UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE
Spécialité : **Informatique**

Présentée par

Raphaël Jamet

Thèse dirigée par **Dr. Pascal Lafourcade**

préparée au sein du laboratoire **VERIMAG**
et de l'**École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (EDMSTII)**

Protocols and Models for the Security of Wireless Ad-Hoc Networks

Thèse soutenue publiquement le **3 Octobre 2014**,
devant le jury composé de :

Foreword

This thesis has been redacted in English first. The introduction and the perspectives of this work have been translated to French, and can be found in the appendix. Furthermore, we provide a global abstract and chapter abstracts in French, respectively at the beginning of the manuscript and at the beginning of each chapter.

Cette thèse a été rédigée principalement en anglais. L'introduction et les perspectives de notre travail ont été traduits en français, et se situent à la fin du manuscrit. De plus, un résumé général de la thèse est disponible en français au début du manuscrit, et un résumé de chaque chapitre en français se situe au début de chaque chapitre.

Abstract

In this document, we focus on ways of increasing the security of wireless ad-hoc networks. These networks, and more specifically wireless sensor networks, look increasingly like the right answer to a lot of problem, such as data collection over a large area, or providing emergency network infrastructure after a disaster. They are also inherently exposed to malicious intents due to their collaborative nature. In order to protect them, we focus on the security aspects of the protocols built for these networks.

We first propose a Secure and Resilient Reputation-based Routing protocol, called SR3. This protocol routes messages according to a reputation metric built using only trusted information. This protocol achieves data confidentiality and data packet unforgeability, which we prove formally using two verification tools: *CryptoVerif* and *Scyther*. We use *Sinalgo*, an event-driven network simulator to run an experimental evaluation of SR3, and we compared our results to several routing algorithms of the literature. This evaluation shows that both the resiliency and fairness accomplished by SR3 are better than for those others protocols, especially when the network is sparse. Moreover, and unlike previous solutions, if the compromised nodes behavior changes, then SR3 will self-adapt in order to ensure an acceptable quality of service.

Analyses of routing protocols security are nearly always supported by simulations, which often evaluate the ability to deliver messages to a given destination. Several competing definitions for secure routing exist, but to our knowledge, they only address source routing protocols. We propose the notion of *incorruptibility*, a quantitative computational definition for routing security based on the attacker's ability to alter the routes used by messages. These definitions are then illustrated with several routing algorithms.

Finally, we study Intrusion Detection Systems (IDS) for WANET, and more specifically their inputs. These systems provide a supplementary layer of defenses for WANETs, and they are able to easily detect attacks who are complicated for the network protocols. We classify the different inputs used by the decision process of these IDS, according to their level of required cooperation, and the source of their data. We then propose the InDICE tool, a decision aid which, given an IDS, allows automated discovery of undetectable attacks according to the inputs used by that IDS. In the end, we apply our framework to discover weaknesses in two existing IDS.

Résumé

Dans cette thèse, nous nous intéressons à plusieurs méthodes pour améliorer la sécurité des réseaux sans fil ad hoc. Ces réseaux, ainsi que la sous-famille des réseaux de capteurs sans fil, sont une des solutions les plus intéressantes pour de nombreux problèmes, comme par exemple la collecte de données dans une large zone, ou bien la création d'infrastructure de communication après une catastrophe. Ces réseaux sont par nature collaboratifs, ce qui les rend très vulnérables à d'éventuels attaquants. Pour les protéger, nous étudions la sécurité des protocoles conçus pour ces réseaux.

Premièrement, nous proposons SR3 (Secure and Resilient Reputation-based Routing), un algorithme de routage sécurisé et résilient pour le routage *convergeant* (tous-vers-un) dans les réseaux de capteurs sans fil. SR3 route ses messages selon une mesure de réputation qui est bâtie sur des informations fiables. Ce protocole garantit la confidentialité de ses données, et l'inforgeabilité de ses paquets. Nous avons prouvé formellement ces propriétés avec deux outils de vérification : *Scyther* et *CryptoVerif*. Nous avons montré expérimentalement à l'aide de Sinalgo, un simulateur à évènements discrets, la résilience de SR3 quand confronté à divers scénarios d'attaque, et nous avons comparé nos résultats à plusieurs algorithmes de routage de la littérature. L'évaluation a montré que la résilience et l'équité fournies par SR3 sont meilleures que celles des autres protocoles, et cette distinction est accentuée si le réseau est peu dense. De plus, et contrairement aux autres protocoles, SR3 est capable de s'auto-adapter aux changements de comportement des attaquants afin d'assurer une qualité de service satisfaisante.

Les analyses de la sécurité des protocoles de routage reposent presque toujours sur des simulations, qui évaluent la capacité du protocole à délivrer ses messages aux bons nœuds. Il existe plusieurs définitions différentes pour concevoir la sécurité du routage, mais à notre connaissance, elles considèrent seulement les protocoles de *source routing*, où les routes sont déterminées avant que le message ne soit envoyé. Nous proposons la notion de *corruptibilité*, une définition calculatoire et quantitative pour la sécurité du routage basée sur la capacité d'un attaquant à altérer les routes empruntées par un message. Nous illustrons ensuite ces définitions par plusieurs analyses de protocoles.

Enfin, nous étudions les systèmes de détection d'intrusions (IDS) pour réseaux sans fil ad hoc, et plus spécifiquement les sources de données utilisées pour leurs mécanismes de décision. Nous classifions celles-ci en fonction du niveau de coopération qu'elles requièrent, et en fonction de l'origine de leurs données. Nous proposons ensuite InDICE, un outil d'aide à la décision qui étant donné un IDS, permet de découvrir automatique-

ment quelles attaques seront indétectables par les sources de données qu'utilise cet IDS. Enfin, nous utilisons cet outil pour découvrir deux vulnérabilités dans des IDS de la littérature.

Remerciements

Je tiens avant toute chose à remercier Pascal Lafourcade, pour avoir accepté de m'encadrer pendant ces trois ans. Son aide, et la patience qui est venue avec, m'auront été indispensables pour apprendre ce qu'est la rigueur d'un travail de chercheur. De même, je souhaite remercier Stéphane Devismes et Karine Altisen pour leur aide pendant une grande partie de mon passage à VERIMAG. A eux trois, ils m'ont permis de m'épanouir, et je leur en suis particulièrement reconnaissant.

Je remercie également les membres du jury, et particulièrement les rapporteurs Mme Marine Minier et M. Michel Misson, pour avoir accepté de juger ce manuscrit, mais également mes examinateurs : Pr. Abdelmadjid Bouabdallah, Pr. Hervé Debar, Dr. Marcelo Dias de Amorim et Pr. Bernard Tourancheau, pour leur intérêt pour mon travail. Leurs questions durant la soutenance m'ont aussi aidé à me rendre compte de l'importance des considérations énergétiques pour la sécurité dans les réseaux ad-hoc, et j'espère avoir l'occasion de continuer dans cette direction.

Enfin, j'exprime ma gratitude envers les membres de VERIMAG. Trois ans au laboratoire auraient été longs sans avoir de compagnie, et pour ça, je souhaite remercier les permanents et les autres doctorants de Verimag, pour avoir éclairé mon passage au laboratoire. Plus particulièrement, les membres de mon bureau: Eduardo, Christian, Jannik, Mathilde, Ali, Vikas, Cristina, et Louis, mais aussi les autres: Julien, Romain, Nicolas, Valentin, Josselin, Ozgun, Alexandre, Antoine, Emmanuel, ... De plus, merci à Securimag et ses membres, pour tous ces jeudi soirs très instructifs.

Enfin, je souhaite remercier mes amis, et ma famille pour leur soutien. Ils sont trop nombreux pour les lister ici, mais ils ont tous été là quand j'en avais besoin, et sans eux ces trois ans n'auraient pas été les mêmes.

Contents

Chapter 1

Introduction

Contents

1.1 Introduction

Wireless networks are everywhere nowadays. Phones, payment systems, urban infrastructure, computers: more and more items use some form of wireless communication system to join networks, whether they are local or connected to the Internet.

In this thesis, we are interested in wireless ad-hoc networks (WANET). An ad-hoc network is a network where all the *nodes* route messages, without relying on a dedicated infrastructure. Communication in such networks will require the cooperation of intermediary nodes if the destination of a message is not in the vicinity of the emitter.

These networks look more and more like the right answer to a lot of data collection problems. For instance, the authors of [AGS11] used a WANET to share a central Internet access point among the geographically spread-out residents of a small town. This allows a good connectivity, without the upfront cost of creating wired links for each resident. In [GSCL+13], the authors describe another system, built on top of the Android operating system. Their system allows communications and services among a group of smartphone-equipped people when there is no other infrastructure available, for instance after a disaster. In that cases, users of the system can reach each other with secure calls, messages, and various other applications.

Wireless sensor networks (WSN) are a special subfamily of wireless ad-hoc networks. They are networks of interconnected sensors and repeaters, built to provide connectivity at a lower cost than wired networks. Sensors are typically small battery-powered devices that generate data about the environment (*e.g.*, temperature) and use them for specific services (*e.g.*, emit an alarm when the surrounding temperature is too high). For in-

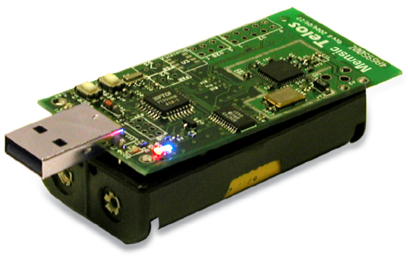


Figure 1.1: A TelosB mote

stance, TelosB motes (Figure 1.1) are equipped with a 8Mhz microcontroller and with 10KB of RAM, which limits their ability to run expensive cryptographic algorithms. These motes are equipped with ZigBee-compliant radios.

Use cases for these type of networks are very varied. For instance, a survey of wireless sensor networks in the Netherlands [MVdZVD$^+$10] showed several uses of these networks in production. One of these, called GuArtNet[1], is a system to monitor valuable items in a museum. Several sensors are placed on individual objects, and detect when they are moved. The alert is then propagated to the rest of the system, through repeaters, towards a *sink* (the central node), who will react accordingly. Sensors are tiny devices, running on battery power, whereas the sink is often a full-fledged server. Sensors regularly send heartbeats to the sink, so that any loss of connectivity is detected, but they can also send alerts in case of movement, tampering, or radio interference for instance.

WANET Specificities

All WANET have some specificities due to their wireless nature. Wireless communications are strongly tied to antennas, which can be *omnidirectional* or have various *directionnal* emission patterns which favors some directions over others, depending on their design. Furthermore, some techniques allow an emitter to knowingly direct its emissions, using for instance *phased array antennas* [Han09]. Similarly, a link can be *symmetric* if both devices can communicate with each other, or *asymmetric* if it only works one way. Finally, the links may fluctuate over time: their quality depend on several factors such as noise, meteorological conditions, or people moving around.

Those wireless specificities need to be taken into account when choosing or designing network protocols for WANET, but there are also challenges specific to their ad-hoc nature. These networks are often collaborative, which makes misbehaving nodes a serious threat, whether they are caused by node compromise (which may be made more difficult with hardened nodes), or because the network is open by design. Furthermore, some networks are dynamic and/or mobile by nature, in which case the protocols must be

[1]`http://www.sownet.nl/index.php/en/products/guartnet`

designed accordingly. Finally, the hardware used by the nodes may have specificities also: nodes may have very low computing power, there may be several different types of nodes in the networks, communication bandwidth may be very low, nodes may have a limited energy supply, human intervention on the devices may not be possible, the network topology may be very sparse or very dense... All these factors must be taken into account.

Security Properties and Secure Systems

Security is usually expected of these systems. Consider for instance the GuArtNet WSN example: intuitively, the network user assumes that alerts are authentic, that heartbeats can be trusted, and that if an alert occurs, then it will reach the sink in a timely manner. Most, if not all applications that are built on top of ad-hoc networks include these kinds of expectations, which correspond to *security properties*, such as *confidentiality* or *authentication* for instance.

Two main models are used to prove that protocols guarantee these properties: computational model, and the symbolic model. These models were created to prove cryptographic constructs and although they evolved independently in the beginning, they have been shown to be related in [AR00]:

- The *computational* model [GM84, GMW91, BDJR97, BKR00] represents the attacker using a probabilistic Turing machine, running in polynomial time, with polynomial memory. Thanks to this model, we can obtain an upper bound on the probability of any attacker breaking the property. For instance, this model was used to prove the security of the RSA-OAEP cryptosystem [FOPS01].
- In the *symbolic* model [DY83, BAN89, AG97], data is represented as function symbols, which can be combined to form other symbols. In this model, the attacker is able to play new sessions of the protocol, and create new messages from its own knowledge. This model was used to discover the attack on the Needham-Schroeder protocol [Low95].

Proof Scope and Security in Depth

Security proofs ensure that in a certain model, the analyzed protocol guarantees the security property at hand. These proofs may also be automated, in which case the proof steps are exhaustively identified, avoiding the errors that may happen with handwritten proofs. However, as with all models, there may be differences between the model and the reality, which may result in out-of-scope attacks.

To illustrate this point, we present a recently discovered (March 2014) vulnerability[2]. on the Transport Layer Security (TLS) protocol suite, caused by the three-way handshake in TLS. This protocol suite is a de facto standard on the internet, widely

[2]The corresponding academic publication is currently being reviewed, more details can be found at `https://secure-resumption.com/`

used to create secure channels between two nodes, and officially adopted by the IETF in [DR08].

TLS includes several protocols in its handshake. An attacker can use three of them and route messages between two actors Alice and Bob, so that they both believe they communicated directly with each other. However, the attacker injected information into what Bob believes he received from Alice. This is a breach of authentication and integrity. This vulnerability lies in the TLS protocol suite itself, where an attacker could exploit the master secret mechanism of TLS in order to reuse previous values. What is interesting is surprising, as the security of TLS has been proven several times using various models [Pau99, HSD$^+$05, GMP$^+$08, MSW08, BFK$^+$13]. To be fair, one should mention that it has also been broken several times, both in its design and implementations (for instance, [CHVV03, DR11, DR12, AFP13]). This TLS vulnerability example shows that even if formal proofs show the security of a system in a certain model, some attacks may not be in scope. In such cases, security in depth is useful: a signature-based intrusion detection system may be able to detect those attacks which are not prevented by the protocol, or an anomaly detection system may be able to detect unexpected behavior in general. Intrusion response systems can then do something about the intrusion, or at least raise an alert for further analysis.

Evolving requirements

In some cases, some systems which were not expected to be secure when they were created became part of security-critical systems. Several such examples can be found in the Internet protocols, as it has been founded upon mutual trust between all actors, and this disposition strongly influenced the underlying software infrastructure.

For instance, the Address Resolution Protocol (ARP, standardized in [Plu82]) is frequently used on local networks to link logical (IP) addresses to physical (MAC) addresses. To give a rough idea, nodes that need to find a logical node broadcast a "who has this IP" packet, which should be answered by the holder of the IP. This protocol however does not include any sort of authentication, and an intruder can take advantage of this fact by forging an answer, thus stealing another device's IP. Over wired networks in controlled physical environments, this weakness seems quite harmless. However, domestic WiFi networks also use this protocol, and it becomes far more dangerous in the case of an open network: as any computer can join without the need to physically access the equipment, the vulnerability allows anyone to impersonate a node from the network. Another notorious example is the Border Gateway Protocol (BGP), which decides where packets are routed between autonomous networks on the Internet. Originally, security was not a design goal of this protocol, and so, regular incidents occur nowadays where third-party nodes obtain a degree of control on the route generated for messages that would not normally be routed by them. Such influence could be used maliciously to do traffic analysis or to artificially degrade a company's quality of service, for instance. To address this, we define in Chapter 3 a quantitative measure of the possible influence an attacker can have over the route of a message, and we define the corresponding security property of *incorruptibility*.

1.2 Contributions and organization

This thesis is built around three main chapters. Our first contribution is SR3, a *Secure and Resilient Reputation-based Routing* protocol built for convergecast routing in WSN, which we present in Chapter 2. We designed this protocol as a reinforced random walk that routes messages according to a reputation metric computed using only trusted information. We prove this protocol in the computational model using CryptoVerif [Bla08], then in the symbolic model using Scyther [Cre08]. We use *Sinalgo* [Dis08], an event-driven network simulator to run an experimental evaluation of SR3, and to compare it to several others from the literature: Greedy-Face-Greedy [BMSU01], the uniform random walk, SIGF [WFSH06], and several variants of the gradient-based routing protocol built for their resiliency [EOMVK11]. This evaluation looks into several criteria: length of routes, delivery rate against several intruder models, and resiliency as presented in [EOMVK11].

