line 2), a vehicle does the following checks. First, the receiver checks whether the message is a *Hello* message (line 3).

Algorithm 14: FS-MBA

```
1  Input: type_message: the type of the received message, i.e Hello message, or broadcast message or
   receipt message;
2  msg: the received message;
3  if ⟨ type_message== Hello message⟩ then
4  |   Receiving Hello message algorithm (Algorithm 8) ;
5  |   Position verification algorithm (Algorithm 9);
6  else
7  |   if ⟨ type_message == broadcast message⟩ then
8  |   |   Receiving broadcast message algorithm (Algorithm 11);
9  |   else
10 |   Receiving receipt message algorithm (Algorithm 13);
```

If this condition holds, the vehicle verifies the presence of the cheating position attack (line 5). If the received message is a *broadcast* message (check done in line 7), the vehicle verifies whether there is a replay broadcast message attack. The verification is done by executing Algorithm 11. Finally, if the message is a *receipt* (check done in line 9), the node verifies whether there is an interrupting forwarding attack (line 10), by running Algorithm 13.

5.9.1 Security Overhead

Security always comes at a price, which translates into processing, bandwidth and storage overhead. Securing FMBA also costs some additional communication and computation overhead, due to the generation and the verification of packet signatures. Moreover, vehicles send information (e.g., position) very frequently. Each $Hello$ message has to be signed and sent to all vehicles in the communication range. In Figure 5.2, we summarize the communication, the computation and the storage overhead experienced by a vehicle X for each message. In Table 5.2, we denote the size (in terms of bytes) of a field S by $\|S\|$. We denote by $sign$ the signature of the message, $sign\_gen\_op$ represents the signature generation operation, and $sign\_verif\_op$ is the signature verification operation. Let N_1 (N_2) be the number of direct (indirect) neighbors (see Section 5.6.3) for a vehicle X. Let V_id be the vehicle identity, and V_pos indicates the vehicle position.

		Overhead
Communication	per Hello msg (send./rec.)	$\|\|timestamp\|\|+\|\|sign\|\|$ $+N_1$ (V_id+V_pos $+drm$ $+\|\|timestamp\|\|+$ $N_2(\|\|V\_id\|\|+\|\|V\_pos\|\|$ $+\|\|drm\|\|+\|\|timestamp\|\|))$
	per broadcast msg (send./rec.)	$\|\|timestamp\|\|+\|\|sign\|\|$
	per receipt msg (send./rec.)	$\|\|receipt\_msg\|\|$
Comp.	per each msg (send.)	$sign\_gen\_op$
	per each msg (rec.)	$sign\_verif\_op$
Storage	per Hello msg (send./rec.)	-
	per broadcast msg (send./rec.)	$\|\|broadcast\_msg\|\|$
	per receipt msg (send./rec.)	$\|\|receipt\_msg\|\|$

Table 5.2 – FS-MBA Overhead

5.10 Performance Evaluation

We carried out an extensive experimental study to test FMBA under the three different attacks (Attack #1, Attack #2, and Attack #3). The main tool utilized for our experiments is the well known NS-2 simulator (version ns-2.29) [198]. We use the Wireless model two-ray ground reflection, the type of road is highway with multiple lanes, and vehicle's speed is between $70 - 140$ km/h. The contention window parameters are $CWMin = 32$ slots, and $CWMax = 1024$ slots. The number of slots for $CWMin$ and $CWMax$ is inspired by the standard IEEE 802.11. A small time slot value could result in increasing the number of collisions, with a lack of reliability on delivering messages. According to the experiments in [31], a time slot of 200 μs provides both a fast and reliable broadcast message delivery. The idle time before $Hello$ message generation is 100 ms. The application message size is 200 Bytes, and $Hello$ message size is 50 B. If no alert message happens, there is a transmission of $Hello$ messages. Our configuration parameters are summarized in Table 5.3. We should mention that, in our evaluated scenarios, the malicious node is randomly positioned within the transmission range of the first sender.

Parameter	Value
Area of Interest	7 TR
$Hello$ message size	50 B
$Broadcast$ message size	200 B
Idle time	100 ms
$CWMin$	32 slots
$CWMax$	1024 slots
Vehicle speed	$70 - 140$ km/h
Slot time	200 μs
Simulation time	40 s
Number of simulations	100

Table 5.3 – Configuration Parameters

5.10.1 Simulation Environment

To provide a clear and general analysis of the proposed schemes, not affected by the implementation details of any particular wireless technology (e.g., 802.11 b/g/p or others), we used a discrete progression of the time based on slots to represent all the time related variables (such as MAC layer contention windows, random waiting, and time intervals). We implement and evaluate the performances of three protocols: the original FMBA; FMBA with different attacks; and FS-MBA. We carried out an extensive set of scenarios, and for each of them we run more than 100 simulations averaging their outcomes to produce chart. For each scenario, we use three different transmission ranges $TR = 300$ m, $TR = 650$ m, and $TR = 1000$ m to show how our systems scale. The choice of focusing on 300 m and 1000 m of transmission range comes straightforward from the $IEEE$ 802.11 p draft, which indicates these two values as the boundaries for a highway scenario [199]. We choose also the value of $TR = 650$ m as an intermediate value between $TR = 300$ m and $TR = 1000$ m.

In the original FMBA, every vehicle in the platoon computes a random waiting time within the contention window before forwarding the message. The adopted contention window is initially set to $CWMin$ and follows a general back off mechanism by which its value doubles every time a transmission attempt results in a collision and decreases linearly with every successful transmission. We let the simulation run for 37 seconds, after which the first vehicle in the platoon generated an alert message. We compare the various schemes, we analyze their ability in quickly forwarding the messages to all interested vehicles: FMBA, FMBA with attacks, and FS-MBA. For each attack, we are interested in evaluating some performance metrics. In the first attack Attack #1, we focus on the following metrics: the average number of slots waiten before transmitting a message, the number of hops required to broadcast the alert message, the estimated transmission range of the sender and the first forwarder of the message, the rate of collisions, and the claimed position of the attacker. In the second and the third attack (Attack #2 and Attack #3), we focus on the average number of slots waiten before transmitting a message, and the percentage of message propagation.

FMBA uses a low transmission rate since it generates an alert message only when an abnormal behavior happens. Thus, the overhead due to *Hello* messages employed by our protocols is very limited, i.e., less than 1 Kb/s within a transmission range area. The alert message has to be propagated from a certain vehicle to all following vehicles in an area of interest of $7 \times TR$. We choose $7 \times TR$ as there is no point in transmitting an instantaneous alarm farther. However, we could set a larger (or smaller) area-of-interest if it is more appropriate. The number of vehicles per Km of (multi-lane) road varies from 25 to 400, representing different density levels; the factual range TR is variable ($TR = 300\ m$, $TR = 650\ m$, and $TR = 1000\ m$). Our first assumption is that for all the different scenarios, the malicious node is on the first hop, which means that the attacker is on the transmission range of the first sender.

5.10.2 Simulations Outcomes

In the following, we present the performances of our protocols under several scenarios. We evaluate FMBA under Attack#1, and we evaluate our position cheating detection attack. We carried out several scenarios to evaluate the impact of Attack #2 on the performances of FMBA. Finally, we implement and evaluate the impact of Attack #3 on FMBA protocol.

Attack#1

We focus on evaluating the performances of FMBA under different scenarios related to the position cheating attack. We compared the various schemes using different transmission ranges of 300 m, 650 m and 1000 m, respectively.

In our evaluation, we consider ten different scenarios:

- FMBA without attacks;
- FMBA with a false position cheating of 1.5 TR;
- FMBA with a false position cheating of 3 TR;
- FMBA with a false random position cheating between 0 and 6 TR;
- FMBA with a false position cheating of 5 TR;
- FS-MBA without attacks;
- FS-MBA with a false position cheating of 1.5 TR;
- FS-MBA with a false position cheating of 3 TR;
- FS-MBA with a false random position cheating between 0 and 6 TR; and
- FS-MBA with a false position cheating of 5 TR

To compare the various schemes under Attack #1 from a propagation delay viewpoint, we focused on parameters that have a direct delay impact. Specifically, we consider the following metrics:

- The total number of hops required to propagate the broadcast message over the whole area-of-interest;
- The total number of MAC layer slots waited before actually transmitting the broadcast message over the various hops;
- The estimated transmission range;
- The total number of MAC layer collisions (and hence time wasting retransmissions) experienced by the broadcast message.

We report the performances of these scenarios in the following charts.

Number of Slots An important question entails deriving for how long, on average, an alert message waited before its transmission to the end of the platoon. In Figure 5.21, we report the average number of slots required to broadcast a message. In the x-axis, we present the vehicle's density, and in the y-axis, we present the number of slots. We evaluate the performances of the original FMBA (without attack), and with the different cheating positions. FMBA, $d = 1.5\ TR$ (FMBA, $d = 3\ TR$) represents that the malicious node cheats about its position and declares a fake one which is $d = 1.5\ TR$ ($d = 3\ TR$) respectively, when running FMBA. FMBA, d = rand(0, 6 TR) means that the malicious node declares a random fake position between 0 and 6 TR. FMBA, $d = 5\ TR$ represents a cheating position attack of 5 TR. For all the schemes, we varied the transmission range (TR).

Figure 5.21(a) represents the average number of slots for the different protocols when using a transmission range TR= 300 m. Let us focus on FMBA algorithm. From this figure, we observe three interesting outcomes. First, with a very low vehicle density we have a reduced precision of the transmission range estimation (this would very slightly increase the number of slots). As a confirmation, when vehicle's density is 25 cars per km, the number of slots is 104, and this number decreases slightly when the density increases (50 and 100 cars per km). We can confirm this decrease since the estimation of the transmission range for all the protocols has a lower precision when vehicle density is low.

Second, with higher vehicle densities, we first have a decrease in the total number of slots due to the fact that with more vehicles at the end of the platoon, there will be a higher probability for one of them to choose a small random value within their contention windows. However, with very high vehicle densities, the number of slots needed in average to reach the end of the platoon has a big increase. This is due to the high chance for two vehicles to transmit simultaneously thus leading to time-wasting message collision and retransmissions.

For instance, when a density of 400 cars per km, the number of slots in average is approximately 178 slots when $TR = 300\ m$. Similar trends are present in the outcomes of the different protocols in Figure 5.21(b), and Figure 5.21(c).