The second contribution also deals with routing: in Chapter 3, we introduce a definition for the security of routing protocols named *incorruptibility*. The other routing security measures we found deal with source routing protocols only: our notion measures the ability for an attacker to significantly change how messages are going to be routed through the network. We first present the formalization of this notion, followed by a generalization of it called *bounded corruptibility*. We provide several example protocols to which we apply this definition, and discuss how the definition can be expanded to more routing protocols.

The third contribution is related to intrusion detection systems. These systems are critical to the security of ad-hoc networks, and evaluating them is very often done experimentally. We therefore propose in Chapter 4 two results for their improvement. The first one is a classification of the inputs on which their decision process rests. We present a review of several IDS of the literature, we identify their inputs, and categorize them according to the level of cooperation they require in the network and to the origin of their data. Following this, we present the Intrusion Detection Input Coverage Evaluation (InDICE), a decision aid to discover oversights in IDS from their inputs. The main idea of this tool is to model the logical progression of attacks, in order to be able to discover if a target attack is doable by an attacker without taking any step that are observed by this IDS's inputs. We finally apply this model to two IDS from the literature: [dSMR$^+$05] and [OM05b].

1.3 Publications

Most of the original work presented here have been published in international conferences. The SR3 protocol, and most of the results presented in Chapter 2 have been published in the international conference DCOSS'13 [ADJL13b], and in the national conference Algotel'13 [ADJL13a]. An extended journal version is currently under review. All the work presented in Chapter 4 has been published in FPS'13 [JL13a], an international conference. The work in Chapter 3 has been accepted at FPS'14, and will be published a short time after the submission of this thesis.

Finally, we built a model to verify protocols revolving around physical properties of wireless networks (especially neighborhoods). This work resulted in a model able to take into account node movement and time, which we used to verify a simple protocol from the literature. We then proposed the building blocks for a protocol able to securely discover the $n+1$-or-less neighborhood of a node, if each node in the network knows its n-or-less neighborhood. This work has been published as a chapter of a book [JL13b], and we did not include it in this document.

Chapter 2

SR3: Secure and resilient reputation-based routing

In this chapter, we propose SR3, a secure and resilient algorithm for convergecast routing in wireless sensor networks. SR3 uses lightweight cryptographic primitives to achieve data confidentiality and data packet unforgeability. The security of SR3 has been proven formally using two verification tools: CryptoVerif and Scyther. We made simulations to show the resiliency of SR3 against various scenarios, where we mixed selective forwarding, blackhole, wormhole, and Sybil attacks. We compared our solution to several routing algorithms of the literature. Our results show that the resiliency accomplished by SR3 is better than the one achieved by those protocols, especially when the network is sparse. Moreover, unlike existing solutions, SR3 self-adapts after compromised nodes suddenly change their behavior.

Dans ce chapitre, nous proposons SR3 (Secure and Resilient Reputation-based Routing), un algorithme de routage sécurisé et résilient pour le routage *convergeant* (tous-vers-un) dans les réseaux de capteurs sans fil. SR3 se sert de primitives cryptographiques légères pour garantir la confidentialité des données routées ainsi que l'inforgeabilité de ses paquets. Ces propriétés de sécurité ont été prouvées formellement avec deux outils de vérification : *Scyther* et *CryptoVerif*. Nous avons montré expérimentalement à l'aide de Sinalgo, un simulateur à évènements discrets, la résilience de SR3 quand confronté à divers scénarios d'attaque, et nous avons comparé nos résultats à plusieurs algorithmes de routage de la littérature. L'évaluation a montré que la résilience et l'équité fournies par SR3 sont meilleures que celles des autres protocoles, et cette distinction est accentuée si le réseau est peu dense. De plus, et contrairement aux autres protocoles, SR3 est capable de s'auto-adapter aux changements de comportement des attaquants afin d'assurer une qualité de service satisfaisante.

Contents

2.1 Introduction

In this chapter, we propose a routing protocol designed for WSNs. This protocol is designed to ensure some basic security properties, and so that messages reach their destination. We consider a routing scheme called *convergecast* routing, where, as described in the introduction, one of the nodes is distinguished as the *sink*, and all non-sink nodes, called *source nodes*, must be able to transmit data to the sink on request or according to an *a priori* unknown schedule. The sink can be arbitrary far (in terms of hops) from other nodes.

A routing protocol in a WSN may have to face many kinds of attacks. Here, we consider the critical scenario, where some sensors are compromised and controlled by an attacker. In particular, such an internal attacker has access to all secret and received information of the compromised nodes.

The attacker can impact the routing protocol at two main levels:

Packet Level. First, he can attack the data packet to learn secret information, *i.e.*, violate the data *confidentiality*, as this property consists in guaranteeing that data remain secret between the source and destination.

He can also make the sink deliver incorrect information, *i.e.*, violate the *integrity* of the data messages. Integrity guarantees that the destination is able to detect whether the data inside a packet have been modified.

Moreover, the attacker can act against the *authentication* of the nodes. Authentication guarantees that the destination is able to detect whether the alleged source in a packet is the truth one.

Routing Level. Secondly, the attacker can affect the routing scheme itself, *e.g.*, he may prevent data from being delivered by the sink (leading to degrade the quality of service, essentially the delivery rate), or create congestion by increasing the load in all or part of the network (leading to reduce the lifetime of the network). Such an attack can also evolve over time, *e.g.*, the attacker can attract traffic to a given intruder node using various means. Thanks to that, the intruder node (called a *sinkhole*) will have more impact for further malicious actions.

The notion of *resiliency* has been introduced in [EOMVK10a] as the ability of a network to “continue operating” in presence of compromised nodes, *i.e.*, the capacity of a network to endure and overcome internal attacks. For example, a resilient routing protocol should achieve a “graceful degradation” in the delivery rate with increasing the number of compromised nodes. It is one of our goals in the design of SR3, and we use that notion along this chapter.

The packet level security consists of a set of properties which are traditionally proved using a cryptographic model of the protocol and of the security objectives, whereas security proofs against routing attacks are fairly rare and only address certain specific properties (see Chapter 3 for more discussion of this issue). Usually, these concerns are addressed through experimental evaluations, which show the algorithm’s resistance to a set of specific attackers. This approach is not exhaustive, as it does not cover the full range of possible routing attacks. In this chapter, we used this method, and we only consider the specific attackers that are relevant to our algorithm. More specifically, we looked at sinkholes as SR3 uses a reputation mechanism, and Sybil nodes since they are very effective against random walks. In the following chapter, we provide the first steps towards a new formal framework for the verification of secure routing.

2.1.1 Outline

This chapter is organized as follows. We first present our contribution and several related routing algorithms. In the next section, we present our routing algorithm, SR3. Section 2.3 deals with the automatic proof of the security properties of SR3 using CryptoVerif [Bla08] and Scyther [Cre08]. In Section 2.5, we present experimental results that show the resiliency of SR3. Section 2.6 is dedicated to concluding remarks.

2.1.2 Contribution

This chapter deals with the convergecast routing in WSNs, where all source nodes have several messages to route. We propose a *Secure, Resilient, and Reputation-based Routing* algorithm, called SR3. This protocol is a reinforced random walk that is partially determinized using a reputation mechanism.

SR3 uses lightweight cryptographic primitives — symmetric cryptography, nonces, and hash functions — to achieve security properties: data confidentiality and data packet unforgeability, this latter property implies integrity and authenticity of the data packets. We prove these properties in the computational model using the formal tool *CryptoVerif* [Bla08]. We also prove in the symbolic model the secrecy of data and authentication of nodes using the tool *Scyther* [Cre08].

Then, we show the resiliency of SR3 against various scenarios, where we mixed selective forwarding, blackhole, wormhole, and Sybil attacks. We compare ourselves to a panel of algorithms from the literature: Greedy-Face-Greedy [BMSU01] which is a geographical routing algorithm, the uniform random walk, SIGF [WFSH06] which is based on a reputation mechanism, and several variants of the gradient-based routing protocol [EOMVK11] including one specifically built to be resilient. Our simulation results show in particular that unlike the other solutions, SR3 self-adapts when compromised nodes change their behavior (*e.g*, an interesting case is when a compromised node behaves well to attract the traffic and then suddenly decide to drop all received messages). Our results show that the resiliency accomplished by SR3 is better than the one achieved by those protocols, especially when the network is sparse.

A shortcoming of our solution is the number of hops to reach the destination, as it is usually greater than other solutions of the literature. However, in our experiments, we observed that this complexity remains sublinear in the number of nodes.

Note also that our solution is *reactive* (*i.e.*, in absence of data to route the protocol eventually stops.), has a low overhead in terms of communications, and does not use any underlying infrastructure, such as spanning tree. Hence, SR3 is well-suited for WSNs.

2.1.3 Related Work

Routing protocols for ad-hoc networks are numerous, and we therefore only present a few protocols, and focus more on the security aspects and reputation-based protocols.

Numerous solutions have been introduced to cope with attacks on data. The confidentiality, authenticity, and integrity properties are mainly guaranteed using cryptographic mechanisms. However, the choice of the cryptographic primitives should be led by the inherent constraints of WSNs. Such network are limited in terms of resource and power, and so *lightweight* cryptographic mechanisms [EK07, CMM13] are mandatory. An example of such a mechanism is *elliptic curve cryptography* [Kob87, Mil86], which is becoming commonplace in wireless networks (see for instance the TinyECC library [LN08]). In contrast, classical asymmetric cryptography, *e.g.*, *RSA* [RSA78], should be excluded due to its computational cost.

Although it is not strictly a routing protocol, *SPINS* [PST$^+$02] is a set of tools for routing, which provides security guarantees without using any costly operations. The first tool is named *SNEP*, and it consists of a packet format that guarantees various security properties, like authentication and confidentiality, using few additional bits per packet. The second one is named μ-Tesla. It is a broadcast authentication protocol that enables receivers of the broadcast data to verify that these data really originate from the alleged sources. μ-Tesla has a low communication and computation overhead thanks to lightweight cryptography, scales a large number of receivers, and tolerates packet loss.

Route discovery protocols

The family of *secure route discovery protocols*, related to source routing protocols, have been introduced [PH02, HPJ05]. They are not strictly routing messages: instead, they only compute a valid route (*i.e.*, the computed path exists in the network) between the source and destination, and for some of them (*e.g.*, [PH02]), they guarantee that nodes in the chosen route achieved a certain security level, *e.g.*, the integrity of the discovered route, which means that the computed route has been effectively traversed during the discovery process. Note that SR3 do not have a route discovery process, as each of the node dynamically determine the next hop for a message when they receive it.

DSR The Dynamic Source Routing protocol (DSR, [JM96]) is an on-demand source routing protocol which is often associated with WSNs. This protocol builds routes upon request, by flooding *route request packets*, to which each node appends its identifier when it rebroadcasts. When a node receives such a packet and it is the intended destination of the request, it sends a *route reply packet* along the route contained in the packet. Finally, when the initiator receives that packet, it uses the route inside to send its data. Several mechanisms are added on top of this in order to cache information, maintain routes when the information is not up-to-date, and manage network interoperability.

SDSR Secure DSR (SDSR, [KGSW05]) is an extension for DSR. It requires the existence of a signed asymmetric key pair for each node in the network, in order to provide route integrity, freshness, to authenticate the nodes participating in the route, and to provide session keys among the participants in a route. The authors formalized the protocol using BAN logic, which allowed them to show that the protocol meets their security claims.Note that the requirement for nodes to be able to use asymmetric cryptography in SDSR is one we actively avoided in SR3, as asymmetric cryptography is costly. Also, SR3 does not authenticate intermediary nodes in any way in its routes, and instead we chose to make it adapt to malicious behavior.

Ariadne and endairA The Ariadne protocol [HPJ05] is another secure variant of DSR. This protocol can use one of several different pre-authentication methods (pre-shared symmetric keys, signatures or Tesla), and so, unlike SDSR, Ariadne does not necessarily require the use of asymmetric cryptography. With this information, Ariadne authenticates the request packets to ensure that they come from a trusted node, which prevents some attacks. Furthermore, they include some preventive methods against denial of service caused by excessive route requests, even when they are coming from an authenticated node. In [ABV06], the authors show that despite the arguments of [HPJ05], Ariadne is still vulnerable to route shortening attacks when two insider attackers collaborate. They propose an optimized version of Ariadne, named endairA. The idea behind this fixed protocol is to authenticate each hop during the reply, instead of authenticating the request.

AODV and related

AODV Ad-hoc On-demand Distance Vector routing (AODV, [PR99]) is notably used in ZigBee. It is a many-to-many routing protocol, designed for WANETs. It is related to DSR in that the route request/reply phases are quite similar. Their most notable differences are the use of sequence numbers in AODV, and route generation policy. AODV generates routes only when they are needed, and opportunistically uses all the information from the route generation of other nodes. The information is then kept and used for subsequent data routing during some time. However, AODV does not provide security mechanisms, and so several extensions have been proposed to enhance this protocol's security: a survey of these extensions and a discussion of their various mechanisms may be found in [VMWS12]. We present two of these extensions.

AODV and AODV-based protocols have several key differences with SR3. First of all, AODV is a many-to-many routing protocol, without acknowledgments, while SR3 is many-to-one with acknowledgments. Furthermore, AODV has a cooperative route discovery process, unlike SR3 who routes messages based on reliable local knowledge, without relying on neighbor's declarations. This allows networks running AODV to be more efficient in routing messages to/from nodes moving or joining the network than SR3, at the cost of more complexity, and no security.

SAODV Secure AODV (SAODV, [Zap02]) is an extension to AODV which adds public-key cryptography to AODV, by signing the control messages fields which are not modified during the control message's lifetime. They also propose a hash chaining mechanism to secure the hop counters of messages. In [VMWS12], the authors show an analysis of SAODV that exposes vulnerabilities to wormholes and rushing attacks, blackholes, and several availability attacks. They also argue that the distribution of the public key material is complicated in the case of networks where new nodes may be added, as key material cannot be distributed prior to deployment anymore.

SEAR Secure Efficient Ad-hoc On-demand Routing (SEAR, [LZW$^+$09]) is another secure variant of AODV. It is designed to secure some of the control packets of AODV, in order to ensure their authenticity. For this, they use a hash-chain mechanism that replaces the hop count field in route request messages, ensuring that intruders need to increment the hop count in order to keep the control message valid. This protocol prevents some sinkhole attacks with this mechanism, but none of the proposed improvements to AODV enhance the delivery rate in case of attackers that drop messages.

Link-state routing

LSR The Link State Routing algorithm (LSR, [MRR80]) is a routing protocol for ad-hoc networks. This algorithm maps the whole network on each node, by having them advertise their neighbor list periodically. This way, all the nodes are aware of the topology of the network, and so they can route messages optimally. Note that LSR and variants are not source routing protocols. As for AODV, this is a many-to-many protocol, unlike SR3. The memory requirement for this algorithm scales linear Furthermore, no security is provided.

OLSR Optimized Link State Routing (OLSR, [CJ03]) is a routing protocol for mobile ANET, based on the previous protocol. OLSR deviates from LSR because of the use of MultiPoint Relays (MPR). These nodes are elected by their neighbors, so that each node is either a MPR, or a neighbor of one. Then, OLSR uses these nodes to optimize the flooding mechanism of LSR by avoiding unnecessary broadcasts from non-MPR nodes. Several other miscellaneous optimizations are provided by OLSR.

Reputation-based protocols

CASTOR A secure routing protocol called *Continuously Adapting Secure Topology-Oblivious Routing* (CASTOR) for WANETs has been proposed in [GPP$^+$10]. This reputation-based routing protocol is centered on the notion of independent *flows*. A flow is created when a source node wants to send messages to a certain destination, and it will be used for a finite amount of messages. Messages identifiers are built in a way that ensures anyone is able to verify they belong to a given flow, using Merkle trees (which are binary trees where each node is the hash of the concatenation of its sons) with the hash of nonces as leafs, and the flow identifier as root. To route messages, each node stores a reputation value per neighbor for each flow. If a neighbor has sufficient reputation, and that neighbor is the most trusted, then the node will chose him as the next hop for the given flow. Otherwise, the node will broadcast the next messages to add redundancy. This process is then repeated along the route of the message. Once the message reaches its destination, it causes the emission of an acknowledgment, and then reputations are updated accordingly along the route this particular message took (several acknowledgments may be sent if there was replication). If a node does not receive an acknowledgment in a given delay for a certain message, then the reputation of the neighbor that routed this particular message is reduced.

At first glance, SR3 and CASTOR look similar, but there are several key differences. First of all, CASTOR is not based on a random walk: where SR3 always route message with a random component, CASTOR is mostly deterministic, except for a small part of the routing mechanism where nodes have a variable probability of broadcasting messages instead of sending them to the most trusted node. This is a cause for concern, as an attacker could predict a part of the behavior of nodes accurately. Also, SR3 is a convergecast routing protocol where the same reputation measure is used for all messages, whereas CASTOR is designed for any-to-any communication, and keeps independent reputations for each flow.

SIGF SIGF is a family of protocols using a reputation metric which has been proposed in [WFSH06]. This family is made out of four protocols, based on a geographic routing protocol. We focus on the SIGF-2 variant (and we refer to it by SIGF), the most secure variant according to the authors. This protocol supposes that nodes are able to overhear their neighbor's communications, that they have access to trustworthy geographical information, and that they know their 2-neighborhoods, which are strong assumptions. Nodes route messages by selecting randomly a next hop among a pool of nodes geographically in the right direction. Nodes also assign a reputation score to each of their neighbors, based on four metrics:

- The proportion of messages being reforwarded by that neighbor (higher is better),
- Its forwarding delay (lower is better),
- How its declared location changes over time (lower variation is better),
- The number of times it has been sent messages before (lower is better).

Nodes under a certain reputation threshold are removed from the pool of candidates for routing. Finally, the messages are authenticated with unspecified pre-shared information and encrypted. Note that the authors acknowledge that the reforwarding rate computation may be altered due to collisions, collusion, and communication asymmetry.

There are a lot of differences between SIGF and SR3. The most obvious are the absence of acknowledgments, the requirement for geographical information, and the assumption that nodes know their 2-neighborhood. Furthermore, SIGF is a many-to-many protocol, unlike SR3. This causes the reputation mechanism to be radically different: in SIGF, nodes base their trust upon the reforwarding rate, time-based information, and localization changes. Out of those, only time and localization changes can be trusted, while the reforwarding rate may be tricked. We detail an attack on this mechanism in 2.4.2.

Gradient-based Routing and variants

Specific approaches have been proposed to maintain a good quality of service in presence of insiders, that drop all or part of messages. For example, In [EOMVK11, EOMVK10b], authors experimentally analyze the resiliency of several classical routing techniques, *e.g.*, random walk [AKL$^+$79], gradient-based routing [SS01], geographic routing [BMSU01]. Their experimental results show that these solutions do not provide satisfactory resiliency, and so they propose several variants of the gradient-based routing (*i.e.*, a protocol in which messages are routed following a breath-first spanning tree) to increase its resiliency. Mainly, they introduce randomization and duplication in that protocol. As a result, the proposed patches increase the delivery rate when the network is subject to selective forwarding or blackhole attacks. However, in their simulations, they always assume that the breath-first spanning tree is available and not attacked by the insiders. Moreover, they mainly consider dense networks in their simulations, *e.g.*, networks with an average degree around 30.

Routing for low-power and lossy networks (RPL)

RPL (”Ripple”, defined in RFC 6550 [WTB$^+$12]) is a routing protocol specifically designed for low-power and lossy networks. RPL mainly allows communications from or to specific central points in the network, which may for example be sinks or interconnexions between networks.

Similarly to gradient-based routing protocols, it uses one or several trees rooted at those central points in order to route messages. However, these trees may differ in both their root, and their criteria for construction: RFC 6550 specifies that the tree-building process may use various objective functions. For instance, a function defined in RFC 6552 [Thu12] aims to minimize the number of hops to a root node (similarly to GBR). Having several routing trees allows modularity, by for instance allowing nodes to route low-priority messages in a low-power mindset, while high-priority messages could be routed in another tree specifically built to optimize latency.

Note that these trees can also be used for communications between arbitrary nodes in the network. RFC 6550 also provides several security settings, such as a network-shared key authentication, with an optional privilege separation between the keys for regular nodes, and terminal nodes (that are only allowed to be leafs in routing trees).

Besides the objective function, RPL routes messages in a deterministic way. This is a fundamental difference with SR3, which is at heart a probabilistic algorithm. A SR3-like reputation mechanism would not be fully portable in RPL, as it requires a probabilistic choice of next hop. Furthermore, RPL allows several routing schemes, while SR3 allows only convergecast communication.

Evaluating a protocol’s resiliency

Resiliency, as defined in [EOMVK10a], is the ability of a network to “continue operating” in presence of compromised nodes. The authors of this definition proposed a metric to quantify this ability in [EOKMV12]. Their metric is based around five axis: the delivery rate of the algorithm, the standard deviation of the nodes' delivery rates, the energy efficiency of the routing, the delay efficiency of the algorithm and the average throughput of the network. These five measures are then arranged into a pentagon, the surface area indicating the resiliency value. This measure is computed for several amount of attackers in the network.

In this chapter, we observe several of these indicators individually (delivery rate, their distribution, and time efficiency) to evaluate our algorithm's resiliency. However, some of the measures are strongly influenced by the simulation parameters: for instance, not simulating radio interference would artificially increase the throughput of algorithms that duplicate messages. We therefore chose not to use their unified metric for resiliency.

2.2 SR3

Randomization is interesting to obtain resilient solutions because it generates behaviors unpredictable by an attacker. However, we note that the "classical" *uniform* random walk, where a node chooses the next hop uniformly at random among its neighbors, is known to be inefficient even against a small number of compromised nodes, where its delivery rate is extremely low [EOMVK10a].

To avoid this problem, we designed SR3 rather as a *reinforced* random walk, based on a reputation mechanism. The idea is to locally increase the probability of a neighbor to be chosen at the next hop, if it behaves well. Such a reputation mechanism is based on acknowledgments. We propose a scheme in which if a process receives a valid acknowledgment, it has the guarantee that the sink actually delivered the corresponding data message. Hence, upon receiving such an acknowledgment, a node can legitimately increase its confidence on the neighbor to which it previously sent the corresponding data message. Eventually, all honest nodes will preferably choose their highly-reputed neighbors, and so the data messages will tend to follow paths where previous message were successfully routed to the sink.

2.2.1 Algorithm

We now present our algorithm. We begin by some definitions and notations, and follow with an explanation of our assumptions. We then present the algorithm. The formal code of our routing protocol, which is referenced during our explanations, is given in Algorithms 1 and 2.

2.2.2 Assumptions and Notations

We consider arbitrary connected networks with bidirectional links, although we focus on Unit Disk Graphs (UDG) when doing simulations. Each node p has a unique identifier (to simplify, we shall identify any node with its identifier, whenever convenient) and knows the set of its neighbors, $\mathcal{N}eig_p$.

Algorithm 1 SR3 for any source node v

```
Input: $k_{vs}$: the key of node $v$, shared with the sink $s$
Variables:
  $L_{Sent}$: List of at most $s_S$ pairs, initially empty
  $L_{AckRouting}$: List of at most $s_A$ pairs, initially empty
  $L_{Reputation}$: List of at most $s_R$ elements, initially empty
On generation of $Data$
 1: $N_v \leftarrow$ NEW_NONCE()
 2: $h \leftarrow \mathrm{H}(N_v)$
 3: $C \leftarrow \mathrm{E}_{k_{vs}}(\langle Data, N_v \rangle)$
 4: $next \leftarrow \textsc{Rand}(\mathcal{N}eig_v, \mathcal{L}^v_{SR3}(L_{Reputation}))$
 5: $L_{Sent} \leftarrow L_{Sent} \odot \langle N_v, next \rangle$
 6: Send $\langle \texttt{MSG}, C, h, v \rangle$ to $next$
On reception of $\langle \texttt{MSG}, C, h, o \rangle$ from $f$
 7: $next \leftarrow \textsc{Rand}(\mathcal{N}eig_v, \mathcal{L}^v_{SR3}(L_{Reputation}))$
 8: if $v = o$ then
 9:     $\langle Data, N_o \rangle \leftarrow \mathrm{E}^{-1}_{k_{vs}}(C)$
10:     if $\mathrm{H}(N_o) = h$ then
11:         $L_{Sent} \leftarrow L_{Sent} \odot \langle N_o, next \rangle$
12:         Send $\langle \texttt{MSG}, C, h, o \rangle$ to $next$
13:     end if
14: else
15:     if $\langle h, \_ \rangle \notin L_{AckRouting}$ then
16:         $L_{AckRouting} \leftarrow L_{AckRouting} \bullet \langle h, f \rangle$
17:     end if
18:     Send $\langle \texttt{MSG}, C, h, o \rangle$ to $next$
19: end if
On reception of $\langle \texttt{ACK}, N_o, o \rangle$ from $f$
20: if $v = o \wedge \langle N_o, \_ \rangle \in L_{Sent}$ then
21:     $first\_hop \leftarrow \textsc{Get}(L_{Sent}, N_o)$
22:     $L_{Reputation} \leftarrow L_{Reputation} \bullet first\_hop$
23:     $L_{Sent} \leftarrow L_{Sent} \setminus \langle N_o, \_ \rangle$
24: else
25:     if $v \neq o$ then
26:         $h \leftarrow \textsc{Hash}(N_o)$
27:         if $\langle h, \_ \rangle \in L_{AckRouting}$ then
28:             $next \leftarrow \textsc{Get}(L_{AckRouting}, h)$
29:             $L_{AckRouting} \leftarrow L_{AckRouting} \setminus \langle h, \_ \rangle$
30:         else
31:             $next \leftarrow \textsc{Rand}(\mathcal{N}eig_v, \mathcal{L}^v_{RW})$
32:         end if
33:         Send $\langle \texttt{ACK}, N_o, o \rangle$ to $next$ with probability $\frac{N-1}{N}$
34:     end if
35: end if
```

Algorithm 2 SR3 for the sink s

```
Input: $keys[]$: array of shared keys, indexed on node identifiers
On reception of $\langle \texttt{MSG}, C, h, o \rangle$ from $f$
36: $\langle Data, N_o \rangle \leftarrow \mathrm{E}^{-1}_{keys[o]}(C)$
37: if $\mathrm{H}(N_o) = h$ then
38:     Deliver $Data$ to the application
39:     Send $\langle \texttt{ACK}, N_o, o \rangle$ to $f$
40: end if
On reception of $\langle \texttt{ACK}, N_o, o \rangle$ from $f$
41: $next \leftarrow \textsc{Rand}(\mathcal{N}eig_s, \mathcal{L}^s_{RW})$
42: Send $\langle \texttt{ACK}, N_o, o \rangle$ to $next$ with probability $\frac{N-1}{N}$
```

Networks are made of one sink (named s), which is the data collector, and numerous source nodes. The source nodes are sensors, and consequently are limited in terms of memory, computational power, and battery. Sensors are non-trustworthy since they are vulnerable to physical attacks and an adversary can compromise them. In contrast, the sink is assumed to be robust and powerful in terms of memory, computation, and energy. So, we assume that it cannot be compromised. We also assume that all source nodes have several data to route; however, the scheduling of the data generation is *a priori* unknown. There is no time synchronization between nodes.

All nodes have access to a lightweight cryptography library. We need several cryptographic primitives:

- A hash function: for an input i, we denote by $\mathrm{H}(i)$ the output of that function.
- Symmetric encryption: for a plaintext pt and a key k, we denote by $\mathrm{E}_k(pt)$ the encryption operation. The decryption operation for a ciphertext ct is denoted $\mathrm{E}_k^{-1}(ct)$.
- Nonce generation: to generate a fresh nonce, *i.e.* an unpredictable random number, we call the function $\textsc{New\_Nonce}()$.

Furthermore, each source node shares a symmetric key with the sink: for a node v, we denote that key k_{vs}, and we suppose the sink holds an array of all those keys called $keys$ (with $keys[v] = k_{vs}$).

Through the algorithm, we use finite-size lists with two different insertion operations, $\odot$ and $\bullet$. The addition of $\langle x, y \rangle$ to the list L using $\odot$ works as follows: first, if L contains any pair with a left member equal to x, that pair is removed from L; then, if L is (still) full, the rightmost pair is removed; finally, $\langle x, y \rangle$ is inserted on the left side of the list. Note that, using $\odot$, any left member of a pair in the list is unique. The other insertion operator is denoted $\bullet$, and works as a FIFO insertion: if the list is full, then the oldest element is removed from the list, and finally the new element is added. Note that there may be several occurrences of the same element in a list updated with $\bullet$.

2.2.3 Reputation Mechanism

To implement our reputation mechanism, we identify each data message (tagged `MSG` in the algorithm) with a nonce that should remain secret between the source and sink until the delivery of the data message. This nonce is hashed to give the message identifier, allowing any node to identify which message an acknowledgment corresponds to. Our complete message format is the following for a sender v: $\langle \texttt{MSG}, \mathrm{E}_{k_{vs}}(\langle Data, N_v\rangle), \mathrm{H}(N_v), src\rangle$, and the corresponding acknowledgment is $\langle \texttt{ACK}, N, v\rangle$.

Assume that node v initiates the routing of some value $Data$. It first generates a nonce N_v (New_Nonce(), Line 1). Then, it encrypts in a ciphertext C the concatenation of $Data$ and N_v using the key k_{vs} it shares with the sink ($\mathrm{E}_{k_{vs}}(\langle Data, N_v\rangle)$, Line 3). Then, both C and the identifier of v (in plaintext) are routed to the sink, and only the sink is able to decrypt C. So, upon receiving the data packet, the sink decrypts C using k_{vs}, delivers $Data$, and sends back to v an acknowledgment `ACK` containing N_v (Lines 36-39). Finally, if v receives this acknowledgment, it has the guarantee that $Data$ has been delivered, thanks to N_v.

Now, during the routing, a compromised relay node can blindly modify the encrypted part of the message. To prevent the sink from delivering erroneous data, we add a hash of the nonce into the data message ($\mathrm{H}(N_v)$, Line 2). This way, when receiving a message $\langle \texttt{MSG}, C, h, v\rangle$, the sink can check the integrity of the message by first decrypting C using the key $k_{vs} = keys[v]$ ($\mathrm{E}^{-1}_{keys[v]}(C)$, Line 36), and then comparing the hash of the nonce in C to the message field h: if they do not match, the message is simply discarded. Similarly, if a compromised node has modified the plaintext identifier in the message, then the sink will decrypt C with a wrong key, and therefore the hash of the decrypted nonce will not match h.

The whole process is illustrated in Figure 2.1.

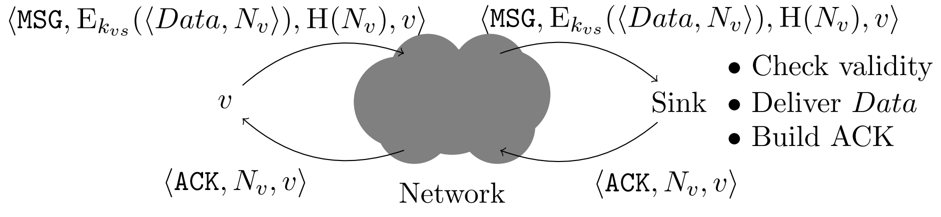


Figure 2.1: A message and its acknowledgment.

Upon receiving an acknowledgment, if the receiving node v is the initiator of the corresponding data message m, v can conclude that m has been delivered. In that case, v should reinforce the probability associated to the neighbor to which it previously sent m. To achieve that, we proceed as follows: when v initiates the routing of m, v saves in the list L_{Sent} the nonce stored in m, together with the identifier of the neighbor to which v sends m (L_{Sent} is appended in Line 5 in using $\odot$, which we described before). Hence, on reception of an acknowledgment, v checks (in Line 20) if it is the destination of the acknowledgment and if the nonce N_o attached to that acknowledgment appears in L_{Sent} (see the test $\langle N_o, \_ \rangle \in L_{Sent}$ in Line 20).[1] In that case, v gets back the corresponding neighbor from the list (GET(L_{Sent}, N_o), Line 21), increases its confidence on that neighbor (the mechanism is discussed in a latter section), and removes the record from L_{Sent} ($L_{Sent} \setminus \langle N_o, \_ \rangle$, Line 23). If v is the destination of the acknowledgment, but N_o does not appear in L_{Sent}, the acknowledgment is simply discarded.

Due to the memory limitations, L_{Sent} must have a maximum size, s_Q. If a node v has some new data to route and L_{Sent} is full (that is, it contains s_Q elements), then the oldest element is removed from the list to make room for the new one. A side effect is that records about lost messages or of messages whose acknowledgment has been lost are eventually removed from L_{Sent}.

Note that it may happen that some data message m comes back to the node v from which it originates because m followed a cycle in the network. In this case (Lines 8-13), the validity of m is checked, and then the routing process of m is restarted. Since the old entry in L_{Sent} is not considered to be relevant anymore, it is simply replaced by the new one.

2.2.4 Compute the Reputation

To choose the next hop of some data message, a node performs a random choice among its neighbors, weighted according to their reputation (see Lines 4 and 7).

The reputation of a neighbor actually corresponds to the number of occurrences of its identifier in the list $L_{Reputation}$: each time a node v wants to reinforce the reputation of some neighbor u, it simply adds an occurrence of u into its list (Line 22).