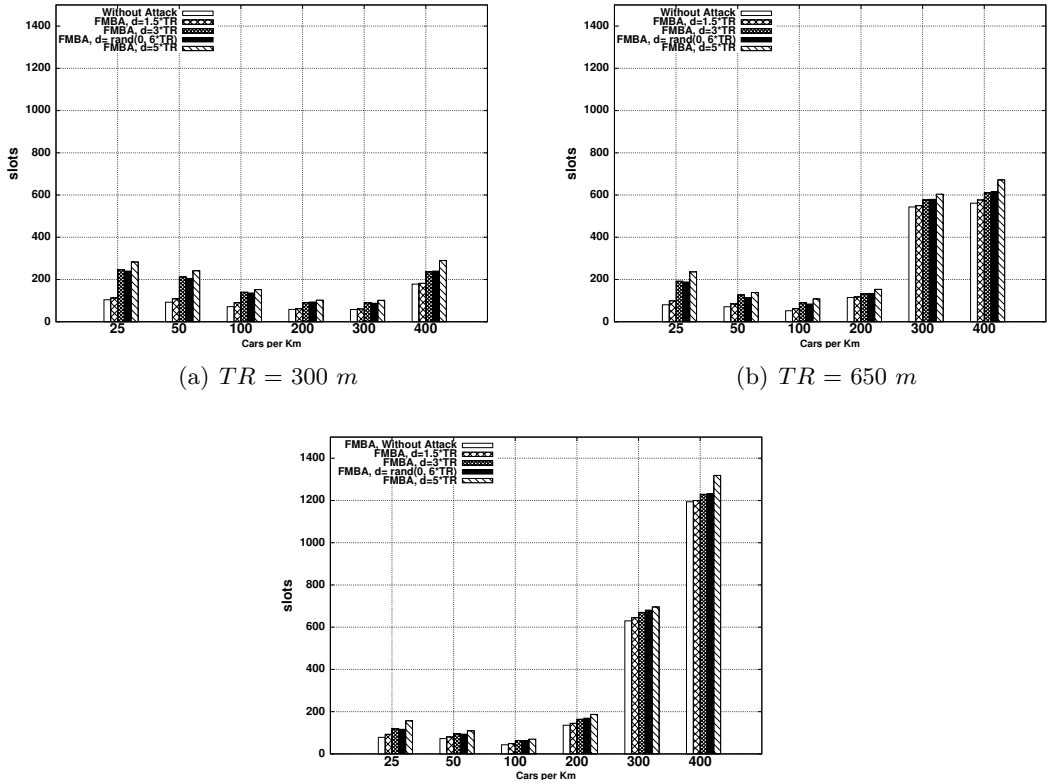
Let us focus on the impact of the four attacks on FMBA when the malicious node cheats about its position ($d = 1.5\ TR$, $d = 3\ TR$, $d = rand(0,\ 6\ TR)$, and $d = 5\ TR$). First, when an attacker runs a position cheating of $1.5\ TR$, we see the degradation of the performances of FMBA is negligible (no more than few slots). For example, in Figure 5.21(a), when the vehicle's density is 25 vehicles per km, the protocol FMBA with a position cheating of $d = 1.5\ TR$ has approximately the same number of slots as FMBA without attack (106 slots in average when vehicle density is 25 in Figure 5.21(a)). Thus, the attacker that claims a false position $d = 1.5\ TR$, has a negligible impact on increasing the average number of slots.

It is clear that the performances of FMBA degrades when an attacker declares a false position very distant from its real position. For example, FMBA with a position cheating of $d = 5\ TR$ has the worst impact among the evaluated scenarios. In this case, the malicious node increases more as compared to an attack with $d = 1.5\ TR$. As a consequence, the transmission of the alert message gets delayed more, as compared to an attack with $d = 1.5\ TR$. For instance, in Figure 5.21(c), when the vehicle's density is 25, the malicious node with a cheating position of $d = 5\ TR$ delays the transmission more than 60 slots, compared to FMBA without attack. Another interesting point, is that the performances of the attacker are almost the same when dealing with a fixed false position of $d = 3\ TR$, and $d = rand(0,\ 6\ TR)$. The explanation to this is that even the attack is in average the same, however the impact of choosing values less than $3\ TR$ decreases the number of slots.

An interesting point lies in understanding the correlation between the average number of slots for different transmission ranges. From Figure 5.21, we notice that the number of slots required to transmit a message with a transmission range $TR = 1000\ m$ is less than the protocols using a transmission range of $TR = 300\ m$ or $TR = 650\ m$. For instance, Figure 5.21(b) needs less slots to transmit the alert message since it uses a transmission range of $TR = 650m$, compared to the plotted protocols in Figure 5.21(a) using a $TR = 300\ m$. For example, when the density is 25 vehicles per km, FMBA under TR =650 m needs in average 80 slots. However, when using FMBA under $TR = 300\ m$, the message needs more than 104 slots to be propagated. From these figures, we can confirm that having a high transmission range helps in quickly transmitting an alert message.

We investigate on evaluating the average number of slots waited using FS-MBA. Let us focus on Figure 5.22. For all the different transmission ranges ($TR =\ 300\ m$, $TR =\ 650\ m$, and $TR =\ 1000\ m$), FS-MBA achieves lower transmission delays compared to FMBA under attack #1. This could be interpreted by the fact that FS-MBA uses a complete view based on local observations of vehicles, and each node is a verifier of a claim.

Let us focus on the impact of the percentage of retransmissions/collisions on the average number of hops. Figure 5.23 shows the percentage of collisions in function of the vehicle density. In Figure 5.23, the x-axis represents the vehicle density, and the y-axis represents the percentage of collisions. We see that the percentage of collisions is low for a low vehicle density. This percentage increases slightly when the density of vehicles increases. In fact, when we increase the vehicle's density too much, then this increases the possibility for two or more vehicles to randomly choose the same smallest number of slots, thus resulting in collisions and retransmissions.



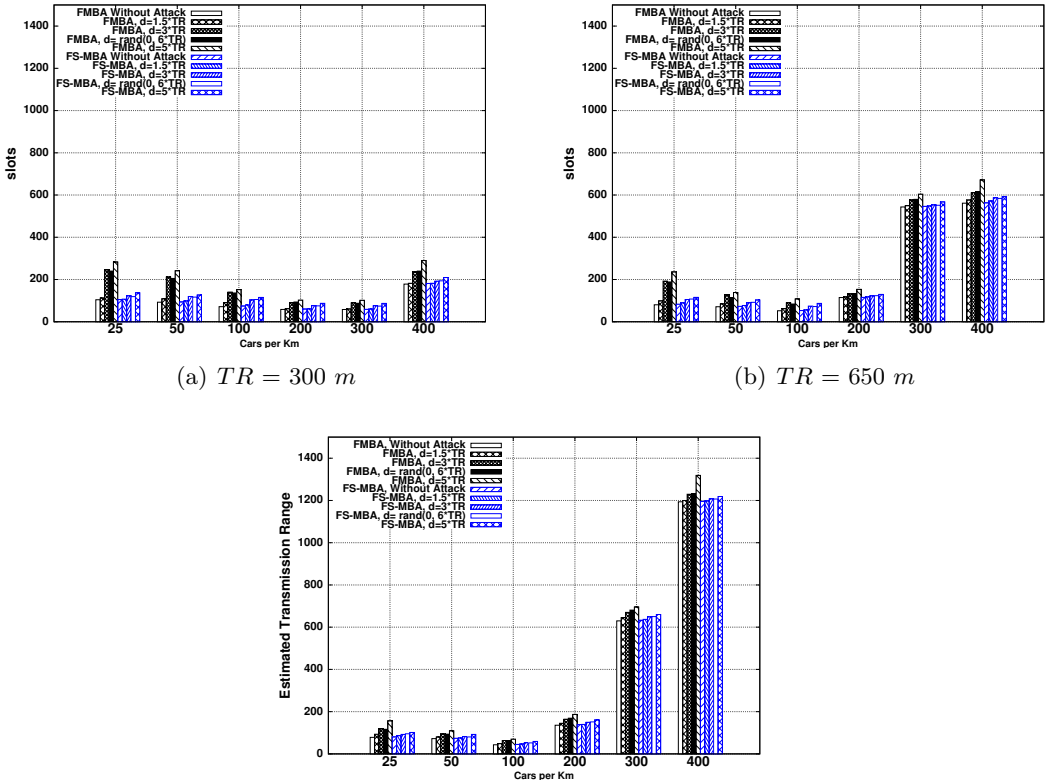
(a) $TR = 300\ m$ (b) $TR = 650\ m$

(c) $TR = 1000\ m$

Figure 5.21. – FMBA under attacks: Average Number of Slots

Another interesting point from Figure 5.23 is that fixing a vehicle density, the number of retransmissions that occured when the transmission range is 1000 m is higher, compared to the two scenarios, i.e., when $TR = 650\ m$ or $TR = 300\ m$. We see that when we increase the transmission range, and we have the same density of vehicles, there is a higher possibility that more than two nodes randomly choose the same smallest number of slots, thus generating collisions. For $TR = 1000\ m$, and with a high density of 400 vehicles per kilometer, the percentage of retransmissions is superior than 40%. However, for the same density, and having $TR = 300m$, the percentage of collisions is less than 22%. The results in Figure 5.23 confirm our findings on increasing the number of slots in Figure 5.21, for a high vehicle density. In fact, the increase of slots when vehicle density is 200 cars per km, 300 cars per km, and 400 cars per km, for all the scenarios with TR smaller than 1000 m and 650 m is due to the increase of the number of retransmissions.

Estimated Transmission Range We study the impact of the false position cheating attack, on the estimation of the transmission range of the vehicles. Using a wrong transmission range parameter (i.e., distance) has consequences on propagation delays (the number of slots). In Figure 5.24, we report the estimated transmission range for both the sender of



(a) $TR = 300\ m$

(b) $TR = 650\ m$

(c) $TR = 1000\ m$

Figure 5.22. – FMBA and FS-MBA under attacks: Average Number of Slots

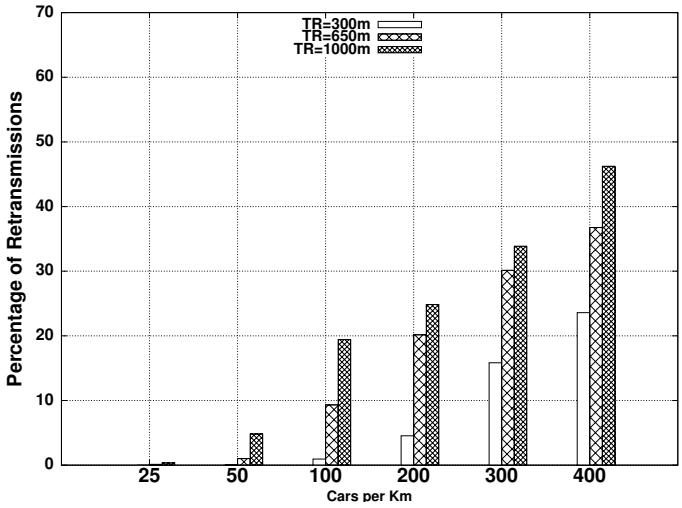


Figure 5.23. – FMBA: Percentage of Retransmissions

the message and the first forwarder of the message. In the x-axis, we represent the vehicles' density, and the y-axis is the estimated transmission range.

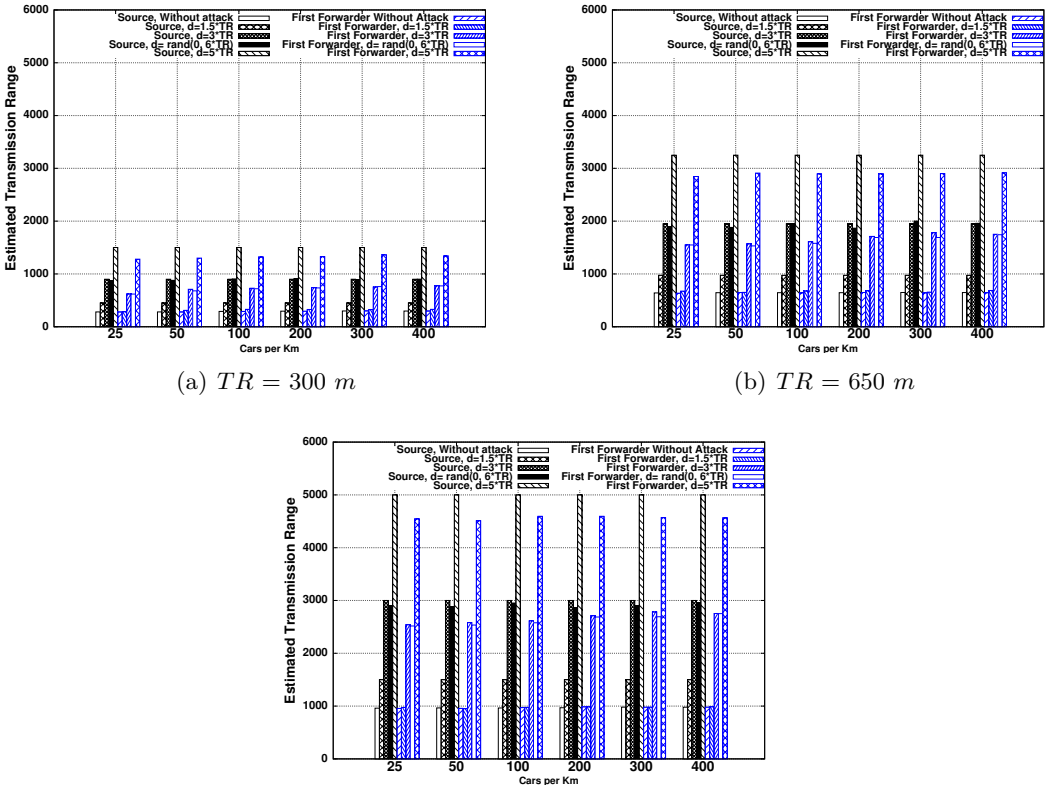
In particular, when an attack happens, the estimated transmission range is larger than the non attacked scheme. Obviously, the cheating attack with $d = 1.5\ TR$ has almost the same estimation of the transmission range as FMBA without attacks (a negligible difference). Moreover, we note that when the cheating position is higher than the real position, then the estimated transmission range is large. For instance, the estimated transmission range is large when the cheating position is 5 TR. Furthermore, the two protocols under attacks (FMBA, $d = 3\ TR$ and FMBA, $d = rand(0,\ 6\ TR)$) have approximately the same estimated transmission range.

An interesting point in Figure 5.24 is the position of the forwarder of the message, that it was affected by the attack. The difference between the sender and the forwarder of the message is approximately around 150 m, 300 m and 500 m for the protocols under transmission range $TR = 300m$, $TR = 650m$, and $TR = 1000m$ respectively. For more details, we refer the reader to the following figures (Figure 5.24(a), Figure 5.24(b), and Figure 5.24(c)). The property that emerges from these charts, is that the first forwarder is not at the end of the transmission range. This is because the estimation is made not by the first vehicle (the sender) but by the first forwarder (which should be at the end of the transmission range), but in case of the attack all vehicles will use high contention windows, thus providing all receiving nodes more or less the same probability to become the next forwarder. This means that the next forwarder will be in average at the middle of the transmission range which is the difference between the values in the sender and the first forwarder in the charts (see Figure 5.24(a), Figure 5.24(b), and Figure 5.24(c)).

In Figure 5.25, we report the estimated transmission range of the source and the first forwarder of the message, using FS-MBA under Attack #1. We want to evaluate the impact of the position cheating attack on our solution FS-MBA. For all the different cheating positions, vehicles have approximately a good estimation of the transmission range.

Declared Attacked Distance In our simulations, the malicious node cheats about its position, by claiming either a fixed position or random position. We considered a scenario where the attacker declares a random position between 0 and 6 TR. In Figure 5.26, we report the average of the declared distance by the attacker when running the random attack. It is worth noting that the average of declared distance does not depend on the vehicle density.

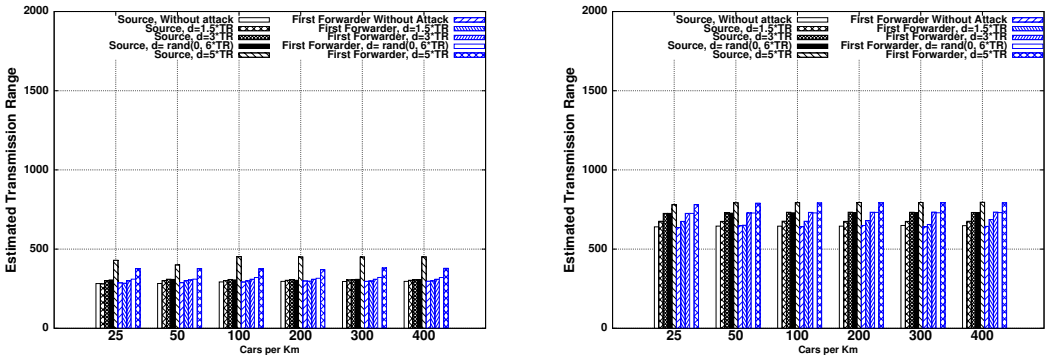
Number of Hops We study the average number of hops that a broadcast message experiences before covering the whole considered portion of road. In Figure 5.27, we report the number of hops. In particular, let us consider the scenario where the actual transmission range is 300 m (see Figure 5.27(a)). The x-axis represents the vehicle density and the y-axis shows the number of hops. It is interesting to note that the impact of the cheating position in this scenario does not affect the number of hops. The explanation to this is that we evaluated our protocols using one malicious vehicle, thus the attack affects approximately one hop. It is worth noting that, for $TR = 300\ m$, the number of hops varies slightly when



(a) $TR = 300\ m$ (b) $TR = 650\ m$

(c) $TR = 1000\ m$

Figure 5.24. – FMBA under Attack #1: Estimated transmission range



(a) $TR = 300\ m$ (b) $TR = 650\ m$

Figure 5.25. – FS-MBA under Attack #1: Estimated transmission range

augmenting the vehicle density. Let us focus on Figure 5.27(c). In this case, the number of hops increases when the density is more than 300 cars per km and higher. This happens thanks to the fact that the election of the next forwarder depends on vehicle's positions

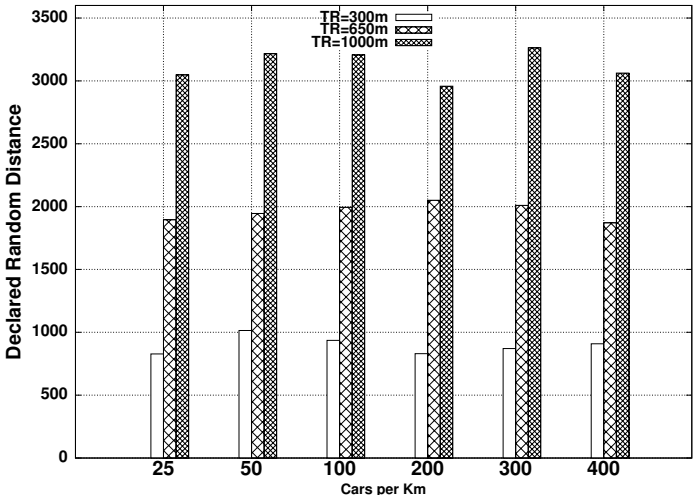


Figure 5.26. – FMBA under Attack #1: Average of declared distance by the attacker

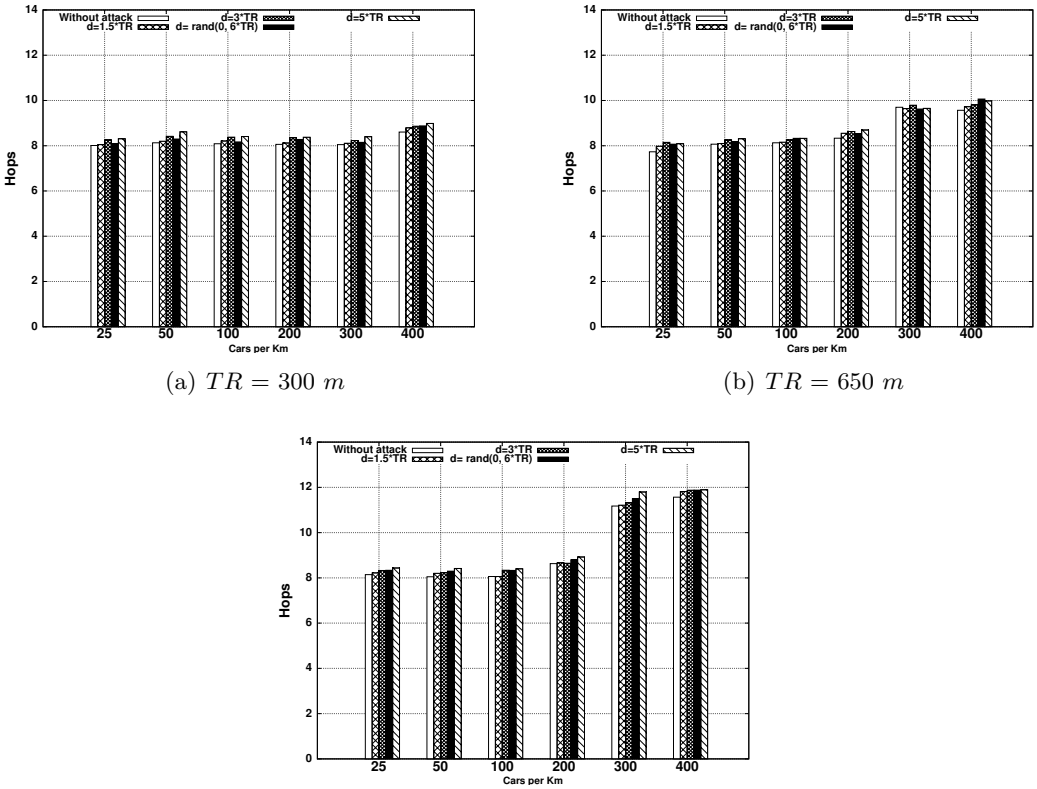
within the sender's transmission range, and also chosen randomly. We should remind that the election of the forwarder on the broadcast phase of FMBA depends on the value of the computed waiting time of each receiver. For a high vehicle density, the forwarder is not all the time at the end of the transmission range, thus the number of hops increases.