Our reputation mechanism is implemented using the probability law denoted by $\mathcal{L}^v_{SR3}(L_{Reputation})$: Let X be a random variable taking value in $\mathcal{N}eig_v$. The probability law $\mathcal{L}^v_{SR3}(L_{Reputation})$ is defined, $\forall x \in \mathcal{N}eig_v$, by:

$$Pr(X = x) = \frac{|L_{Reputation}|_x + \delta_v^{-1}}{|L_{Reputation}| + 1}$$

Where δ_v is the degree of v, $|L_{Reputation}|$ is the number of elements in $L_{Reputation}$, and $|L_{Reputation}|_x$ is the number of occurrences of x in $L_{Reputation}$. Hence, when v wants to route a data message, it chooses its next destination according to $\mathcal{L}^v_{SR3}(L_{Reputation})$ (see RAND($\mathcal{N}eig_v$,$\mathcal{L}^v_{SR3}(L_{Reputation})$) in Lines 4 and 7).

Informally, when a node needs to route a message, it draws at random a value from $L_{Reputation}$ plus a blank element. If the blank element is drawn, it selects a neighbor uniformly at random, and sends the message to that neighbor. Otherwise, the message is sent to the neighbor whose identifier has been drawn. This way, the more a neighbor is trusted, the more it will be selected. However, because of the blank element, there is always a positive probability of selecting a neighbor without taking trust into account. Note that, initially $L_{Reputation}$ is empty, and consequently the first selections are made uniformly at random.

[1] "_" means "any value". So, $\langle N_o, \_ \rangle$ is any record whose left value is N_o.

To ensure a better resiliency against attackers that change their behavior over time, and to reduce memory consumption, $L_{Reputation}$ is defined as a FIFO list of maximum size, s_R. The insertions in $L_{Reputation}$ use the operator $\bullet$ that satisfies the following condition: when the list is full, the next insertion is preceded by the removing of the oldest (and consequently, less relevant) element.

An example illustrating the probability law is provided in Figure 2.2. In this example, we focus on the node v and assume a $L_{Reputation}$ of at most 3 elements. From the given configuration, v will route most messages through z, because it has the greatest number of occurrences in the list $L_{Reputation}$ of v.

Using such a FIFO finite list, a node only stores the freshest information. Interestingly, if a compromised node first behaves well, its reputation increases, resulting in attracting the traffic. Then, it may change its behavior to become a blackhole (a node dropping all messages it receives). Now, thanks to our mechanism, regularly some messages will be routed via other nodes and consequently the reputation of the compromised node will gradually decrease, inducting then a severe reduction of the traffic going through that node.

Consider again Figure 2.2. If z turns out to be compromised and starts dropping all messages, then all messages going through z and w will either get lost or loop back to v. However, there is still a positive probability that v routes messages through y, which will retransmit them. Some of these messages will be delivered and consequently acknowledged. So, the identifiers currently stored in $L_{Reputation}$ will be progressively replaced by occurrences of y, increasing its probability (resp. decreasing the probability of w and z) of being chosen.

$$v\text{'s } L_{Reputation},\ s_R = 3 \quad [w,\ z,\ z]$$

Next hop probabilities for v:

$$P(X = w) = \frac{1 + 1/3}{4} = \frac{4}{12} \approx 33.33\%$$

$$P(X = z) = \frac{2 + 1/3}{4} = \frac{7}{12} \approx 58.33\%$$

$$P(X = y) = \frac{0 + 1/3}{4} = \frac{1}{12} \approx 8.33\%$$

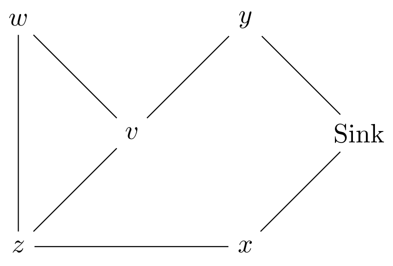


Figure 2.2: An example of how the reputation affects the routing process

2.2.5 Acknowledgment Routing

Let *ack* be an acknowledgment message. Since *ack* has been emitted because the corresponding data message m has been successfully delivered by the sink, we can suppose that the path followed by m was safe. Therefore, we can use the bidirectionality of the links to route *ack* (as much as possible) through the reverse path followed by m.

This reverse routing is accomplished by letting a trail along the path followed by m. This trail is stored thanks to the list $L_{AckRouting}$ maintained at each node: after the reception of each data message, the relaying nodes store the hash of the nonce available in the message, together with the identifier of the neighbor from which they received the message (Lines 14-17). This information will be then used during the return trip of the acknowledgment: when a node v receives an acknowledgment containing the nonce N_x, it checks whether it is the final destination of that acknowledgment (Lines 20 and 25). If this is not the case, v checks if an entry containing $\mathrm{H}(N_x)$ exists in $L_{AckRouting}$ (Lines 26-30). If v finds such an entry, it sends the acknowledgment to the corresponding neighbor and removes the entry from $L_{AckRouting}$ (Line 29). Otherwise, the next hop of the acknowledgment is chosen uniformly at random ($\mathcal{L}^v_{RW}$ denotes the probability law of the uniform random walk, see Line 31).

If a data message loops back to a node it already visited, the most relevant information regarding acknowledgments for this node is the oldest one. Therefore, before inserting a new trail, the node checks if $L_{AckRouting}$ already contains a trail for that message. If a related entry exists, we do not update $L_{AckRouting}$ (Lines 14-17).

Acknowledgments can be still dropped by compromised nodes. The trail for such lost acknowledgments would unnecessarily clutter the memory of nodes. To avoid this, we manage $L_{AckRouting}$ similarly to $L_{Reputation}$, *i.e.*, $L_{AckRouting}$ is a list of bounded size s_A, appended using operator $\bullet$.

Finally, an intruder may build acknowledgments with false nonces. These fake acknowledgments will increase the load of the network, and impact the energy consumption. Now, some nodes being compromised, a safe node cannot trust information coming from its neighbors to decide whether it should forward or drop an acknowledgment. To circumvent that problem, a relay node decides to drop a received acknowledgment with probability $\frac{1}{N}$, where N is an upper bound on the number of nodes (Lines 33 and 42). So, on the average, an acknowledgment makes N hops in the network before being dropped. An interesting side effect of this method is the following: in a safe network (*i.e.*, a network without attackers), the acknowledgments that follow long routes are often dropped before reaching their final destination. Since the length of the routes followed by the acknowledgments are directly related to the length of the route taken by the corresponding messages, the reputation mechanism ends up favoring shorter routes, thus improving the overall hops complexity.

2.3 Cryptographic Proof of the Security Properties

We evaluate the security of SR3 in two phases. The first phase focuses on the packet format, for which we prove the following three properties: unforgeability, confidentiality of the data, and confidentiality of the nonce before packet delivery. This analysis uses the tool *CryptoVerif* [Bla08]. We first detail the modeling of SR3 and our intruder model. Then, we model the different security properties we considered. CryptoVerif automatically finds bounds on the security of these properties. These bounds allow the user to determine the desired trade-off between message sizes and the expected security level.

The second phase is a symbolic analysis of the protocol, in order to prove its security when running several sessions. This analysis supposes that the cryptographic primitives are perfect. For this, we use the tool *Scyther* [Cre08], which we chose among other symbolic analysis tools for its speed (see [CLN09]) and ease of use. We first describe how we model SR3 and the attacker for these two analysis.

Notations

When describing a game, we write $a \overset{\$}{\leftarrow} X$ to denote that a is a random value obtained according the distribution represented by X. If X is a set, a is drawn at random using the uniform law on X. Similarly, if X is a probabilistic algorithm, a is drawn at random using the algorithm.

We denote by $\mathrm{E}_{k_{src}}(x)$ the result of the encryption of x, using the block cipher with the key of *src* and by $H(x)$ the hash of x.

We recall that stating that a function $\mu(x) : \mathbb{N} \to \mathbb{R}$ is negligible in x means that for every positive polynomial P there exists an integer I such that for all $x > I$, $\mu(x) < |\frac{1}{P(x)}|$, as presented in [Bel02].

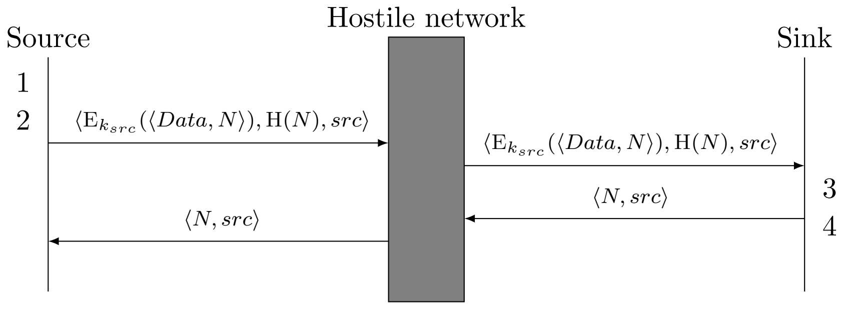
2.3.1 Modeling of SR3

SR3, as described in the previous section, routes messages through several nodes. There are three distinct roles in this process: the *source* (whose identifier is denoted here *src*) of the considered data message, the *relays*, and the *sink*. The data message is initially created by the source, which then forwards it either to a relay or the sink. Relays forward the data message without changing it, and the sink delivers it. If the received data message is valid, the sink generates an acknowledgment $\langle \mathtt{ACK}, N, src \rangle$, that is routed through relays until it reaches the source. A data message can loop back to its source, and an acknowledgment can loop back to the sink: they act as relays in these cases.

When a message arrives to the sink, it first checks whether the packet respects the format $\langle \mathtt{MSG}, C, h, s \rangle$. If so, the data message is valid when h is equal to the hash of the right part of C, deciphered with the key of the node identified by s.

The honest relays do not alter the messages in any way, and the protocol works with any number of them, or none at all. Also, our intruder model specifies that any node can be compromised, except the sink. Therefore, all relays R_i are suspicious, and we lump them, whether honest or compromised, together in one single entity, called the *hostile network*. All communications between the source and the sink happen through the hostile network, as depicted in Figure 2.3. The hostile network can modify or drop messages, and can also create messages using information deduced from previous communications. Finally, as the types `MSG` and `ACK` can be deduced from the message format, we omit them in our modeling.

The model is summarized in Figure 2.3. Next, we use it to analysis of SR3.


Source
Hostile network
Sink
1
2
3
4
$\langle \mathrm{E}_{k_{src}}(\langle Data, N\rangle), \mathrm{H}(N), src\rangle$
$\langle \mathrm{E}_{k_{src}}(\langle Data, N\rangle), \mathrm{H}(N), src\rangle$
$\langle N, src\rangle$
$\langle N, src\rangle$

1. Generate *Data*, 2. Draw a nonce N,

3. Check validity, 4. Deliver *Data*

Figure 2.3: Modeling of one session of SR3

2.3.2 Game-based Proof of the Security Properties

For our analysis, we use a common cryptography model, called the *random oracle model* [BR93], to formalize hash functions and encryptions as pseudo-random functions. This model uses the concept of *random oracle* as follows: for every input i, the first time the random oracle is queried with i, it returns a value v, picked uniformly at random from its output domain; then, each time it is queried again with i, it returns the same value v.

We evaluate whether our protocol provides specific security properties using *games*. A game is a probabilistic algorithm where an adversary, given as a probabilistic polynomial-time Turing machine, faces a challenge linked to a specific property. The goal is to quantify the ability of the adversary to win the challenge. This is called the *advantage* of the adversary. The advantage of an adversary $\mathcal{A}$ is often computed as the difference between the probability of $\mathcal{A}$ to win the game minus the probability of $\mathcal{A}$ to win the game against an idealized version of the protocol.

Modeling SR3's Primitives

Hash function Our algorithm uses a hash function of input size η_n and of output size η_h. We model it as a random oracle, and we refer to this modeling using $\mathcal{H} : \{0,1\}^{\eta_n} \to \{0,1\}^{\eta_h}$. The number of times the adversary calls $\mathcal{H}$ is denoted q_H.

Nonces Nonces are modeled as random numbers of size η_n.

Block Cipher We assume that the block cipher is *PRP-CCA-secure*.[2] This property was introduced in [LR88] and is defined using a game as shown below.

[2] *PRP-CCA* means *PseudoRandom Permutation under Chosen Ciphertext Attack.*

Let $\mathcal{K}(\eta_k)$ be a set of keys (with η_k the size of keys), η_c the size of a block, and let $F : \mathcal{K}(\eta_k) \times \{0,1\}^{\eta_c} \to \{0,1\}^{\eta_c}$ a family of permutations. For all keys k in $\mathcal{K}(\eta_k)$, F_k is a permutation of $\{0,1\}^{\eta_c}$ (that is, a bijection from $\{0, ..., 2^{\eta_c} - 1\}$ to $\{0, ..., 2^{\eta_c} - 1\}$), and F_k^{-1} its inverse. Also, we denote by $Perm$, the set of all possible pairs of a permutation of $\{0,1\}^{\eta_c}$ and their inverses.

Given a permutation family F, the *PRP-CCA* game works as follows. Intuitively, an adversary $\mathcal{A}$ should guess whether an oracle $\mathcal{O}$ is a permutation extracted from F or a random permutation; the guess of $\mathcal{A}$ about $\mathcal{O}$ is noted $b \xleftarrow{\$} \mathcal{A}^{\mathcal{O},\mathcal{O}^{-1}}()$. This game is described using two experiments shown in Figure 2.4.

```
$\mathbf{Expt}_F^{PRP\text{–}CCA-1}(\mathcal{A}, \eta_k)$ :
$k_{src} \xleftarrow{\$} \mathcal{K}(\eta_k)$
$(\mathcal{O}, \mathcal{O}^{-1}) \leftarrow (F_{k_{src}}, F_{k_{src}}^{-1})$
$b \xleftarrow{\$} \mathcal{A}^{\mathcal{O},\mathcal{O}^{-1}}()$
**Return** $b$
```

```
$\mathbf{Expt}_F^{PRP\text{–}CCA-0}(\mathcal{A}, \eta_k)$ :
$(\mathcal{O}, \mathcal{O}^{-1}) \xleftarrow{\$} Perm$
$b \xleftarrow{\$} \mathcal{A}^{\mathcal{O},\mathcal{O}^{-1}}()$
**Return** $b$
```

Figure 2.4: The PRP-CCA experiments

The advantage of $\mathcal{A}$ in this game is noted $\mathbf{Adv}_F^{PRP\text{–}CCA}(\mathcal{A}, \eta_k)$, and defined as follows:

$$\mathbf{Adv}_F^{PRP\text{–}CCA}(\mathcal{A}, \eta_k) = Pr[\mathbf{Expt}_F^{PRP\text{–}CCA-1}(\mathcal{A}, \eta_k) = 1] - Pr[\mathbf{Expt}_F^{PRP\text{–}CCA-0}(\mathcal{A}, \eta_k) = 1]$$

From the adversary $\mathcal{A}$'s point of view, the PRP-CCA game consists in guessing whether an oracle $\mathcal{O}$ is a permutation sampled from F (in which case it should return 1) or a random permutation from $Perm$ (it should return 0). To do this, $\mathcal{A}$ is allowed access to both $\mathcal{O}$ and $\mathcal{O}^{-1}$ for a bounded total number of queries.

The game *PRP-CPA* (for *Chosen Plaintext Attack*) is similar to *PRP-CCA*, except that the adversary only has access to $\mathcal{O}$. The advantage of $\mathcal{A}$ in this game is noted $\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{A}, \eta_k)$. Note that *PRP-CPA* is weaker that *PRP-CCA*, *i.e.*, for every adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$, using the same resources as $\mathcal{A}$, such that $\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{A}, \eta_k) \le \mathbf{Adv}_F^{PRP\text{–}CCA}(\mathcal{B}, \eta_k)$.

As we assume that the block cipher of SR3 is *PRP-CCA-secure*, by definition, both $\mathbf{Adv}_F^{PRP\text{–}CCA}(\mathcal{A}, \eta_k)$ and $\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{A}, \eta_k)$ are *negligible* in the size of the key η_k.

Modeling the Protocol

Based on the above presented model, we describe actions performed by the source and sink using several phases. The attacker has access to some of these, depending on the game. For instance, the function *Gen* below is a model of the creation of a new message by an honest node. An attacker having access to this function is in a situation analog to a chosen-plaintext attack.

- $k_{src} \overset{\$}{\leftarrow} \mathcal{K}(\eta_k)$ denotes the initialization of SR3 on a node, that is, simply selecting the symmetric key used by this node. $\mathcal{K}(\eta_k)$ is the space of all the keys of length η_k used for the symmetric cipher. This is done before the WSN is deployed, and so we suppose k_{src} is known only by src and the sink.

- $Gen_{src}^{\mathcal{O}(\cdot)}(Data)$ is the function which generates a packet produced by src containing $Data$ (this packet has length η_d), using the oracle $\mathcal{O}$ as an encryption function from $\{0,1\}^{\eta_c}$ to $\{0,1\}^{\eta_c}$ (where $\eta_c = \eta_d + \eta_n$), and returns the packet and its corresponding nonce. This packet is made of $\langle C, h, src\rangle = \langle \mathcal{O}(\langle Data, N\rangle), \mathcal{H}(N), src\rangle$, where N is a fresh unpredictable nonce of size η_n, and $\mathcal{O}(\langle Data, N\rangle)$ is the encryption of the data, concatenated to that nonce. The function $Gen_{src}^{\mathcal{O}(\cdot)}(Data)$ returns the pair $\langle C, h, src\rangle, N$. The nonce is given in cleartext to represent the knowledge of an attacker that listens to traffic in the network. Indeed, such an attacker may have access to both the messages and their acknowledgments; he may then know the nonces contained in the original messages. However, depending on the game, the attacker may be provided only $\langle C, h, src\rangle$ as a challenge, to model the fact that the attack should happen before the message reaches the sink. We denote by q_G the total number of calls made to Gen by the adversary, and we store all the returned messages in a set called $Queries$, initially empty.

- $Verif^{\mathcal{O}^{-1}}(\langle C, h, s\rangle)$ is the function that checks whether the packet is considered valid or not, using both the hash function $\mathcal{H}$ and the inverse of the oracle $\mathcal{O}$ used as the block cipher. It verifies:

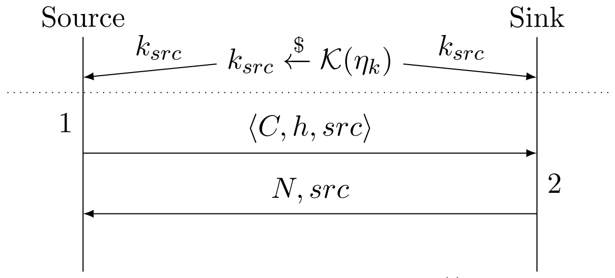
  - Whether the fields are the right sizes: $C \in \{0,1\}^{\eta_c}$, and $h \in \{0,1\}^{\eta_h}$,
  - Whether the message is from the right node: s is equal to src,
  - Whether the encrypted nonce and the hash contained in the message match, *i.e.*: $\mathcal{H}(Right(\mathcal{O}^{-1}(C))) = h$, where $Right$ is he function that give the right part of the pair produced by the oracle.

  If all of these three conditions are satisfied, then the function outputs 1 (meaning that the packet is valid), and otherwise it outputs 0. We denote q_V the number of calls made by the adversary to $Verif$.

These three functionnalities allow us to model the SR3 protocol, as shown in Figure 2.5.

2.3.3 The Properties

Informally, the three properties we want to prove are the following:


Source
Sink
k_{src}
$k_{src} \xleftarrow{\$} \mathcal{K}(\eta_k)$
k_{src}
1
$\langle C, h, src\rangle$
N, src
2

1. $(\langle C, h, src\rangle, N) = Gen_{src}^{E_{k_{src}}(\cdot)}(Data)$
2. Packet verification phase: $Verif^{E^{-1}_{k_{src}}(\cdot)}(\langle C, h, src\rangle)$

Figure 2.5: The SR3 protocol, using functions

- Confidentiality of the data: the probability of the adversary getting information about the data in a message is negligible, even when the acknowledgment has been sent.
- Confidentiality of the nonce: the adversary has a negligible probability of guessing the nonce N contained in a challenge message, before its delivery.
- Unforgeability of the messages: the probability that the adversary creates a new message m such that $Verif^{E^{-1}_{k_{src}}(\cdot)}(m) = 1$ is negligible.

With the help of CryptoVerif, we analyzed those three properties of SR3. Each of these three properties is evaluated thanks to a game. For each game, CryptoVerif outputs a bound on the advantage of any adversary in that game. This bound is obtained automatically after successive game reductions. The complete verification code is available online [ADJL13c] and in Appendix A.

Data Confidentiality

The first property we consider is the confidentiality of the data. The game (named FG, for *Find-then-Guess*) is based on the idea that even if the adversary chooses the set of possible data, it cannot guess which of those data is inside a given packet. On the other hand, if the attacker were able to win reliably, it would also effectively be able to recover some information about the data contained in messages, without knowledge of the key.

Let $\mathcal{A}$ be an adversary running in two phases: $\mathcal{A}_1$ and $\mathcal{A}_2$. First, $\mathcal{A}_1$ outputs two data, $Data_0$ and $Data_1$, together with some information *state* used to link the two attacker phases together. One of these two data is selected uniformly at random, and a `MSG` message $\langle C, h, src\rangle$ is generated using the selected data. Then, $\mathcal{A}_2$ is given the message $\langle C, h, src\rangle$, the nonce N this message contains, and *state*. To win, $\mathcal{A}_2$ should guess which of $Data_0$ and $Data_1$ is contained in $\langle C, h, src\rangle$. During this game, $\mathcal{A}$ can query $Gen_{src}^{\mathcal{O}(\cdot)}$ and $\mathcal{H}$.

```
$Experiment\ \mathbf{Expt}_F^{FG}(\mathcal{A}, \eta_k)$ :
    $k_{src} \xleftarrow{\$} \mathcal{K}(\eta_k)$
    $(\mathcal{O}, \mathcal{O}^{-1}) \leftarrow (F_{k_{src}}, F^{-1}_{k_{src}})$
    $(Data_0, Data_1, state) \xleftarrow{\$} \mathcal{A}_1^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}()$
    $b \xleftarrow{\$} \{0, 1\}$
    $(\langle C, h, src\rangle, N) \xleftarrow{\$} Gen_{src}^{\mathcal{O}(\cdot)}(Data_b)$
    If $(b = \mathcal{A}_2^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}(\langle C, h, src\rangle, N, state))$
        Return 1
    Else
        Return 0
```

We define the find-then-guess advantage of $\mathcal{A}$ against F as the probability of $\mathcal{A}$ winning the game (*i.e.* $Pr[\mathbf{Expt}_F^{FG}(\mathcal{A}, \eta_k) = 1]$) minus the probability of winning for an adversary that outputs a random bit:

$$\mathbf{Adv}_F^{FG}(\mathcal{A}, \eta_k) = Pr[\mathbf{Expt}_F^{FG}(\mathcal{A}, \eta_k) = 1] - \frac{1}{2}$$

We then model this game in CryptoVerif, so that it outputs a bound on the advantage of an adversary. This bound depends on the ability of that attacker to break the block cipher, *i.e.*, this game can be reduced to the PRP-CPA game on the block cipher used by SR3.

The Find-then-guess proof outline is as follows. We first show the full initial experiment in Figure 2.6a. Recall that we assume the block cipher is a PRP-CCA secure function: therefore, the probability of an attacker being able to distinguish the initial experiment $\mathbf{Expt}_F^{FG}$ (described in Figure 2.6a) from another experiment $\mathbf{IntermExpt}_F^A$ (Figure 2.6b) that uses a random permutation instead of the block cipher is related to the advantage $\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k)$ an arbitrary adversary $\mathcal{B}$ can have. Now, as $\mathbf{IntermExpt}_F^A$ uses a random permutation, we can separate the cases where the output of that random permutation in the *Gen* oracle is a fresh random value, or where it is the repetition of an previous value. Since we know the probability of nonce collisions between the outputs of the *Gen* oracle, we know the probability of an attacker being able to distinguish between that experiment $\mathbf{IntermExpt}_F^A$, and an experiment $\mathbf{IntermExpt}_F^B$ (Figure 2.6c) where all encryptions are replaced by random values and $\mathcal{O}$ is not used. In this last experiment, the attacker is never given anything that depends on the value of b, and so it is clearly impossible for an attacker to distinguish the value of b. The probability of an attacker being right in the experiment $\mathbf{IntermExpt}_F^B$ is therefore $0, 5$, and so, we know that the probability of an attacker being right in $\mathbf{Expt}_F^{FG}$ is bounded

```
$\textit{Oracle } Gen_{src}^{\mathcal{O}(\cdot)}(Data):$
    $N \xleftarrow{\$} \{0,1\}^{\eta_n}$
    $\textbf{Return } \langle \mathcal{O}(\langle Data, N\rangle), \mathcal{H}(N), src\rangle, N$
$\textit{Experiment } \textbf{Expt}_F^{FG}(\mathcal{A}, \eta_k):$
    $k_{src} \xleftarrow{\$} \mathcal{K}(\eta_k)$
    $(\mathcal{O}, \mathcal{O}^{-1}) \leftarrow (F_{k_{src}}, F_{k_{src}}^{-1})$
    $(Data_0, Data_1, st) \xleftarrow{\$} \mathcal{A}_1^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}()$
    $b \xleftarrow{\$} \{0,1\}$
    $(\langle C, h, src\rangle, N) \xleftarrow{\$} Gen_{src}^{\mathcal{O}(\cdot)}(Data_b)$
    If $(b = \mathcal{A}_2^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}(\langle C, h, src\rangle, N, st))$
      $\textbf{Return } 1$
    Else
      $\textbf{Return } 0$
```

(a) The original $\textbf{Expt}_F^{FG}$ experiment

```
$\textit{Oracle } Gen_{src}^{\mathcal{O}(\cdot)}(Data):$
    $N \xleftarrow{\$} \{0,1\}^{\eta_n}$
    $\textbf{Return } \langle \mathcal{O}(\langle Data, N\rangle), \mathcal{H}(N), src\rangle, N$
$\textit{Experiment } \textbf{IntermExpt}_F^{A}(\mathcal{A}, \eta_k):$
    $(\mathcal{O}, \mathcal{O}^{-1}) \xleftarrow{\$} Perm$
    $(Data_0, Data_1, st) \xleftarrow{\$} \mathcal{A}_1^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}()$
    $b \xleftarrow{\$} \{0,1\}$
    $(\langle C, h, src\rangle, N) \xleftarrow{\$} Gen_{src}^{\mathcal{O}(\cdot)}(Data_b)$
    If $(b = \mathcal{A}_2^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}(\langle C, h, src\rangle, N, st))$
      $\textbf{Return } 1$
    Else
      $\textbf{Return } 0$
```

(b) The $\textbf{IntermExpt}_F^{A}$ experiment

```
$\textit{Oracle } Gen_{src}^{\mathcal{O}(\cdot)}(Data):$
    $N \xleftarrow{\$} \{0,1\}^{\eta_n}; C \xleftarrow{\$} \{0,1\}^{\eta_c}$
    $\textbf{Return } \langle C, \mathcal{H}(N), src\rangle, N$
$\textit{Experiment } \textbf{IntermExpt}_F^{B}(\mathcal{A}, \eta_k):$
    $(\mathcal{O}, \mathcal{O}^{-1}) \xleftarrow{\$} Perm$
    $(Data_0, Data_1, st) \xleftarrow{\$} \mathcal{A}_1^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}()$
    $b \xleftarrow{\$} \{0,1\}$
    $(\langle C, h, src\rangle, N) \xleftarrow{\$} Gen_{src}^{\mathcal{O}(\cdot)}(Data_b)$
    If $(b = \mathcal{A}_2^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}(\langle C, h, src\rangle, N, st))$
      $\textbf{Return } 1$
    Else
      $\textbf{Return } 0$
```

(c) The $\textbf{IntermExpt}_F^{B}$ experiment

Figure 2.6: Intermediary experiments for the find-then-guess proof

by the ability for an attacker to distinguish between the initial experiment $\mathbf{Expt}_F^{FG}$ and the final one, $\mathbf{IntermExpt}_F^B$.

The bound on this game depends on the advantage $\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k)$ of any adversary $\mathcal{B}$ in a PRP-CPA game. Formally, for all adversaries $\mathcal{A}$ making q_G queries to *Gen* and q_H queries to the hash function, there exists an adversary $\mathcal{B}$ (making $q_G + 1$ queries to $\mathcal{O}$ in its game) such that:

$$\mathbf{Adv}_F^{FG}(\mathcal{A}, \eta_k) \leq \frac{2q_G^2 + 2q_G}{2^{\eta_c}} + \frac{2q_G^2 + 4q_G + 2}{2^{\eta_n}} + 2\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k)$$

Note that q_H does not appear in this bound. $\mathcal{H}$ is a random oracle whose outputs do not leak information about their corresponding inputs.

We assumed that the block cipher of SR3 is *PRP-CCA-secure*, and therefore, for all adversaries $\mathcal{A}$, $\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{A}, \eta_k)$ is negligible in η_k. As $\eta_c \geq \eta_n$, we find that this bound on $\mathbf{Adv}_F^{FG}(\mathcal{A}, \eta_k)$ is negligible in η_n and η_k.

This bound allows us to select the necessary trade-off between the desired level of security and the mandatory minimization of the message overhead. For instance, we can try to reach an advantage smaller than 2^{-50} for the confidentiality of the data, against an adversary $\mathcal{A}$ that can query each oracle up to 2^{20} times (around 1 million queries). To achieve this, we need to set η_n to 96 bits (12 bytes). This way, given data packets of 32 bits (4 bytes), we would have:

$$\begin{aligned}
\mathbf{Adv}_F^{FG}(\mathcal{A}, \eta_k) &\leq \frac{2 \times (2^{20})^2 + 2 \times 2^{20}}{2^{96+32}} + \frac{2 \times (2^{20})^2 + 4 \times 2^{20} + 2}{2^{96}} + 2\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k) \\
&= \frac{2^{41} + 2^{21}}{2^{128}} + \frac{2^{41} + 2^{22} + 2}{2^{96}} + 2\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k) \\
&\approx 2^{-54.999} + 2\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k) \\
\mathbf{Adv}_F^{FG}(\mathcal{A}, \eta_k) &\leq 2^{-54} + 2\mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k)
\end{aligned}$$

Finally, we need to choose a block cipher, as its quality will determine the PRP-advantage. If we use AES-128 as block cipher (which outputs 128 bits, using an input of 128 bits and a key of 128 bits), the best attack known to this day needs $2^{126.1}$ operations [BKR11]. Therefore, we can expect $\mathbf{Adv}_{AES-128}^{PRP\text{–}CPA}(\mathcal{B}, 128)$ to be much smaller than 2^{-54} for any $\mathcal{B}$, and consequently our security bound of 2^{-50} is satisfied, with an overhead of 12 bytes.

Nonce Confidentiality

In the next game, we evaluate whether an adversary can extract a nonce from an undelivered message. Let $\mathcal{A}$ be an adversary running in two phases ($\mathcal{A}_1$ and $\mathcal{A}_2$) that communicate using a variable named *state*. The game consists in giving a challenge `MSG` message $\langle C, h, src \rangle$ to an adversary $\mathcal{A}$, who should guess the nonce inside this message in nb_A tries. To do this, the adversary is allowed to call $Gen_{src}^{\mathcal{O}(\cdot)}$ and $\mathcal{H}$, and to choose the data that is going to be contained in the challenge message.

```
Experiment $\mathbf{Expt}_F^{N\text{–}conf}(\mathcal{A}, \eta_k)$ :
    $k_{src} \xleftarrow{\$} \mathcal{K}(\eta_k)$
    $(\mathcal{O}, \mathcal{O}^{-1}) \leftarrow (F_{k_{src}}, F_{k_{src}}^{-1})$
    $(Data, state) \xleftarrow{\$} \mathcal{A}_1^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}()$
    $(\langle C, h, src\rangle, N) \xleftarrow{\$} Gen_{src}^{\mathcal{O}(\cdot)}(Data)$
    $Answers \leftarrow \mathcal{A}_2^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot)}((\langle C, h, src\rangle, state))$
    If $(|Answers| \leq nb_A \wedge N \in Answers)$
        Return 1
    Else
        Return 0
```

The nonce confidentiality advantage of $\mathcal{A}$ against F is defined as the probability of winning the game ($Pr[\mathbf{Expt}_F^{N\text{–}conf}(\mathcal{A}, \eta_k) = 1]$) minus the probability of guessing the challenge nonce in nb_A random tries. This is a Bernoulli trial, and this probability is equal to the complementary probability of having all wrong guesses in less than nb_A identical and independent uniform random tries. Therefore, this is equal to $1 - (\frac{2^{\eta_n}-1}{2^{\eta_n}})^{nb_A}$. Replacing in context, we get:

$$\mathbf{Adv}_F^{N\text{–}conf}(\mathcal{A}, \eta_k) = Pr[\mathbf{Expt}_F^{N\text{–}conf}(\mathcal{A}, \eta_k) = 1] - \left(1 - \left(\frac{2^{\eta_n} - 1}{2^{\eta_n}}\right)^{nb_A}\right)$$

CryptoVerif outputs that for all adversaries $\mathcal{A}$ making q_G queries to Gen, q_V queries to $Verif$, and q_H queries to the hash function, there exists an adversary $\mathcal{B}$ making q_G+1 queries to $\mathcal{O}$ such that:

$$\mathbf{Adv}_F^{N\text{–}conf}(\mathcal{A}, \eta_k) \leq \frac{nb_A + q_H + q_H q_G + 2q_G + 2q_G^2}{2^{\eta_n}}$$
$$+\frac{q_G + q_G^2}{2^{\eta_c}} - \left(1 - \left(\frac{2^{\eta_n} - 1}{2^{\eta_n}}\right)^{nb_A}\right) + \mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k)$$

Similarly to the previous property, this bound becomes negligible when increasing η_n and η_k. Again, we can find the right values to obtain a given security bound. We use the same primitives as for the previous property: $\eta_n = 96$ bits (12 bytes), and data packets of 32 bits (4 bytes).

$$\mathbf{Adv}_F^{N\text{–}conf}(\mathcal{A}, \eta_k) \leq \frac{2^{20} + 2^{20} + 2^{20} \times 2^{20} + 2 \times 2^{20} + 2 \times (2^{40})^2}{2^{96}}$$
$$+\frac{2^{20} + (2^{20})^2}{2^{128}} - \left(1 - \left(\frac{2^{96} - 1}{2^{96}}\right)^{2^{20}}\right) + \mathbf{Adv}_F^{PRP\text{–}CPA}(\mathcal{B}, \eta_k)$$

$$= \frac{2^{20} + 2^{20} + 2^{40} + 2^{21} + 2^{41}}{2^{96}} + \frac{2^{20} + 2^{40}}{2^{128}} - \left(1 - \left(\frac{2^{96} - 1}{2^{96}}\right)^{2^{20}}\right) + \mathbf{Adv}_F^{\mathit{PRP–CPA}}(\mathcal{B}, \eta_k)$$

$$\approx 2^{-54.415} + \mathbf{Adv}_F^{\mathit{PRP–CPA}}(\mathcal{B}, \eta_k)$$

$$\mathbf{Adv}_F^{\mathit{N–conf}}(\mathcal{A}, \eta_k) \leq 2^{-54} + \mathbf{Adv}_F^{\mathit{PRP–CPA}}(\mathcal{B}, \eta_k)$$

We obtain results similar to the confidentiality results. Using again the AES-128 block cipher and the result in [BKR11], we can expect $\mathbf{Adv}_{AES-128}^{\mathit{PRP–CPA}}(\mathcal{B}, 128)$ to be much smaller than 2^{-54} for any $\mathcal{B}$, and our security bound of 2^{-50} is satisfied, with an overhead of 12 bytes.