Attack #2: Replay Broadcast Message

We evaluate the performances of FMBA under Attack #2. In this attack scenario, the malicious node does not wait for the expiration of its time interval, it broadcasts the message, and keeps sending the same message. The aim of the attacker is either to block the message, and not allow the forwarding of the alert message, or to delay the message transmission.

Number of Slots We study the impact of Attack #2 on the average number of slots required before forwarding the broadcast message. In Figure 5.28, we present the vehicles density in the x-axis, and in the y-axis, we present the average number of slots.

As expected, with a high broadcast rate, the number of slots increases, since the attacker delays the transmission of the alert message, and keeps sending the same message. In fact, vehicles which are on front of the malicious node will exit the forwarding process, and vehicles which are on back of the malicious node will restart the broadcast process, thus increasing the delay of the message transmission. From the different charts (Figure 5.28(a), Figure 5.28(b), and Figure 5.28(c)), we can confirm that increasing the broadcast rate of the alert message, this would delay the transmission of the packet, or the broadcast message will be postponed, and did not achieve the last vehicle in the platoon.



(a) $TR = 300\ m$

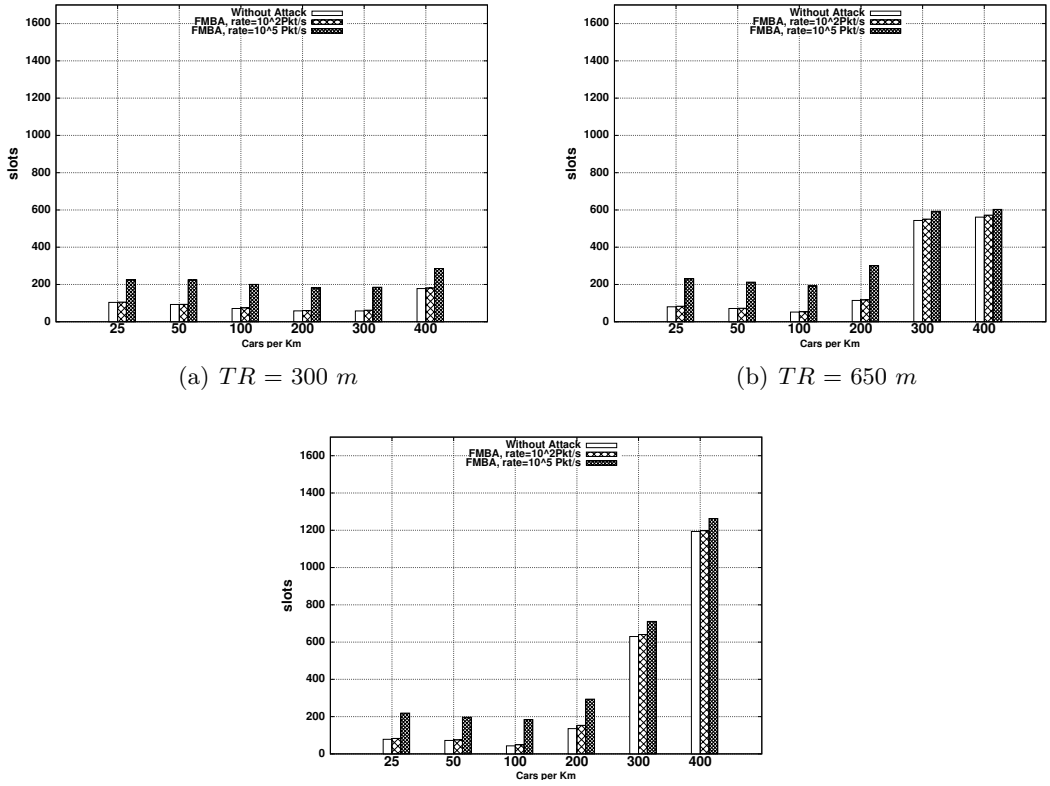
(b) $TR = 650\ m$

(c) $TR = 1000\ m$

Figure 5.27. – FMBA under Attack #1: Average Number of Hops

With varying the density of vehicles, and the rebroadcast rate, we consider that for a low rebroadcast rate, the number of slots increases slightly (see Figure 5.28). When the rebroadcast rate is 10^5 packets per second, then the average number of slots increases. From Figure 5.28(a), we see that for a low vehicle density (for example 25), the attack has more impact on increasing the number of slots compared to vehicles with high density. This could be interpreted that with more vehicles, there is a higher possibility that the message propagates to the end of the platoon.

Percentage of Message Propagation We investigate on studying the impact of Attack #2 on the percentage of message propagation. We report the percentage of message propagation using FMBA, and FMBA under Attack #2 in Figure 5.29. In the x-axis, we present the density of vehicles, and in the y-axis, we present the percentage of message propagation. We evaluate FMBA under two different broadcast rates (rate = 10^2 packets per second, and rate = 10^5 packets per second), and for different transmission ranges (TR=300 m, TR=650 m, and TR=1000 m). The percentage of message propagation increases when the rebroadcast rate decreases (see Figure 5.29). From Figure 5.29, we notice that with a high vehicle density, the percentage of message propagation for all the schemes



(a) $TR = 300\ m$

(b) $TR = 650\ m$

(c) $TR = 1000\ m$

Figure 5.28. – FMBA under Attack #2: Average Number of slots required to propagate a message to the end of the platoon

under different transmission ranges, has a high value. In fact, there is a higher probability to find a forwarder of a message when the vehicle density increases. For a low vehicle density (25 vehicles per kilometer), the message propagation is about 80%, when the rebroadcast rate is 10^5 packets per second, and TR=300 m.

Another interesting point is that, with a low transmission range ($TR = 300\ m$), the percentage of message propagation is lower compared to the protocols using a transmission range $TR = 650\ m$, or $TR = 1000\ m$. For instance, with a density of 25 vehicles per km, the percentage of message propagation is about 81% (89%), for TR=300 m (TR=1000 m) respectively.

Attack#3: Interrupting Forwarding Attack

In order to evaluate the performances of FMBA under Attack#3, we consider two scenarios. In the first scenario, the malicious node is at a random position between the

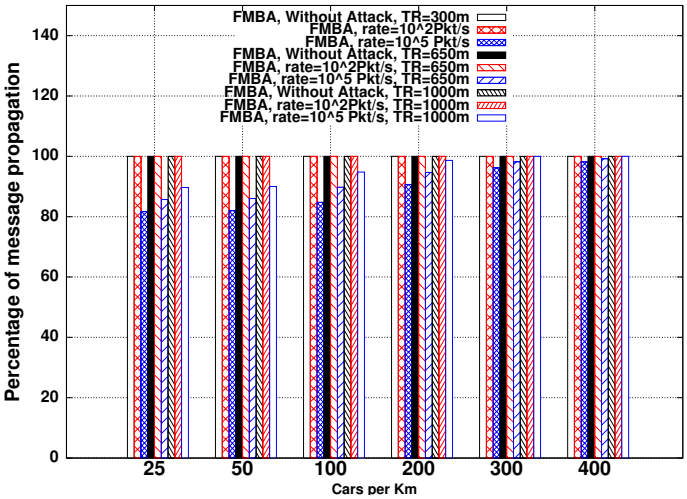


Figure 5.29. – FMBA under Attack #2: Percentage of Message Propagation to the end of the platoon

first sender of the message and the end of the transmission of this sender. In the second scenario, the malicious node is at the end of the transmission range.

In these scenarios, we evaluate the percentage of message propagation. We note that in the second scenario, the forwarder selection is the same as the original FMBA, however, the first forwarder of the message will behave maliciously and runs Attack #3.

From Figure 5.30, we notice that the percentage of message propagation is high. For instance, for a density of 25 cars per km, the percentage of message propagation is more than 80% for FMBA under different transmission ranges. In fact, when the density of vehicles increases, there is a high probability that the message reaches the end of the platoon. This could be explained by the fact that increasing the vehicle's density, there is a forwarder of the message.

In Figure 5.31, we report the scenario where the malicious node should be the first forwarder of the message. In order to implement this scenario, we assume that the malicious node is at the end of the transmission range. In this case, we notice that the attacker did block the transmission of the alert message with different percentages. In Figure 5.31, the message propagation of FMBA under Attack#3 is approximately 23% with a TR =300 m, and with a density of 25 cars per km. This confirms our findings that when an attacker is at the end of the transmission range of the sender, then it has a higher possibility to block the transmission of the packet.

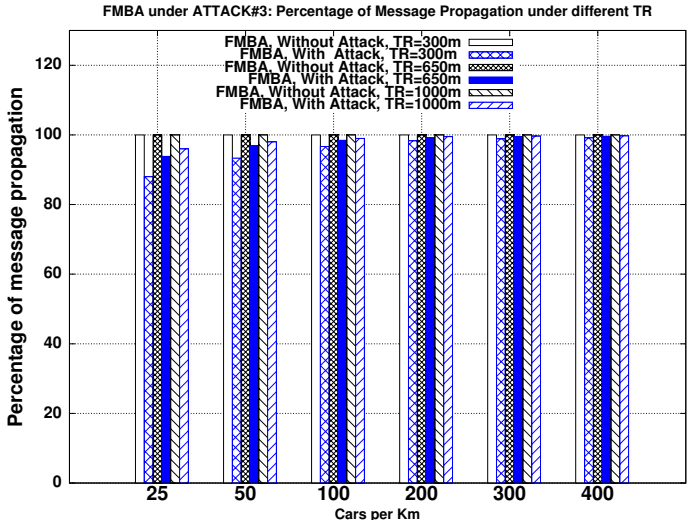


Figure 5.30. – FMBA under Attack#3 with a Random Position of Malicious node: Percentage of Message Propagation

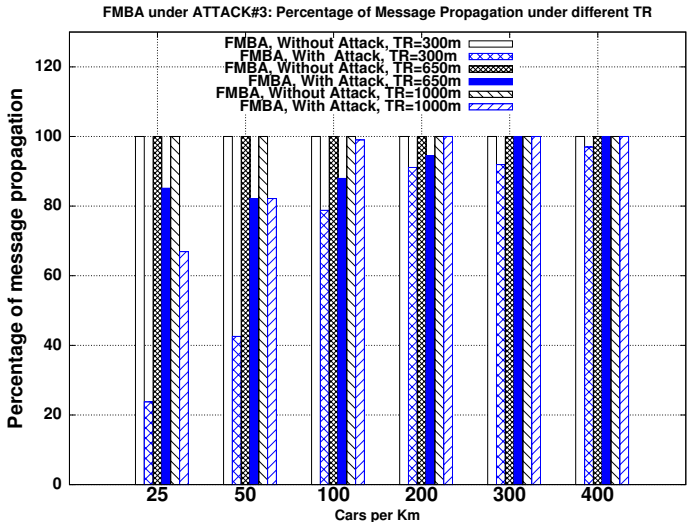


Figure 5.31. – FMBA under Attack #3 with Malicious Node at the End of the Transmission Range of the First Sender: Percentage of Message Propagation

5.11 Summary

The main goal of Inter Vehicular Communications (IVC) consists in increasing people's safety by exchanging warning messages between vehicles. This chapter scratches the surface of what is promising to be a new and fertile area of research in IVC security. We have elaborated on security issues in IVC considering a general class of applications based on multi-hop broadcast; yet, without loss of generality, we have chosen a representative case

study for this class, FMBA, to concretely discuss issues and possible solutions. In this context, we have provided an overview of the different attacks and security weaknesses, also proposing possible countermeasures.

CHAPTER 6

Conclusion

The number of areas and problems to which Wireless Sensor Networks and vehicular communications are applied continuously, grow while known and unknown threats affect these technologies. Researchers are called to address the design of efficient protocols that are secure against possible attacks. This chapter summarizes the contributions of the research presented in this dissertation over the previous state of the art and suggests directions for future work.

6.1 Contributions

This dissertation studies communication security in WSNs with respect to three important aspects: broadcast/multicast security, attacks on broadcast authentication, secure group communications, and secure and fast multi hop broadcast applications.

- Broadcast Authentication. We identified the problem of broadcast authentication in WSNs and pointed out a serious security vulnerability inherent to the symmetric-key based μTESLA-like schemes [5, 13, 49, 119, 121]. We then proposed several key disclosure based schemes to address the proposed problem with resistance to DoS attack, and with minimized computational and communication costs. We achieved our goal by integrating several cryptographic building blocks, such as the Bloom filter, and the staggered authentication in an innovative manner. We also analyzed both the performance and security resilience of the proposed schemes. Our research on this topic appears in [200–205].

- Delayed Authentication Compromise. To address the delayed authentication compromise attack in broadcast authentication schemes, we first presented two state of the art protocols that are inherently demanded by WSNs. We then demonstrate that these protocols are vulnerable to two kind of attacks: drop command attack, and switch command attack. Our proposal advances the current state-of-the art by enabling immediate message authentication with low overhead. The efficiency and se-

curity properties of MASS were justified through analysis. Our research in this topic appears in [206].

- Secure Group Communications. Although many works in [9, 11] discussed source authentication in wireless sensor networks, and especially group key for secure group communications, the problem is still challenging because we have to manage the trade-off between acceptable levels of security and, conserving scarce resources, in particular energy needed for network operations. Our proposal in this topic appears in [207,208].

- Secure Vehicular Communications. The main goal of Inter Vehicular Communications (IVC) consists in increasing people's safety by exchanging warning messages between vehicles. We have elaborated on security issues in IVC considering a general class of applications based on multi-hop broadcast; yet, without loss of generality, we have chosen a representative case study for this class, FMBA, to concretely discuss issues and possible solutions. We have provided an overview of the different attacks and security weaknesses, also proposing possible countermeasures. Our proposal on this topic appears in [209–211].

6.2 Future Direction

In this section, we briefly mention areas of future work based on our thesis. We outline six directions as follows.

- Secure Vehicular Communications. In this dissertation, we focused on securing information for fast broadcast applications used to alert messages. We will extend our schemes to secure more applications related to vehicular communications. Moreover, we will extend our work using different malicious nodes under different positions. Furthermore, a good direction is to validate formally our proposals, and evaluate the behavior of the malicious node.

- Broadcast Authentication. We will extend our proposal MASS using a thorough assessment with real devices, to confirm the efficiency of our solution. Moreover, we will extend the behavior of the attackers in memory DoS attacks to evaluate the performances of Staggered multi-level-μTESLA, and Bloom Filter multi-level-μTESLA. Furthermore, we will focus on the formal verification of our schemes.

- Secure Data Aggregation. In many applications, the raw information sensed by individual sensors should be aggregated for the purpose of reducing the communication cost and energy expenditure in data collection. In this dissertation, we did not take in consideration the secure data aggregation, however in order to preserve the privacy and to know the identification of the intermediate nodes in the path, a mechanism of authentication should be established. We will focus on building a new mechanism that deal about this issue.

- Privacy-aware Security Services. Data and network communication privacy can be a big concern in many applications. As WSNs and vehicular networks are envisioned to become more and more pervasive, privacy-aware security services should be further developed. Privacy must be achieved in the sense that device/user-related information should be protected.
- We aim to design a reference security architecture for vehicular safety applications, with a set of coherent and complete assumptions.
- Finally, we aim to investigate specific security problems in heterogeneous WSNs.

Bibliography

[1] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, jul 1993.

[2] X. Cao, W. Kou, L. Dang, and B. Zhao, "IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks," *Journal of Computer Communications*, vol. 31, pp. 659–667, 2008.

[3] X. Fan and G. Gong, "Accelerating signature-based broadcast authentication for wireless sensor networks," *Journal of Ad Hoc networks*, vol. 10, pp. 723–736, 2012.

[4] P. Ning, D. Liu, and W. Du, "Mitigating DoS Attacks against Broadcast Authentication in Sensor Networks," MSP.

[5] D. Liu and P. Ning, "Multi-Level µTESLA: Broadcast Authentication for Distributed Sensor Networks," *ACM Transactions in Embedded Computing Systems (TECS)*, vol. 3, pp. 800–836, Nov. 2004.

[6] K. Xiong, R. Wang, W. Du, and P. Ning, "Containing bogus packet insertion attacks for broadcast authentication in sensor networks," *ACM Transactions on Sensor Networks Journal (TOSN)*, vol. 8, jul 2012.

[7] Q. Dong, D. Liu, and P. Ning, "Providing DoS resistance for signature-based broadcast authentication in sensor networks," *ACM Transactions on Embedded Computing Systems Journal (TECS)*, vol. 12, pp. 1–12, mar 2013.

[8] R. D. Pietro, C. Soriente, A. Spognardi, and G. Tsudik, "Collaborative authentication in unattended WSNs," Proceedings of the second ACM conference on Wireless network security WiSec, pp. 237–244, 2009.

[9] T. Kavitha and D. Sridharan, "Security Vulnerabilities In Wireless Sensor Networks: A Survey," *Journal of Information Assurance and Security*, vol. 5, pp. 31–44, 2010.

[10] I. Krontiris and T. Dimitriou, "Scatter – secure code authentication for efficient reprogramming in wireless sensor networks," *International Journal of Sensor Networks*, vol. 10, pp. 14–24, 2011.

[11] O. D. Mohatara, A. F. Sabatera, and J. M. Sierrab, "A light-weight authentication scheme for wireless sensor networks," *Journal of Ad hoc networks*, vol. 9, pp. 727–735, July 2011.

[12] A. K. Das, P. Sharma, S. Chatterjee, and J. K. Sing, "A dynamic password-based user authentication scheme for hierarchical wireless sensor networks," *Journal of Network and Computer Applications*, vol. 35, pp. 1646–1656, sep 2012.

[13] A. Perrig, R. Szewczyk, D. Culler, V. Wen, and D. Tygar, "Spins: Security protocols for sensor networks," in *Proc. Seventh Annual International Conference on Mobile Computing and Networks*, 2002.

[14] B. Wu, J. Wu, E. B. Fernandez, M. Ilyas, and S. Magliveras, "Secure and efficient key management in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 30, pp. 937–954, august 2007.

[15] M. Chorzempa, J.-M. Park, and M. Eltoweissy, "Key management for long-lived sensor networks in hostile environments," *Computer Communications*, vol. 30, no. 9, pp. 1964–1979, 2007.

[16] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, pp. 521–534, Sept. 2002.

[17] K. Ren, S. Yu, W. Lou, and Y. Zhang, "Multi-user broadcast in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 4554–4564, oct 2009.

[18] R. Di Pietro, F. Martinelli, and N. V. Verde, "Broadcast Authentication for Resource Constrained Devices: A Major Pitfall and Some Solutions," in *IEEE SRDS*, pp. 213–218, oct 2012.

[19] P. Schaller, S. Čapkun, and D. Basin, "BAP: Broadcast Authentication Using Cryptographic Puzzles," *Applied Cryptography and Network Security*, vol. 4521, pp. 401–419, 2007.

[20] Y. Kim, A. Perrig, and G. Tsudik, "Group Key Agreement Efficient in Communication," *IEEE Transactions on Computers*, vol. 53, 2004.

[21] J.-H. Son, J.-S. Lee, and S.-W. Seo, "Topological Key Hierarchy for Energy-Efficient Group Key Management in Wireless Sensor Networks," *Journal of Wireless Personal Communications*, vol. 52, no. 2, pp. 359–382, 2010.

[22] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, pp. 60–96, feb 2004.

[23] Y. Amir, Y. Kim, C. Nita-rotaru, J. Schultz, J. Stanton, and G. Tsudik, "Secure group communication using robust contributory key agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, pp. 468–480, 2004.

[24] A. M. Eskicioglu and M. R. Eskicioglu, "Multicast Security Using Key Graphs and Secret Sharing," in *Proceedings of the Joint International Conference on Wireless LANs and Home Networks (ICWLHN 2002) and Networking (ICN 2002)*, pp. 228–241, 2002.

[25] Y. Wang and B. Ramamurthy, "Group Rekeying Schemes for Secure Group Communication in Wireless Sensor Networks," in *Proc. IEEE International Conference on Communications*, pp. 3419–3424, 2007.