Unforgeability

Finally, the last game evaluates the unforgeability of the messages, *i.e.*, the ability of an intruder to create a new valid message. Note that this property implies both *indistinguishability* and *authenticity* of the `MSG` messages. To evaluate this, we give to the attacker $\mathcal{A}$ access to both *Gen* and $Verif$. To win, $\mathcal{A}$ should return a packet which is valid and which has never been returned by *Gen* previously (recall that the *Queries* set contains the messages returned by *Gen*).

```
$\mathit{Experiment}\ \mathbf{Expt}_F^{\mathit{UF–CMVA}}(\mathcal{A}, \eta_k)$
    $\mathit{Queries} \leftarrow \emptyset$
    $k_{src} \xleftarrow{\$} \mathcal{K}(\eta_k)$
    $(\mathcal{O}, \mathcal{O}^{-1}) \leftarrow (F_{k_{src}}, F_{k_{src}}^{-1})$
    $\langle C, h, s\rangle \leftarrow \mathcal{A}^{Gen_{src}^{\mathcal{O}(\cdot)}(\cdot), \mathcal{H}(\cdot), Verif^{\mathcal{O}^{-1}(\cdot)}(\cdot)}()$
    If $(\langle C, h, s\rangle \notin \mathit{Queries} \wedge Verif^{\mathcal{O}^{-1}(\cdot)}(p))$
        Return 1
    Else
        Return 0
```

The unforgeability advantage of $\mathcal{A}$ against F is defined as the probability of $\mathcal{A}$ winning this game minus the probability of winning for an attacker that outputs a well-formed message $\langle C', h', src\rangle$ with C' and h' random values of size respectively η_c and η_h, without calling any oracle. Such a message is valid if and only if the hash of the nonce extracted from C' (which we call N) is equal to h' (a random value by hypothesis on the attacker). Since $\mathcal{H}$ is a random oracle, and the attacker did not previously query it, we know that $\mathcal{H}(N)$ is a random value of size η_h, and so we are looking for the probability of equality between two random values of size η_h, which is equal to $\frac{1}{2^{\eta_h}}$. Therefore:

$$\mathbf{Adv}_F^{\mathit{UF\text{–}CMVA}}(\mathcal{A}, \eta_k) = Pr[\mathbf{Expt}_F^{\mathit{UF\text{–}CMVA}}(\mathcal{A}, \eta_k) = 1] - \frac{1}{2^{\eta_h}}$$

Using CryptoVerif, we find that, similarly to the previous game, the advantage of $\mathcal{A}$ depends on the strength of the block cipher. Formally, for all adversaries $\mathcal{A}$ making q_G queries to Gen, q_V queries to $Verif$, and q_H queries to the hash function, there exists an adversary $\mathcal{B}$ making q_G queries to $\mathcal{O}$ and $q_V + 1$ queries to $\mathcal{O}^{-1}$ such that:

$$\mathbf{Adv}_F^{\mathit{UF\text{–}CMVA}}(\mathcal{A}, \eta_k) \leq \frac{q_H + q_H q_V + q_V q_G + q_G + q_H q_G + 2q_G^2}{2^{\eta_n}} +$$

$$\frac{q_G^2 + q_G + 2q_V q_G + q_V + q_V^2}{2^{\eta_c}} + \frac{q_V}{2^{\eta_h}} + \mathbf{Adv}_F^{\mathit{PRP\text{–}CCA}}(\mathcal{B}, \eta_k)$$

Similarly to the previous properties, this bound becomes negligible when η_n, η_k and η_h increase.

If we suppose that the adversary can make a million queries to each oracle, and that we want the same 2^{-50} security bound, we can choose the same sizes as for the confidentiality proof. However, this time, there are two major differences: the hash function size appears in the bound, and the game used for the block cipher is PRP-CCA instead of PRP-CPA. To achieve the security bound, we choose a hash function output size of 96 bits (12 bytes). With these settings, we get:

$$\mathbf{Adv}_F^{\mathit{UF\text{–}CMVA}}(\mathcal{A}, \eta_k) \leq \frac{2^{20} + 2^{20} \times 2^{20} + 2^{20} \times 2^{20} + 2^{20} + 2^{20} \times 2^{20} + 2 \times (2^{20})^2}{2^{96}} +$$

$$\frac{(2^{20})^2 + 2^{20} + 2 \times 2^{20} \times 2^{20} + 2^{20} + (2^{20})^2}{2^{96+32}} + \frac{2^{20}}{2^{96}} + \mathbf{Adv}_F^{\mathit{PRP\text{–}CCA}}(\mathcal{B}, \eta_k)$$

$$= \frac{2^{20} + 2^{40} + 2^{40} + 2^{20} + 2^{40} + 2^{41}}{2^{96}} +$$

$$\frac{2^{40} + 2^{20} + 2^{41} + 2^{20} + 2^{40}}{2^{128}} + \frac{2^{20}}{2^{96}} + \mathbf{Adv}_F^{\mathit{PRP\text{–}CCA}}(\mathcal{B}, \eta_k)$$

$$\approx 2^{-53.678} + \mathbf{Adv}_F^{\mathit{PRP\text{–}CCA}}(\mathcal{B}, \eta_k)$$

$$\mathbf{Adv}_F^{\mathit{UF\text{–}CMVA}}(\mathcal{A}, \eta_k) \leq 2^{-53} + \mathbf{Adv}_F^{\mathit{PRP\text{–}CCA}}(\mathcal{B}, \eta_k)$$

Once again, we use AES-128 as block cipher, and since the best attack known to this day needs $2^{126.1}$ operations [BKR11], we expect $\mathbf{Adv}_{AES-128}^{\mathit{PRP\text{–}CCA}}(\mathcal{B}, 128)$ to be considerably smaller than 2^{-53}. Therefore, using such assumptions and sizes, we find that the unforgeability security bound is low enough, still with an overhead of 12 bytes.

2.3.4 Symbolic Analysis of the Protocol

We then conducted a symbolic analysis of SR3, focusing on authentication. Overall, the symbolic analysis focuses more on the protocol than on the cryptography, because of a few key differences with the previous section.

First, this analysis is done in the *symbolic* model instead of the *computational* model. This model assumes the perfect encryption hypothesis, which specifies that cryptographic primitives are perfect black-boxes, and the attacker can only interact with them through their expected properties: for instance, the attacker can decrypt $\mathrm{E}_k(\mathrm{x})$ if and only if he has knowledge of k. This knowledge is built from a Dolev-Yao model [DY83], which specifies that the attacker knows what can be built or deduced from the communications happening in the network. The attacker may also send new messages, replay sessions of the protocol, and manipulate, redirect or delete messages going through the network.

The symbolic model is more restrictive for an attacker than the computational model and allows different assumptions in the other parts of the model. Here, we still use the description from Figure 2.3 page 38, but instead of proving security properties for a single session of the protocol, we allow several sessions for the attacker to execute in order to achieve its goal.

We focus on authentication, more precisely non-injective agreement for both participants. This property is defined in the hierarchy of [Low97]. Consider two actors, A and B running a protocol. If the protocol verifies this property, it guarantees that if A completes a run of the protocol, apparently with B, then B has previously been running the protocol, apparently with A, and both A and B agreed on the same data (in our case, this data is both *Data* and N).

We use the tool *Scyther* [Cre08], a symbolic prover for cryptographic protocols, in order to obtain the proof. *Scyther* reports that the protocol provides this form of authentication for both participants to the protocol, for a bounded number of sessions. The modelization file is available online [ADJL13c] and in Appendix A.

2.4 Simulation Methodology

We then evaluate our protocol with respect to classical measures, namely, delivery rate of the messages, fairness and number of hops. We also study the resiliency of SR3 against several attack scenarios. For these purposes, we ran simulations in *Sinalgo* [Dis08], an event-driven simulator for WSNs, and we compared the performances of SR3 to those of six other routing protocols.

Sinalgo is one among many network simulators that are suitable to our purposes. For instance, to simulate WSNs, there are several specialized frameworks. Various levels of detail are possible: for instance, TOSSIM [LLWC03], JiST/SWANS [BHvR05a, BHVR05b] and ATEMU [PBM$^+$04] emulate the node's operating system, while others such as SENS [SKA04] and WSNet [CFH06] model nodes with protocol layers (note that WSNet has a tool for full emulation). Several others network simulation frameworks exist, which are not specific to WSNs: for instance, ns-2 and its successor ns-3 [FV07], OMNeT++ [V+01], J-sim [SHK+06], SSFNet [CON02]. In order to focus more on the routing behavior of SR3, and to avoid getting lost in side-effects, we chose to use a simple model for the radio medium and link-layer protocol. Using any full-stack simulator would add unwarranted complexity. In the end, we chose Sinalgo [Dis08] for its simplicity, and because of favorable previous experiences.

Note that the simulations in this chapter aim to show properties that are related to those presented in Chapter 3. The reasons behind our choice of using simulations to analyze SR3 is detailed in Chapter 3, where we compare the goals and limitations of the method we chose here, and the formal proofs in the model presented in that chapter.

2.4.1 Topologies

We deployed sensors uniformly at random on a square plane. We positioned the sink at the center of the square plane. The compromised nodes are selected uniformly at random among the sensors, and we consider that they are compromised after all the required initialization of the evaluated protocol. Two nodes can communicate if and only if their Euclidean distance is less or equal to a preset fixed range, *i.e.*, the topology is a Unit Disk Graph (UDG). We only considered *connected* topologies.

The communication links are asynchronous and FIFO. The transmission time of each link follows an exponential random distribution of constant parameter 1. Only honest sensors generate data to route. The time between two consecutive data generations at the same sensor also follows an exponential random distribution, whose parameter is the same for all sensors and whose value depends on the average degree and the number of sensors in the network, to prevent congestion.

If we fix the number of nodes n and the range of the UDG, we can tune the size of the simulation area to control the expected average degree $\overline{\delta}$ of the network. In our simulations, n varies from 50 to 400 with a step of 50, and $\overline{\delta}$ is either 8, 16, or 32. Three examples of topologies are provided in Figure 2.7, with nodes in blue and the sink in red. The percentage of compromised nodes varies from 0 to 30%, depending on the context: we chose the amount to have a noticeable, but not overwhelming effect.

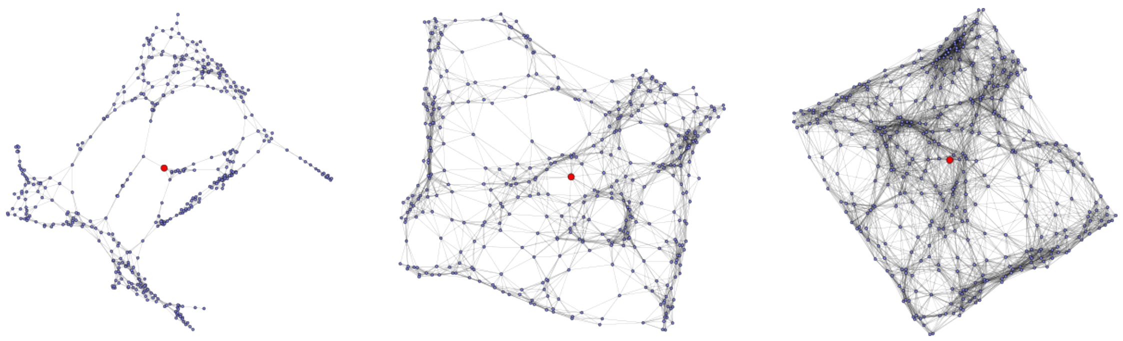


Figure 2.7: Three random topologies of $n = 200$ and respectively $\overline{\delta} = 8$, 16 and 32.

We considered various attack scenarios, where compromised nodes made *selective forwarding*: each compromised node drops received messages with a probability $p \in (0, 1]$ (if $p = 1$, the node is called a *blackhole*). In addition, some compromised nodes may have some additional "bad" skills, *e.g.*, they may be *wormholes* or *Sybil*.

For each setting (number of nodes, average degree, attack scenario, amount of compromised nodes, and routing algorithm), we ran 20 simulations over 20 different UDG, which were randomly generated. We chose 20 simulations as the whole campaign of simulations included here already take several weeks to run, and running more of those would not improve significantly the precision of our results.

In each simulation, 500 000 data packets are generated. The simulation stops once all packets have been routed or lost. We made more than 13 000 simulation runs and the overall number of generated data is greater than 6 billions.

2.4.2 Benchmark Protocols

We present a few protocols from the literature, and our reasons for choosing or excluding them from our simulations.

Panel of Selected Protocols

We selected a panel of six protocols from the literature: the uniform Random Walk (RW) [AKL+79], the Greedy-Face-Greedy protocol (GFG) [BMSU01], the Gradient-Based Routing (GBR) [SS01], and three of its variants, Randomized GBR (RGBR), Probabilistic Randomized GBR (PRGBR), and Probabilistic Randomized Duplicating GBR (PRDGBR) [EOMVK11]. Note that RGBR, PRGBR, and PRDGBR have been introduced for purpose of resiliency.

GFG is a geographic routing protocol, where two modes are alternatively used: Greedy and Face. The Greedy mode routes messages to the neighbor of the current node that is geographically the closest to the destination. This mode is preferably used, but may lead a message to a dead end. In this case, the Face mode allows the message to escape from this dead end, by routing messages along faces of the network graph (see [SS01] for a description of this algorithm). This protocol requires a planarization of the network graph, which virtually reduces the amount of links available for routing.

Planarizing the network from the standpoint of individual nodes is a very complex problem in a real wireless network, due to the eventual presence of physical obstacles [Kar01]. Fortunately, we are in a simulated context with UDGs, and so we can simply planarize the network using Gabriel Graphs [GS69], where the edge between two nodes a and b is removed if there exists a node in the disk of diameter $[ab]$.

GBR consists in routing messages along the breadth-first spanning tree (BFS tree) rooted at the sink. In our context, this spanning tree is built at the beginning of the simulation, before any nodes are compromised in the network. So, even if intruder nodes change behavior over time, the initial tree is kept.

RGBR uses the levels of neighbors in the BFS tree: each sensor chooses the next hop for each message uniformly at random among its lowest-level neighbors. In PRGBR, each sensor chooses between two modes: (1) with probability 0.4 the message is routed according to RGBR; (2) with probability 0.6 the message is routed to a neighbor of same level (if no such a neighbor exists, the sensor uses mode 1).