[26] M. Tubaishat, J. Yin, B. Panja, and S. Madria, "A secure hierarchical model for sensor network," vol. 33, pp. 7–13, 2004.

[27] M. L. Sichitiu and M. Kihl, "Inter-vehicle communication systems: a survey," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 2, pp. 88–105, 2008.

[28] C. Wu and Y. Liu, "Queuing network Modeling of Driver Workload and Performance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 528–537, 2007.

[29] P. Papadimitratos, L. Buttyan, T. Holcze, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure vehicular communication systems: design and architectures," *IEEE Communications Magazine*, vol. 46, pp. 100–109, Nov. 2008.

[30] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, and M. Gerla, "How Do You Quickly Choreograph Inter-Vehicular Communications? A Fast Vehicle-to-Vehicle Multi-Hop Broadcast Algorithm, Explained," in *IEEE CCNC*, jan 2007.

[31] C. E. Palazzi, M. Roccetti, and S. Ferretti, "An Intervehicular Communication Architecture for Safety and Entertainment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, pp. 90–99, Mar. 2010.

[32] *http://ec.europa.eu/transport/roadsafety/specialist/statistics/.*

[33] "http://www.who.int/en/."

[34] S. Biswas, R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *IEEE Communications Magazine*, vol. 44, pp. 74–82, jan 2006.

[35] A. Weimerskirch, J. J. Haas, Y.-C. Hu, and K. P. Laberteaux, *VANET: Vehicular Applications and Inter-Networking Technologies Kenneth P Laberteaux*, ch. 9 :Data Security in Vehicular Communication Networks. Chichester, UK: John Wiley & Sons,, Nov. 2009.

[36] L. Zhang, Q. Wu, A. Solanas, and J. Domingo-Ferrer, "A Scalable Robust Authentication Protocol for Secure Vehicular Communications," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 1606–1617, May 2010.

[37] B. Mishra, P. Nayak, S. Behera, and D. Jena, "Security in vehicular adhoc networks: a survey," in *ICCCS*, feb 2011.

[38] G. Calandriello, P. Papadimitratos, J. P. Hubaux, and A. Lioy, "On the Performance of Secure Vehicular Communication Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, pp. 898–912, 2011.

[39] M. Raya and J.-P. Hubaux, "The security of vehicular ad hoc networks," in *ACM SASN*, nov 2005.

[40] A. Aijaz, B. Bochow, F. Dötzer, A. Festag, M. Gerlach, R. Kroh, and T. Leinmüller, "Attacks on Inter Vehicle Communication Systems - an Analysis," in *Workshop on Intelligent Transportation*, Mar. 2006.

[41] Q. Wu, J. Domingo-Ferrer, and U. Gonzalez-Nicolas, "Balanced Trustworthiness, Safety and Privacy in Vehicle-to-Vehicle Communications," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 559–573, Mar. 2010.

[42] H. Hartenstein and K. Laberteaux, "A tutorial survey on vehicular ad hoc networks ," *IEEE Communications Magazine*, vol. 46, pp. 164–171, jun 2008.

[43] J. Guo, J. Baugh, and S. Wang, "A Group Signature Based Secure and Privacy-Preserving Vehicular Communication Framework ," 2007 Mobile Networking for Vehicular Environments , pp. 103–108, may 2007.

[44] C. Zhang, X. Lin, R. Lu, and P.-H. Ho, " An Efficient Message Authentication Scheme for Vehicular Communications ," *IEEE Transactions on Vehicular Technology*, vol. 5704, pp. 3357–3368, nov 2008.

[45] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 10, pp. 393–422, 2002.

[46] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, "Real-time commu- nication and coordination in embedded sensor networks ," pp. 1002–1022, 2003.

[47] Y. Jiang, M. Shi, X. Shen, and C. Lin, "BAT: A robust signature scheme for vehicular networks using Binary Authentication Tree," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 1974–1983, apr 2009.

[48] Y. Zhou, Y. Fang, and Y. Zhang, "Securing wireless sensor networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 10, pp. 6–28, 2008.

[49] A. Perrig, R. Canetti, B. Briscoe, J. Tygar, and D. X. Song, "Tesla: multicast source authentication," in *Internet Draft, Internet Engineering Task Force*, 2000.

[50] J. Elson and K. Romer, "Wireless sensor networks: a new regime for time synchronization," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 149–154, jan 2003.

[51] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronisation protocol," in *2nd international conference on Embedded networked sensor systems*, pp. 39–49, Nov. 2004.

[52] *http://www.willow.co.uk/html/telosbmoteplatform.html.*

[53] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, " Protocols for self-organization of a wireless sensor network ," *IEEE Personal Communications*, vol. 7, pp. 16–27, oct 2000.

[54] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks ," Proceedings of the 9th ACM conference on Computer and Communications Security , pp. 41–47, 2002.

[55] D. Liu and N. Peng, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," vol. 3 of *Proc. of NDSS*, pp. 263–276, feb 2003.

[56] A. D. Wood and J. A. Stankovic, " Denial of service in sensor networks ," *Computer*, vol. 35, pp. 54–62, oct 2002.

[57] "http://www.zigbee.org/."

[58] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, (New York, NY, USA), pp. 259–268, ACM, 2004.

[59] J. R. Douceur, "The sybil attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, (London, UK), pp. 251–260, Springer-Verlag, 2002.

[60] C. Bekara, M. Laurent-maknavicius, and K. Bekara, "SAPC: A Secure Aggregation Protocol for Cluster-Based Wireless Sensor Networks ," Mobile Ad-Hoc and Sensor Networks MSN, pp. 784–798, 2007.

[61] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG : a Tiny AGgregation Service for Ad-Hoc Sensor Networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, 2002.

[62] S. Jarecki, J. Kim, and G. Tsudik, "Flexible Robust Group Key Agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 879–886, may 2011.

[63] R. Di Pietro, D. Ma, C. Soriente, and G. Tsudik, "Self-healing in unattended wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, nov 2012.

[64] Y. Jianga, C. Lin, M. Sh, and X. S. Shen, "Self-healing group key distribution with time-limited node revocation for wireless sensor networks," *Ad Hoc Networks*, vol. 5, pp. 14–23, jan 2007.

[65] A. Shikfa, M. Önen, and R. Molva, "Local key management in opportunistic networks," *International Journal of Communication Networks and Distributed Systems (IJCNDS)*, vol. 9, no. 1, 2012.

[66] X. Du, Y. Xiao, M. Guizani, and H.-H. Che, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Networks*, vol. 5, pp. 24–34, jan 2007.

[67] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, pp. 228–258, may 2005.

[68] G. YANG, C. ming RONG, C. VEIGNER, J. tao WANG, and H. bing CHENG, "Identity-based key agreement and encryption for wireless sensor networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 13, pp. 54–60, dec 2006.

[69] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing sensor networks with location-based keys ," IEEE WCNC , pp. 1909–1914, mar 2005.

[70] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, nov 1976.

[71] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks , SASN'04, pp. 59–64, 2004.

[72] N. Gura, A. Patel, W. Arvinderpal, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," *Cryptographic Hardware and Embedded Systems*, pp. 119–132, 2004.

[73] D. J. Malan, M. Welsh, and M. D. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography," pp. 71–80, 2004.

[74] A. Seshadri, A. Perrig, L. V. Doorn, and P. Kholsa, "SWATT: softWare-based attestation for embedded devices," In Proceedings of the 2004 IEEE Symposium on Security and Privacy , pp. 272–282, may 2004.

[75] M. A. S. Jr, P. S. Barreto, C. B. Margi, and T. C. Carvalho, "A survey on key management mechanisms for distributed Wireless Sensor Networks," *Journal of Computer Networks*, vol. 54, pp. 2591–2612, Oct. 2010.

[76] J. Zhang and V. Varadharajan, "Wireless sensor network keymanagement survey and taxonomy," *Journal of Network and Computer Applications*, vol. 33, pp. 63–75, Mar. 2010.

[77] T. Aura, P. Nikander, and J. Leiwo, "DOS-resistant authentication with client puzzles," in *Revised Papers from the 8th International Workshop on Security Protocols*, (London, UK), pp. 170–177, Springer, 2001.

[78] R. Merkle, "A digital signature based on a conventional encryption function," in *CRYPTO*, pp. 369–378, 1988.

[79] T. Leinmüller, C. Maihöfer, E. Schoch, and F. Kargl, "Improved security in geographic ad hoc routing through autonomous position verification," in *VANET workshop*, 2006.

[80] J.-P. Hubaux, S. Capkun, and J. Luo, "The security and privacy of smart vehicles," *IEEE Security & Privacy*, vol. 2, pp. 49–55, May 2004.

[81] N. Sastry, S. Umesh, and D. Wagner, "Secure verification of location claims," in *WiSE* , 2003.

[82] G. Yan, S. Olariu, and M. C. Weigle, "Providing vanet security through active position detection," *Journal of Computer Communications*, vol. 31, pp. 2883–2897, July 2008.

[83] A. Vora and M. Nesterenko, "Secure Location Verification Using Radio Broadcast," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, Oct. 2006.

[84] Z. Ren, W. Li, and Q. Yang, "Location Verification for VANETs Routing," in *IEEE WIMOB*, Oct. 2009.

[85] T. Leinmüller, E. Schoch, and F. Kargl, "Position Verification Approaches for Vehicular Ad Hoc Networks," *IEEE Wireless Communication Magazine*, vol. 13, pp. 16–21, Oct. 2006.

[86] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, vol. 1, pp. 113——127, Elsevier, Sept. 2003.

[87] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, vol. 31, pp. 1–13, Citeseer, 2002.

[88] F. Nait-Abdessalem, "Detecting and avoiding wormhole attacks in wireless ad hoc networks," *IEEE Communications Magazine*, pp. 127–133, apr 2008.

[89] W. Wang and B. Bhargava, "Visualization of wormholes in sensor networks," Proceedings of the 3rd ACM workshop on Wireless security (WiSe'04), pp. 51–60, 2004.

[90] Q. Zhang, P. Wang, D. S. Reevers, and P. Ning, "Defending against sybil attacks in sensor networks," 25 th International Conference on Distributed Computing Systems Workshops, pp. 185–191, jun 2005.

[91] Z. Yan, P. Zhang, and T. Virtanen, "Trust Evaluation based Security Solution in Ad hoc Networks," In Proceedings of the Seventh Nordic Workshop on Security IT Systems (NordSec 2003) , 2003.

[92] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy, "Location-centric isolation of misbehavior and trust routing in energy-constrained sensor networks ," In IEEE Workshop on Energy-Efficient Wireless Communications (EWCN), in conjunction with IEEE IPCCC'04, pp. 14–17, 2004.

[93] Z. Liang and W. Shi, "Enforcing cooperative resource sharing in untrusted peer-to-peer environment," *ACM Journal of Mobile Networks and Applications (MONET) special*, vol. 10, 2005.

[94] Z. Liang and W. Shi, "Pet: A personalized trust model with reputation and risk evalua- tion for p2p resource sharing ," In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005.

[95] Y. Lu, K. Li, W. Zhou, and P. Li, "Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures," *Elsevier Journal of Network and Computer Applications*, vol. 35, pp. 867–880, may 2012.

[96] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, may 2008.

[97] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heam, J. Mantovani, S. Modersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turu, L. Vigano, and L. Vigneron, "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications," in *Proceeding of the 17th International Conference on Computer Aided Verification (CAV'05)*, vol. 3576 of *LNCS*, Springer, 2005.

[98] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in *First ACM Conference on Embedded Networked Sensor Systems Sensys*, Nov. 2003.

[99] E. Perla, R. S. Carbajo, M. Huggard, and C. M.Goldrick, "PowerTossim Z: realistic energy modeling for wireless sensor network envrionments," in *3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pp. 35–42, Oct. 2008.

[100] M. Mitzenmacher, "Compressed Bloom Filters," *IEEE/ACM Transactions on Networks*, vol. 10, pp. 613–620, Oct. 2000.

[101] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, pp. 770–772, nov 1981.

[102] N. Haller, "The S/KEY One-Time Password System," In Proceedings of the Internet Society Symposium on Network and Distributed Systems, pp. 151–157, 1994.

[103] M. Jakobsson, "Fractal hash sequence representation and traversal ," IEEE International Symposium on Information Theory, p. 437, 2002.

[104] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 162–175, ACM, 2004.

[105] S. Y. Chang, Y. H. Lin, H. M. Sun, and M. E. Wu, "Practical RSA signature scheme based on periodical rekeying for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 8, Mar. 2012.

[106] F. Hess, "Efficient identity based signature schemes based on pairings," in *Proc.SAC,St.John's*, (Newfoundland,Canada), 2002.

[107] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An Efficient Signature-Based Scheme for Securing Network Coding Against Pollution Attacks," in *Proc. the 27th Conference on Computer Communications INFOCOMM*, pp. 1409–1417, 2008.

[108] C. Zhang, R. Lu, X. Lin, P. Ho, and X. Shen, "An Efficient Identity-Based Batch Verification Scheme for Vehicular Sensor Networks," in *Proc. the 27th Conference on Computer Communications INFOCOMM*, pp. 246–250, 2008.

[109] G. Gaubatz, J.-P. Kaps, E. Ozturk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks ," pp. 146–150, 2005.

[110] A. Liu and P. Ning, " TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks ," IPSN, pp. 245–256, apr 2008.

[111] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public key cryptography on small wireless devices," in *Proc. Percom Conf.*, 2008.

[112] T. Cheneau, A. Boudguiga, and M. Laurent, "Significantly improved performances of the cryptographically generated addresses thanks to ECC and GPGPU," *Computers & Security*, vol. 29, pp. 419–431, jan 2010.

[113] G. M. Bertoni, L. Chen, P. Fragneto, K. A. Harrison, and G. Pelosi, "Computing tate pairing on smartcards," 2005.

[114] V. Miller, "Short programs for functions on curves," 1986.

[115] R. W. Zhu, G. Yang, and D. S. Wong, "An Efficient Identity-based key exchange protocol with KGS forward secrecy for low-power devices," *Theoretical Computer Science*, vol. 378, pp. 198–207, 2007.

[116] D. Johnson and A. M. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[117] T. Kwon and J. Hong, "Secure and Efficient Broadcast Authentication in Wireless Sensor Networks," *IEEE Transactions on Computers*, vol. 59, pp. 1120–1133, Aug. 2010.

[118] X. Zhang, H. Heys, and C. Li, "Energy efficiency of symmetric key cryptographic algorithms in wireless sensor networks," in *Communications (QBSC), 2010 25th Biennial Symposium on*, pp. 168–172, IEEE, 2010.

[119] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical Broadcast Authentication in Sensor Networks," The second Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services MobiQuitous, pp. 118–129, July 2005.

[120] R. C. Merkle, "Protocols for Public Key Cryptosystems," IEEE Symposium on Security and Privacy, 1980.

[121] Y.-S. Chen, I.-L. Lin, C.-L. Lei, and Y.-H. Liao, "Broadcast Authentication in Sensor networks Using Compressed Bloom Filters," *Distributed Computing in Sensor Systems*, vol. 5067, 2008.

[122] Y. Zhou and Y. Fang, "BABRA: Batch-based Broadcast Authentication in Wireless Sensor Networks," GlobeCom, Nov. 2006.

[123] T. Wu, Y. Cui, B. Kusy, A. Ledeczi, J. Sallai, N. Skirvin, J. Werner, and Y. Xue, "A fast and Efficient Source Authentication Solution for Broadcasting in Wireless Sensor Networks," IFIP, IEEE Proceeding of New Technologies, Mobility and Security,NTMS, May 2007.

[124] A. Perrig, R. Canetti, D. Song, and J.D.Tygar, "Efficient and Secure Source Authentication for Multicast," Network and Distributed System Security Symposium, NDSS, 2001.

[125] Q. Li and W. Trappe, "Reducing Delay and enhancing DoS resistance attacks in multicast authentication through multigrade security," *ACM Transactions in Embedded Computing Systems (TECS)*, vol. 1, pp. 190–204, June 2006.

[126] Y. Matsumoto, H. Hazeyama, and Y. Kadobayashi, "Adaptive Bloom Filter: A space Efficient Counting Algorithm for Unpredictable Network Traffic," *Journal EICE - Transactions on Information and Systems*, vol. E91-D, May 2008.

[127] S. Gritzalis, D. Spinellis, and P. Georgiadis, "Security Protocols over Open Networks and Distributed Systems: Formal Methods for their analysis, design, and verification," *Journal of Computer Communications*, vol. 22, pp. 697–709, May 1999.

[128] D. Dolev and A. C. Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, mar 1983.

[129] L. Bozga, Y. Lakhnech, and M. Périn, "Hermes, a Tool verifying secrecy properties of unbounded security protocols," 15th International Conference on Computer Aided Verification (CAV'03), July 2003.

[130] M. Burrows, M. Abadi, and oger Needham, "A Logic of Authentication," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, pp. 18–36, Feb. 1990.

[131] E. Clarke, E. Grumberg, and D. A. Peled, *Model checking*. 2000.

[132] L. Tobarra, D. Cazorla, F. Cuartero, G. Diaz, and E. Cambronero, "Model Checking Wireless Sensor Networks Security Protocols: TinySec+LEAP+TinyPK," First IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07), pp. 95–106, 2007.

[133] "Crossbow technology inc., www.xbow.com,"

[134] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical broadcast authentication in sensor networks," in *MOBIQUITOUS*, pp. 118–132, 2005.

[135] A. Norouzi, A. Abhari, and T. Yang, "Enhancing broadcast authentication in sensor networks ," in *14th Communications and Networking Symposium* , pp. 125–132, 2011.

[136] L. Lamport, "Constructing digital signatures from a one-way function," in *SRI International Computer Science Laboratory*, no. October, 1979.

[137] R. Meier and R. Wattenhofer, "ALPS: Authenticating Live Peer-to-Peer Streams," in *Proceedings of the 27th Annual IEEE International Symposium on Reliable Distributed Systems (SRDS), Naples, Italy*, 2008.

[138] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol," in *Proceedings of the 8th ACM conference on Computer and Communications Security*, pp. 28–37, ACM, 2001.

[139] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Information Security and Privacy*, pp. 1–47, Springer, 2002.

[140] S.-M. Chang, S. Shieh, W. W. Lin, and C.-M. Hsieh, "An efficient broadcast authentication scheme in wireless sensor networks," *Proceedings of the 2006 ACM Symposium on Information, computer and communications security - ASIACCS '06*, p. 311, 2006.

[141] NIST, "FIPS 180-2: Secure hash standard," tech. rep., 2002.

[142] M. Wena, Y. F. Zhengb, W. J. Ye, K. F. Chenb, and W. D. Qiub, "A key management protocol with robust continuity for sensor networks," *Journal of Computer Standards and Interfaces*, vol. 31, pp. 642–647, 2009.

[143] S. L. Keoh, E. Lupu, and M. Sloman, "Securing body sensor networks: Sensor association and key management," in *Proc. IEEE International Conference on Pervasive Computing and Communications PerCom*, pp. 1–6, Mar. 2009.

[144] Y. Zhang, W. Lou, W. Liu, and Y. Fang, "A Secure Incentive Protocol for Mobile Ad Hoc Networks," *Wireless Networks*, vol. 13, pp. 569–582, Oct. 2007.

[145] J. Pan, L. Cai, X. S. Shen, and J. W. Mark, "Identity-Based Secure Collaboration in Wireless Ad Hoc Networks," *Computer Networks (Elsevier)*, vol. 51, no. 3, pp. 853–865, 2007.

[146] M. M. E. A. Mahmoud and S. Shen, "Credit-Based Mechanism Protecting Multi-Hop Wireless Networks from Rational and Irrational Packet Drop," in *IEEE Global Telecommunications Conference (GLOBECOM 2010), 2010*, pp. 1–5, Dec. 2010.

[147] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol," in *Proc. of ACM MobiHoc 2002*, pp. 226–236, ACM, June 2002.

[148] P. Mukerjee and S. Sen, "Comparing Reputation Schemes for detecting malicious nodes in sensor networks," *The computer Journal*, vol. 54, no. 3, pp. 482–489, 2011.

[149] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik, "On the performance of group key agreement protocols," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, pp. 457–488, aug 2004.

[150] Z. Yu and Y. Guan, "A robust group-based key management scheme for wireless sensor networks ," IEEE Wireless Communications and Networking Conference, pp. 1915–1920, 2005.

[151] W. Zhang and G. Cao, "Group rekeying for filtering false data in sensor networks: a predistribution and local collaboration-based approach," in *IEEE INFOCOM*, 2005.

[152] J. Deng, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM conference on Computer and communication security - CCS '03*, vol. V, (New York, New York, USA), p. 42, The University of North Carolina at Greensboro, 2005.

[153] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Internet Draft, Internet Engineering Task Force*, pp. 93–104, 2000.