Finally, PRDGBR duplicates the messages before each hop and routes the two messages independently using PRGBR. To avoid congestion, each node drops the received copies of messages it has already seen, which we implement using an infinite memory on each node. Note that this algorithm can lose messages in a safe network: we illustrated this problem in Figure 2.8, where each arrow denotes a copy of a single message, and the edge labels represent the chronological order. This problem stems from the fact that it is possible for a node to send all copies of a message to a same-height neighbor, and then drop all further copies of this message. If several neighboring same-height nodes (a and b in our example) send all duplicates of the message to each other (communications 3 to 6), the message will be lost. This is especially problematic on ladder-like topologies, but quite marginal (less than a percent of all messages) on the random topologies we use in this chapter. Note that we assumed duplicatas are routed independently as it is specified [EOMVK11]. If message duplicata were required to be routed to two different nodes, then message loss would require at least three nodes and it would be less likely to occur.

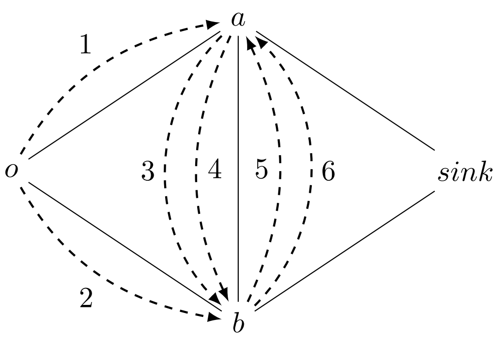


Figure 2.8: PRDGBR message loss in a safe network

Other Protocols

There are several notable protocols which we did not include in our panel. We chose the previous protocols either because they illustrate well the family in which they belong (GFG, RW, GBR), or because they are designed with objectives similar to SR3 (all randomized versions of GBR).

Both LSR and OLSR are optimal in the number of hops as long as nodes learn an up-to-date topology. Since topologies are static in our model, and we only have one destination node, these two protocols will behave similarly to PRGBR after the topology is discovered, although with residual control messages noise.

The protocols based on AODV we mentioned have not been designed for their resiliency against blackholes. AODV provide a routing mechanism based on route requests and replies, that may be compromised by attackers. The additions that SAODV and SEAR propose will only prevent some specific flaws of AODV, but it will not improve the delivery rate when facing a standard blackhole intruder. In the end, this mechanism of route request-then-reply will end up routing messages along shortest edges, which is also similar to the PRGBR protocol, as our topologies are static and the quality of links stays the same.

CASTOR We chose not to include the CASTOR protocol [GPP$^+$10] in our panel, as it is not optimized for all-to-one routing, and it is also unsuited for the low memory of wireless sensors. Using it in our experimental setting (a WSN where all nodes send data) requires having at least as much concurrent flows as there are sender nodes. Nodes close to the sink may therefore be expected to keep track of up to as much flows as there are nodes in the network, which requires each node to store four reputation counters per neighbor per flow (which amounts to up to 12KB of memory for a degree 16 node in a 200 nodes network, given counters are 8-bits). On top of this, nodes need to remember for a given duration which messages have been seen, from which neighbor do they come from, to whom they have been sent, and the eventual corresponding acknowledgment, for a total of 4 bytes per message. This information must be held by each node for an amount of time depending on the network settings, which is related to the time needed for a round-trip in the network. For networks that are slow and wide, and if a node has to support a large part of the network traffic as it may be the case for the sink's neighbors, then the memory requirement becomes very large when compared to the requirements of the other protocols of the panel. Furthermore, CASTOR requires nodes to broadcast messages. This happens when nodes do not have at least one high-reputation neighbor in the current flow, in which case they broadcast messages. If several flows are created around the same time, as it is the case in our model, then the corresponding messages will overload the network during the start-up phase.

SIGF We chose not to include protocols from the SIGF family in our panel [WFSH06]. This algorithm rests upon the assumptions that nodes can overhear their neighbor's communications, that they reliably know their 2-neighborhoods, and that they have access to geographical information. These three hypothesis are stronger than what we require for SR3, and they would unfairly advantages SIGF here as our simple radio model does not include interference, radio concurrency issues or any probability of message loss during transfers. The authors also acknowledge major flaws in the algorithm, which would allow an intruder to bypass entirely the forwarding check in the reputation computation and to appear as attractive as a regular node to its neighbors:

- The algorithm is vulnerable to intruders able to adjust their transmission patterns (for instance, using steerable antennas). Such an intruder is able to forward packets in a way that ensures its upstream neighbor's trust, while not actually forwarding packets to its downstream neighbor. With this behavior, an intruder node blocks messages from reaching their destination (*i.e.* it is a blackhole) without suffering a reputation loss.

- The authors show that SIGF-2 is resistant against Sybil attacks, where the intruder node acts as if it were several distinct nodes, using fabricated identities, which all act as blackholes. We argue that this attack is a lot weaker than what an attacker with these capabilities can do. Consider the following attack: A Sybil attacker can make one of its identities a simple blackhole, and all its other identities will forward all of their messages to that blackhole. This way, the only node losing reputation for not forwarding messages will be the blackhole, whereas all the other nodes will keep a good reputation. The Sybil node will still drop all the messages it receives, while only sacrificing a single identity's reputation, as opposed to the authors' evaluated attack where all identities of the Sybil node ends up blacklisted.

To provide some perspective on this issue, we provide an experimental evaluation of the impact of the first attacker, compared to a regular blackhole further in Section 2.5.6.

2.4.3 Choice of SR3 Parameters

SR3 uses three lists, whose respective sizes are bounded. We made several experiments to set the size of each list to the appropriate value.

Choosing $L_{Reputation}$'s Size

We consider first the list $L_{Reputation}$ (of size s_R), the list that holds the reputations of the neighbors. As the choice of next hop includes a $\frac{1}{s_R+1}$ probability of that choice being done uniformly at random, the size of the list is important. If the list is too short, then this probability will be relatively high, and the overall routing will always be too random. If the list is too long, once $L_{Reputation}$ is filled, then the next hop choices will nearly always follow the previous experiences, which will hurt the algorithm's adaptivity.

This is best illustrated with a sinkhole attack. Consider an intruder node which behaves well for some time to attract traffic, and then starts to drop messages. A short list will cause nodes to send more messages uniformly at random, which will cause the list to be refreshed faster, but the delivery rate is also going to be reduced. On the other hand, when s_R is large, the system spends more time to recover, but it will recover to a higher delivery rate as the probability of a choice being done uniformly at random decreases.

To create that sinkhole attack, we use *wormholes*, which are compromised nodes that directly communicate with the sink. This ability makes them more attractive than other nodes, as they allow delivering messages faster to the sink. During the first third of the simulation, wormholes send all received data messages in this communication channel and route acknowledgments as honest nodes in order to obtain an high reputation. Next, they become blackholes, *i.e.*, they drop all messages they receive.

To determine which size to choose for $L_{Reputation}$, we experiment several possible values in the sinkhole scenario. In these experiments, we implement $L_{AckRouting}$ and

L_{Sent} as infinite lists, which correspond to their ideal behavior. We chose to evaluate this attack in a network of 200 nodes, average degree 16, containing 10% of blackholes and 10% of wormholes that become blackholes after the first third of the simulation. We present some results in Figure 2.9, where we represent the delivery rate over time of three simulations where only s_R varies. For each point (x, y) of the curves, y is the delivery rate computed over a window of 50 000 messages, from the $(x - 50000)^{th}$ to the x^{th} emitted message.

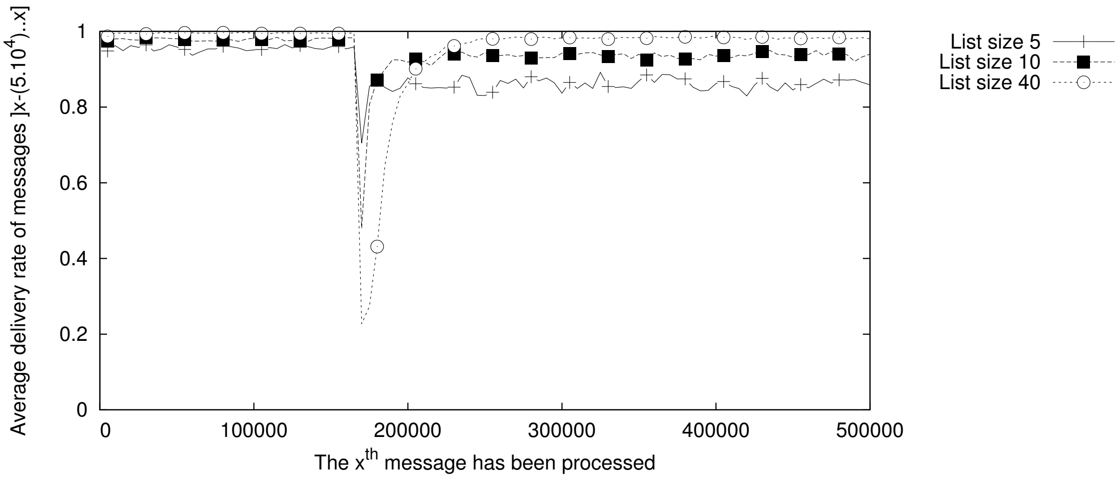


Figure 2.9: Average delivery rate (10% of WH/BH, 10% of BH, $n = 200$, $\overline{\delta} = 16$)

We test the aforementioned scenario with several values for s_R — from 5 to 40 with a step of 5. For each candidate s_R, we run simulations on networks of either 100, 200, or 400 nodes, with an expected average degree of either 8, 16, or 32, containing both 10% of blackholes (BH) and 10% of wormholes/blackholes (WH/BH) as previously described. We made 10 simulations (each with a different topology) for each setting (list size, number of nodes, degree). The overall results are summarized in Figure 2.10. In each cell of the table, we print the value of s_R that offers the best average delivery rate. We also notice that the average delivery rate is very similar for a large range of list sizes (*e.g.*, there is not much difference between our results for lists of size 5 and 10). We chose $s_R = 10$, which is a good compromise in each setting.

		Number of nodes		
		100	200	400
Average degree	8	10	5	5
	16	15	10	5
	32	20	15	5

Figure 2.10: Best observed $L_{Reputation}$ size s_R, when facing 10% BH and 10% WH/BH.

Choosing L_{Sent}'s Size

Next, we consider the list L_{Sent}. The size of both L_{Sent} and $L_{AckRouting}$ are bounded for practical reasons only. Indeed, sensors have tight local memories. Now, having infinite sizes for those lists would allow acknowledgments to always add reputation, and to always be routed in the message's footsteps. The goal here is to find a reasonable size that achieves the adequate trade-off between performance and resource consumption.

To set the size s_S of L_{Sent}, we led experiments, where $s_R = 10$ (the value we choose previously) and $L_{AckRouting}$ is still implemented as an infinite list.

In those experiments, our goal is to minimize the proportion of valid acknowledgments that return to their destination, while their corresponding nonce has been removed from L_{Sent}. This causes the acknowledgment to be discarded. We denote this event a *false negative.*

We made our simulations in safe networks, because this corresponds to the case where the routes followed by data messages and acknowledgments are longer, and the longer the routes are, the more messages get lost.

We tested L_{Sent} of size 1 to 5. The results for a particular setting is depicted in Figure 2.11. Actually we obtained similar results for each setting. We can see value 2 for s_S is sufficient to reach our objective of 99% of accepted acknowledgments. Therefore, we chose to set s_S to 2.

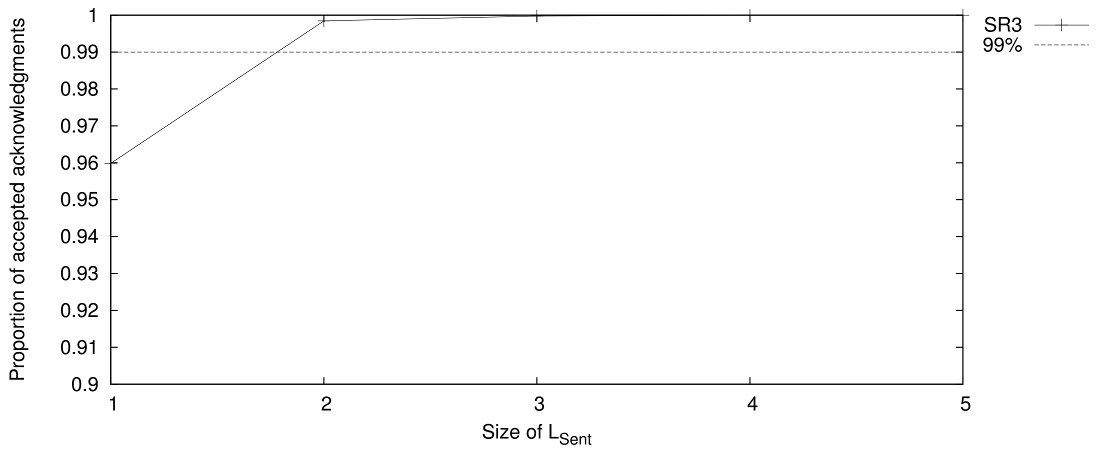


Figure 2.11: Proportion of accepted acknowledgments, depending on L_{Sent} size ($n = 200$ and $\overline{\delta} = 8$)

Choosing $L_{AckRouting}$'s Size

Finally, we repeated the same process to set the size s_A of $L_{AckRouting}$ using the sizes previously selected for $L_{Reputation}$ (10) and L_{Sent} (2). Again, we tried to minimize the proportion of false negatives. If the list is too small, more messages would be randomly routed, which would in turn increase the delay before they reach their destination, and consequently, increase the number of false negatives.

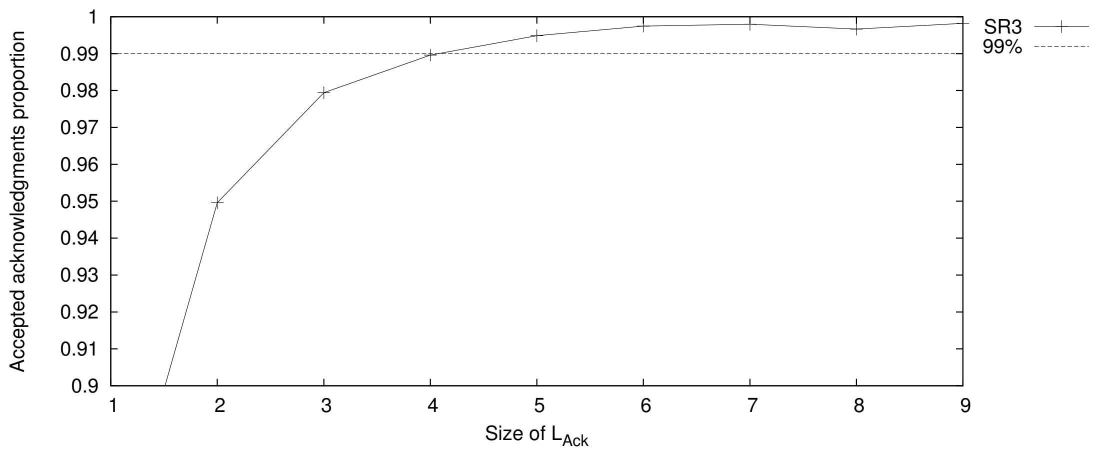


Figure 2.12: Proportion of accepted acknowledgments, depending on $L_{AckRouting}$ size ($n = 200$ and $\overline{\delta} = 8$)

Figure 2.12 shows the results we obtained using the same setting as the previous example (200 nodes, average degree 8). We tried networks of 100, 200 and 400 nodes, with degree 8, 16 and 32, and they all give similar results. The proportion of accepted acknowledgments is always above 99% using a $L_{AckRouting}$ of size 5, so we chose that value for s_A.

2.5 Experimental Evaluation of SR3

We now present the results of the experimental evaluations we made on SR3. We first present the results of SR3 and our panel of algorithms against blackholes, focusing on the delivery rate in Section 2.5.1 and the fairness in Section 2.5.2 of these algorithms. We then follow in Section 2.5.3 by observing the potential damage caused by availability attacks on SR3, and continue with the performances in safe networks of SR3 and the other algorithms in Section 2.5.4. We investigate the effects of various sinkhole attackers in Section 2.5.5: simple wormholes that drop all traffic after some time, and wormholes who alternate between attracting and dropping traffic. Finally, in Section 2.5.6, we show the simulations we made to evaluate the impact of SIGF-specific adversaries against this algorithm.

2.5.1 Average Delivery Rate

We study the delivery rates observed in networks containing 30% of blackholes (BH). The number of nodes in these networks varies from 50 to 400 nodes (with a step of 50).

Figures 2.13, 2.14, and 2.15 show our results, respectively when the expected average degree is $\overline{\delta} = 8$, 16, and 32. Note that with 30% of blackholes, several honest nodes cannot safely reach the sink and consequently have delivery rate zero. We remark that SR3 always offers a better delivery rate than the other protocols on networks of average degrees 8 and 16. In networks of average degree 32, its delivery rate is approximately the same as PRDGBR, while still better than the other protocols. In particular, the larger (in terms of number of nodes) the networks are, the greater the difference in delivery rates is.