[154] L. Li, J. Song, F.-Y. Wang, W. Niehsen, and N. Zheng, "New Developments and Research Trends for Intelligent Vehicles," *IEEE Intelligent Systems*, vol. 20, no. 4, pp. 10–14, 2005.

[155] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[156] F. Qu, F.-Y. Wang, and L. Yang, "Intelligent transportation spaces: Vehicles, Traffic, Communications, and BeyondLiuqing Yang," *IEEE Communications Magazine*, vol. 48, pp. 136–142, Nov. 2010.

[157] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A survey of Inter-Vehicle Communication Protocols and Their Applications," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 2, 2009.

[158] J. Luo and J.-P. Hubaux, *A Survey of Research in Inter-Vehicle Communications*. Embedded Security in Cars –Securing Current and Future Automotive IT Applications, Springer-Verlag, 2005.

[159] A. Amoroso, M. Ciaschini, and M. Roccetti, "The farther relay and oracle for VANET. preliminary results," in *IEEE WICON*, 2008.

[160] M. D. Felice, A. Ghandour, H. Hartail, and L. Bononi, "Enhancing the performance of safety applications in IEEE 802.11p/WAVE Vehicular Networks," in *IEEE WOWMOM*, June 2012.

[161] G. Korkmaz, E. Ekici, F. Ozguner, and U. Ozguner, "Urban multi-hop broadcast protocols for inter-vehicle communication systems," in *ACM Workshop VANET*, oct 2007.

[162] R. A. Uzcátegui and G. Acosta-Marum, "Wave: A tutorial," *IEEE Communications Magazine*, may 2009.

[163] *Performance Evaluation of the IEEE 802.11p WAVE Communication Standard*, IEEE Vehicular Technology Conference, VTC-2007 Fall., sep 2007.

[164] "Etsi tr 102 638 v1.1.1: Intelligent transport systems (its); vehicular communications; basic set of applications; definitions," tech. rep., 2009.

[165] M. van Eenennaam, W. K. Wolterink, G. Karagiannis, and G. Heijenk, "Exploring the solution space of beaconing in vanets.," IEEE Vehicular Networking Conference, VNC2009, oct 2009.

[166] Y. Wang, A. Ahmed, B. Krishnamachari, and K. Psounis, "Ieee 802.11p performance evaluation and protocol enhancement," IEEE International Conference on Vehicular Electronics and Safety, ICVES 2008., pp. 317–322, sep 2008.

[167] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *IEEE Vehicular Technology Magazine,*, vol. 2, pp. 12–22, jun 2007.

[168] Y.-W. LIN, Y.-S. CHEN, and S.-L. LEE, "Routing Protocols in Vehicular Ad Hoc Networks: A Survey and Future Perspectives," *Journal of Information Science and Engineering*, vol. 26, pp. 913–932, 2010.

[169] F. Qu, F.-Y. Wang, and L. Yang, "Intelligent transportation spaces: vehicles, traffic, communications, and beyond.," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 136–142, 2010.

[170] A. Broggi, P. Cerri, S. Ghidoni, P. Grisleri, and H. G. Jung, "A New Approach to Urban Pedestrian Detection for Automatic Braking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 594–605, 2009.

[171] M.-T. Sun, W.-C. Feng, K. Fujimura, T.-H. Lai, H. Okada, and K. Fujimura, "GPS-Based Message Broadcasting for Inter-vehicle Communication," in *ICPP*, aug 2000.

[172] M. Roccetti and G. Marfia, "Modeling and Experimenting with Vehicular Congestion for Distributed Advanced Traveler Information Systems," *Computer Performance Engineering Lecture Notes in Computer Science*, vol. 6432/2010, pp. 1–16, 2010.

[173] N. Ravi, S. Smaldone, L. Iftode, and M. Gerla, "Lane Reservation for Highways (Position Paper)," in *IEEE ITSC*, 2007.

[174] C. E. Palazzi, M. Roccetti, S. Ferretti, G. Pau, and M. Gerla, "Online Games on Wheels: Fast Game Event Delivery in Vehicular Ad-hoc Networks," in *IEEE V2VCOM*, June 2007.

[175] M. Fazio, C. E. Palazzi, S. Das, and M. Gerla, "Facilitating Real-time Applications in VANETs through Fast Address Auto-configuration," in *IEEE CCNC* , 2007.

[176] T.-S. Dao, K. Y. K. Leung, C. M. Clark, and J. P. Huissoon, "Markov-Based Lane Positioning Using Intervehicle Communication," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, pp. 641–650, Dec. 2007.

[177] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, *The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds*, pp. 645–700. Wiley-IEEE Press, 2013.

[178] G. Yan, D. Wen, S. Olariu, and M. C. Weigle, "Security Challenges in Vehicular Cloud Computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 284–294, mar 2013.

[179] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang, "Data Naming in Vehicle-to-Vehicle Communications," in *IEEE INFOCOM/NOMEN*, apr 2012.

[180] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik, "Cache Privacy in Named-Data Networking," in *IEEE ICDCS*, jul 2013.

[181] F. Angius, C. Westphal, J. Wei, M. Gerla, and G. Pau, "Prefix Hopping: Efficient Many-To-Many Communication Support in Information Centric Networks," in *IEEE INFOCOM/NOMEN*, apr 2013.

[182] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating Interest Flooding DDoS Attacks in Named Data Networking," *arXiv preprint arXiv:1303.4823*, 2013.

[183] E. Fernandes, B. Crispo, and M. Conti, "FM 99.9, Radio Virus: Exploiting FM Radio Broadcasts for Malware Deployment," *IEEE Transactions on Information Forensics and Security*, vol. PP, no. 99, pp. 1–1, 2013.

[184] G. Kambourakis, E. Konstantinou, and S. Gritzalis, "Revisiting WiMAX MBS security," *Journal Computers and Mathematics with Applications*, vol. 60, pp. 217–223, 2010.

[185] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[186] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *In ACM Communications Magazine*, vol. 21, pp. 120–126, Feb. 1978.

[187] J.-H. Song, V. W. Wong, and V. C. Leung, "Secure Location Verification for Vehicular Ad-Hoc Networks," in *IEEE GLOBECOM*, 2008.

[188] P. Golle, D. Greene, and J. Staddon, "Detecting and Correcting Malicious Data in VANETs," in *ACM Vanet Workshop*, oct 2004.

[189] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," *IETF RFC 2401*, Nov. 1998.

[190] P. Michiardi and R. Molva, "CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *IFIP CMS MANET*, Sept. 2002.

[191] Q. He, D. Wu, and P. Kholsa, "A secure incentive architecture for ad hoc networks: Research Articles," *Journal in Wireless Communications and Mobile Computing - Wireless Network Security*, pp. 333–346, May 2006.

[192] A. Weyland, T. Staub, and T. Braun, "Comparison of motivation-based cooperation mechanisms for hybrid wireless networks," *Journal of Computer Communications*, vol. 24, pp. 2661–2670, Aug. 2006.

[193] M. E. Mahmoud and X. S. Shen, "Stimulating cooperation in multi-hop wireless networks using cheating detection system," in *INFOCOM*, apr 2010.

[194] L. M. Feeney, "An Energy-Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks," *Mobile Networks and Applications*, vol. 3, no. 6, pp. 239–249, 2001.

[195] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *ACM MobiCom*, aug 2000.

[196] J. J. Jaramillo and R. Srikant, "DARWIN: Distributed and Adaptive Reputation mechanism for Wireless ad-hoc Networks," in *ACM MobiCom* , Sept. 2007.

[197] L. Buttyan and J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *Mobile Networks and Applications*, vol. 8, pp. 579–592, Oct. 2003.

[198] Q. Chen, D. Jiang, V. Taliwal, and L. Delgrossi, "IEEE 802.11 based Vehicular Communication Simulation Design for NS-2 ," VANET, 2006.

[199] "Dedicated Short Range Communications (DSRC) Home. [Online]. Available: http://www.leearmstrong.com/dsrc/dsrchomeset.htm .".

[200] W. B. Jaballah, H. Youssef, and A. Meddeb, "Efficient Signature schemes in Wireless Sensor Networks," SenSornet (Poster), pp. 1–2, dec 2010.

[201] W. Ben Jaballah, A. Meddeb, and H. Youssef, "An efficient source authentication scheme in wireless sensor networks," in *ACS/IEEE International Conference on Computer Systems and Applications- AICCSA 2010*, May 2010.

[202] *A Survey on Source Authentication Protocols in Wireless Sensor Networks*, International Symposium on Distance Education and Networks, dec 2010.

[203] W. B. Jaballah, M. Mosbah, H. Youssef, O. Ly, and A. Meddeb, "Modeling Source Authentication Protocols in Wireless Sensor Networks Using HLPSL," IEEE International Conference on Network and Information Systems Security, pp. 1–9, may 2011.

[204] W. Ben Jaballah, M. Mosbah, and H. Youssef, "Performance Evaluation of Key Disclosure Delay Based Schemes in Wireless Sensor Networks," in *IEEE PERCOM/PERSENS*, mar 2013.

[205] W. B. Jaballah, M. Mosbah, and H. Youssef, "Key Disclosure Delay based Schemes in Wireless Sensor Networks," *Elsevier Journal of Network and Computer Applications*, 2013. under review.

[206] W. B. Jaballah, M. Conti, R. D. Pietro, M. Mosbah, and N. V. Verde, "MASS: An Efficient and Secure Broadcast Authentication Scheme for Resource Constrained Devices," International Conference on Risks and Security of Internet and Systems , pp. 1–10, oct 2013.

[207] W. B. Jaballah, M. Mosbah, H. Youssef, and A. Zemmari, "Lightweight Source Authentication Mechanisms in Wireless Sensor Networks," 27th International Conference on Advances Information Networking and Applications, pp. 598–605, mar 2013.

[208] W. B. Jaballah, M. Mosbah, H. Youssef, and A. Zemmari, "Lightweight secure group communications for resource constrained devices," *International Journal of Space-Based and Situated Computing IJSSC (under review)*, 2014.

[209] W. B. Jaballah, M. Conti, M. Mosbah, and C. E. Palazzi, "Secure Verification of Location Claims on a Vehicular Safety Application," International Conference on Computer Communications and Networks, pp. 1–8, jul 2013.

[210] W. B. Jaballah, M. Conti, M. Mosbah, and C. E. Palazzi, "A secure alert messaging system for safe driving," *Elsevier Computer Communications ComCom*, 2013. under review.

[211] W. B. Jaballah, M. Conti, M. Mosbah, and C. E. Palazzi, "Fast and Secure Multihop Broadcast Solutions for Inter-Vehicular Communication," *IEEE Transactions on Intelligent Transportation Systems*, aug 2013.