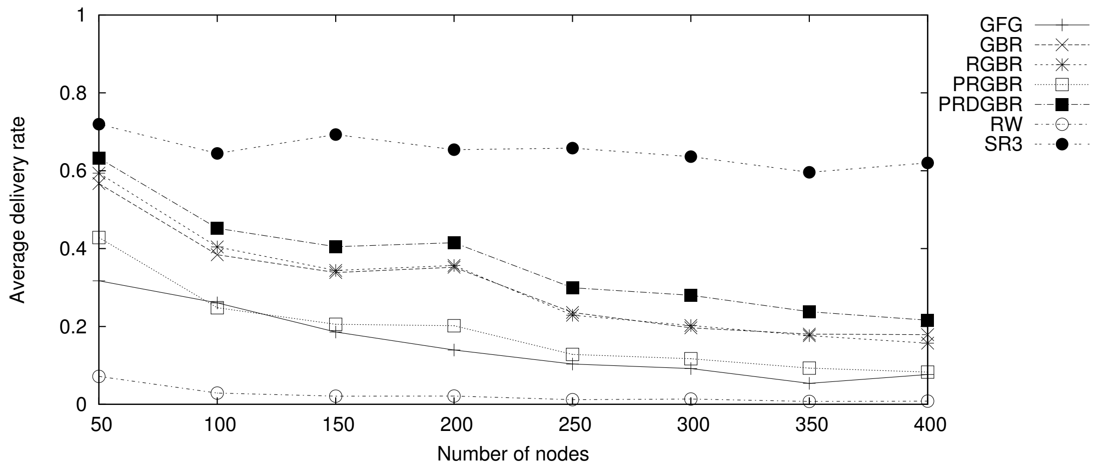


Figure 2.13: Average delivery rate (30% of BH nodes, $\overline{\delta} = 8$)

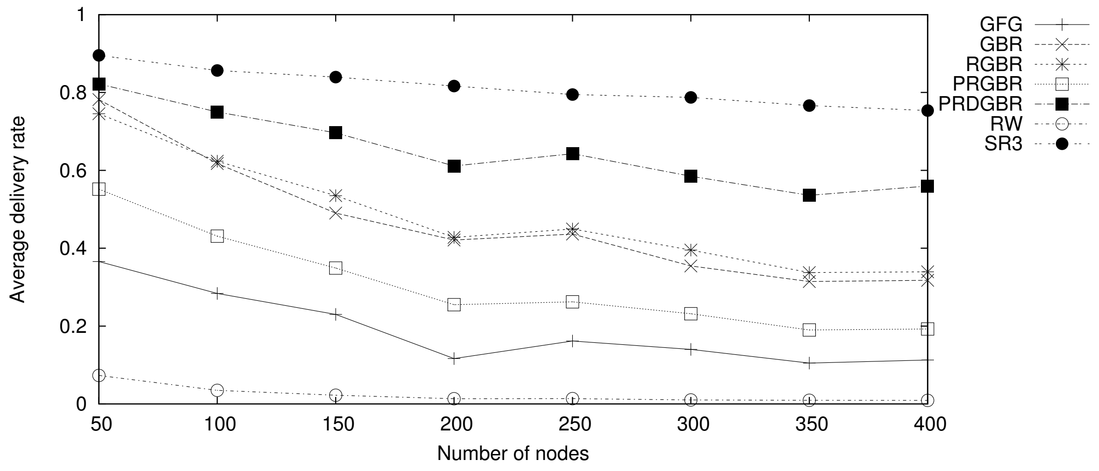


Figure 2.14: Average delivery rate (30% of BH nodes, $\overline{\delta} = 16$)

Figure 2.16 shows the delivery rates observed in networks of size $n = 200$ facing 30% of blackholes (BH). The average degree of the networks varies from 8 to 32. Again, we can remark that SR3 always offers the best delivery rate in that case. Moreover, as for

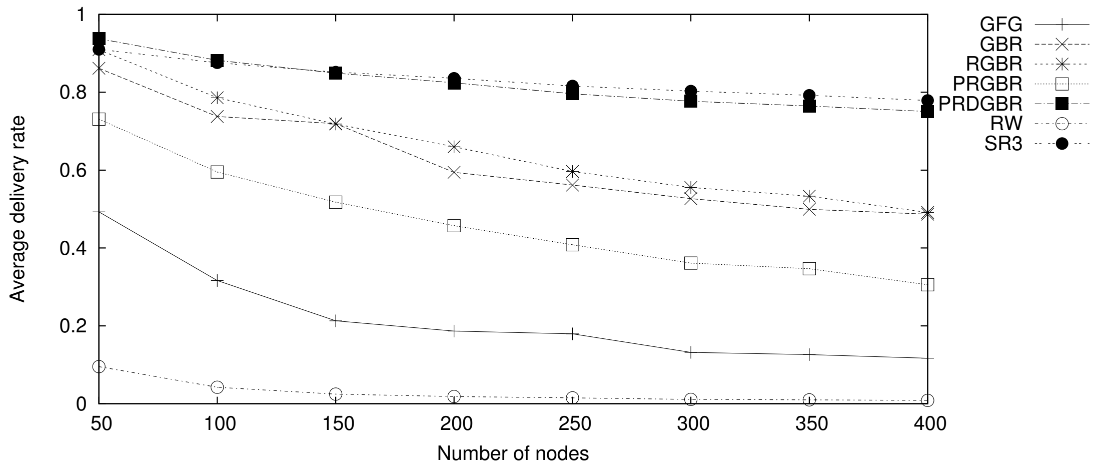


Figure 2.15: Average delivery rate (30% of BH nodes, $\overline{\delta} = 32$)

RW and GFG, the average delivery rate of SR3 is insensitive to the degree variation. In contrast, the observed delivery rates for gradient-based protocols are low in sparse networks. In high-density networks, the performances of PRDGBR match those of SR3. However, SR3 use only two messages per data, while PRDGBR duplicates the messages at each hop, and consequently heavily increases the load of the network.

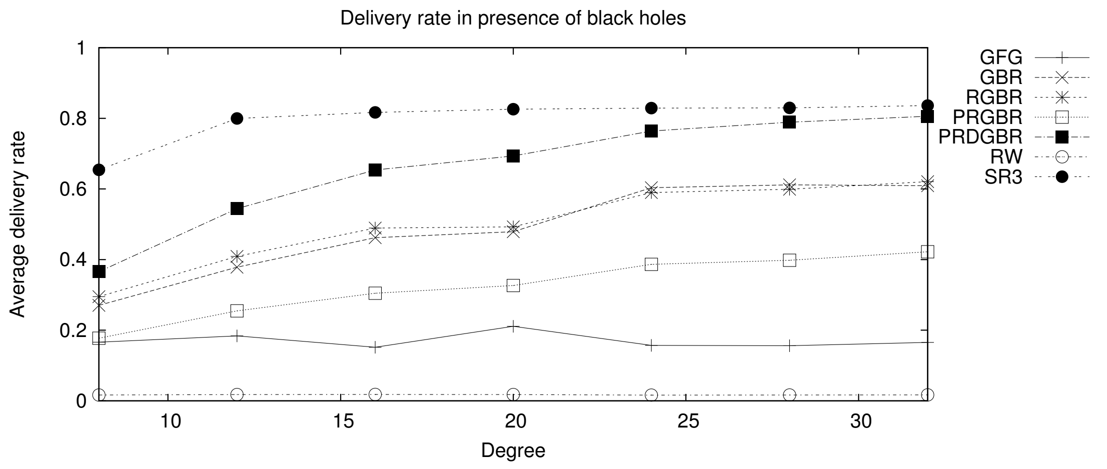


Figure 2.16: Average delivery rate (30% of BH nodes, $n = 200$)

SR3 also efficiently combats the selective forwarding (SF) attacks. Figure 2.17 shows the average delivery rates observed in networks of size $n = 200$ and average degree $\overline{\delta} = 8$ that have to face 20% of compromised nodes, according to the drop rate of these nodes. We can observe that, except RW, all protocols of the panel achieve a slow degradation in delivery rate when the drop rate increases. Still, SR3 offers one of the best performance.

Only PRDGBR has performances close to those of SR3 when the drop rate is of 100% (that is, when compromised node are actually blackholes).

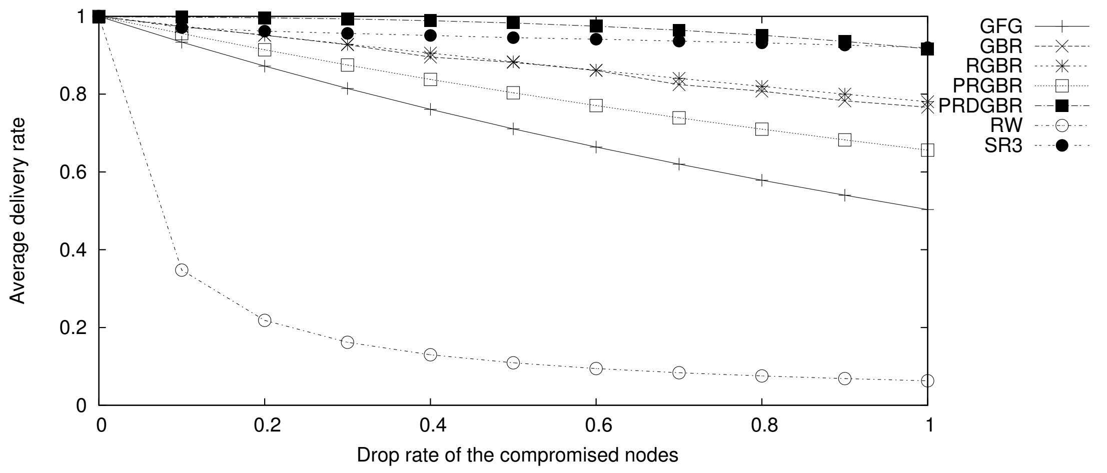


Figure 2.17: Average delivery rate (20% of SF nodes, $n = 200$, $\overline{\delta} = 8$)

We also considered networks of size $n = 200$ and average degree $\overline{\delta} = 8$, where 10% of nodes are both blackholes and Sybil (SY). The number of pseudonymous identifiers of these compromised nodes varies from 1 to 10. Note that in this scenario, the list of neighbors available to nodes is not accurate, as expected from the attacker. We can observe in Figure 2.18, that except for GFG, adding Sybil nodes does not change the relative performances in the panel. Actually, GFG is insensitive to Sybil attacks because it does not use node identifiers. Now, still in that case, SR3 offers the best performances.

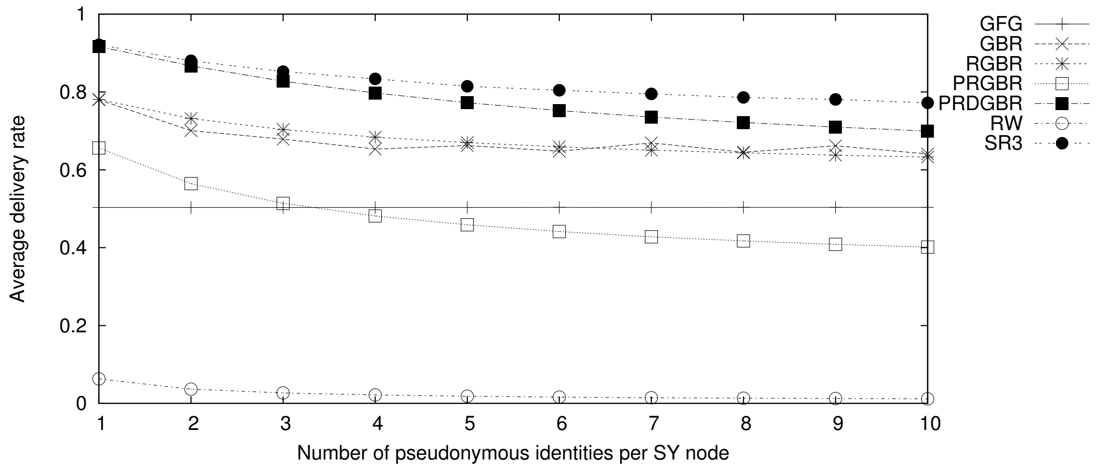


Figure 2.18: Average delivery rate (10% of SY nodes, $n = 200$, $\overline{\delta} = 8$)

2.5.2 Fairness

Fairness among the delivery rates of honest nodes is a desired property in routing protocols. A classical way to capture this property is to compute the standard deviation of the delivery rates of honest nodes. Figure 2.19 shows the average and standard deviation of delivery rates observed in networks of size $n = 200$ and average degree $\overline{\delta} = 32$, when facing 30% blackholes. The smaller the standard deviation is, the fairer the algorithm is. Now, a shortcoming of this measure is that when the delivery rates are uniformly very low (like for example in RW), the observed fairness is good. So, analyzed alone, this measure is misleading.

Algorithm	**Average delivery rate**	**Standard deviation**
GFG	0.186	0.359
GBR	0.594	0.469
RGBR	0.660	0.279
PRGBR	0.457	0.228
PRDGBR	0.824	0.158
RW	0.018	0.020
SR3	0.836	0.040

Figure 2.19: Average delivery rate and standard deviation of the delivery rate of nodes (30% of BH, $n = 200$, $\overline{\delta} = 32$)

Instead, we propose here to visualize the distributions of delivery rates. Figure 2.20 shows an example of our method. In this figure, we consider the same simulations as in Figure 2.19. There is one chart per algorithm of the panel. Each of these charts represents the range of possible delivery rates from 0 to 100%, by intervals of 10%. The color shade encodes the proportion of nodes having the corresponding delivery rate. Consider, for example, the RW protocol: almost all nodes have a delivery rate of less than 10%. In contrast, using SR3, almost all nodes have a delivery rate greater or equal to 70%. We can clearly observe two classes of processes when looking at GFG and GBR: nodes have either 0% or 100% of delivery rate; these protocols are unfair. The probabilistic variants of GBR are fairer: the delivery rates are spread on the whole range, but still these results are weaker than those observed for SR3.

We also provide other results in Figure 2.21. Simulations were run on networks of size $n = 200$ with an average degree $\overline{\delta} = 8$, also when facing 30% blackholes. Overall, we observe results similar to the previous setting, with a few differences. First, GBR and the variants have overall lower deliver rate and fairness, as they behave better in highly connected networks. We also remark that more nodes lost all of their messages, even when running a randomized algorithm: these are nodes for which no safe path to the sink exist.

2.5.3 Availability Attacks

We then evaluated how availability attacks interfere with the good operation of the protocol. Our objective here is to get some intuition on the damage such attacks may

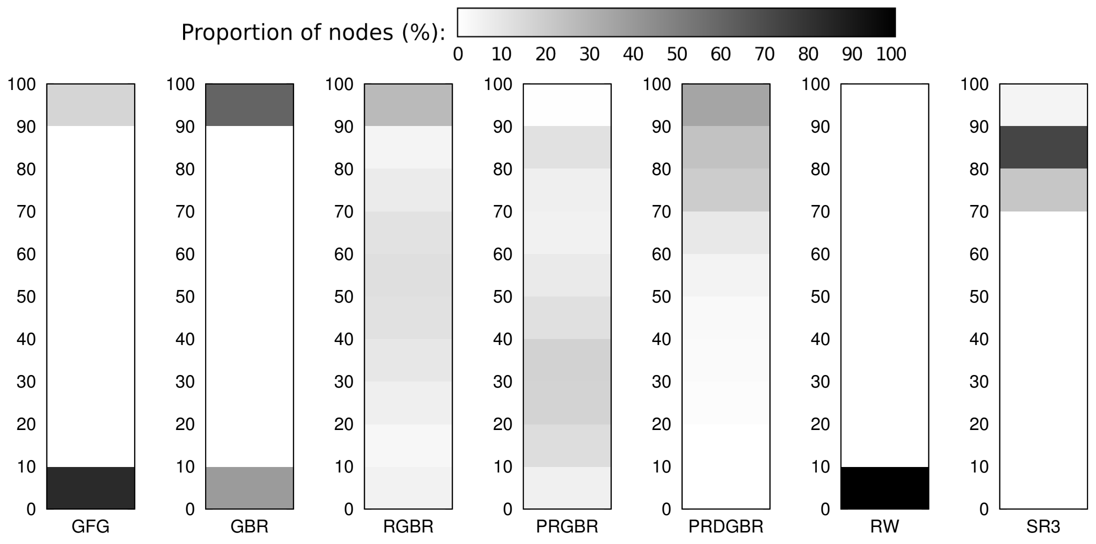


Figure 2.20: Average delivery rate distribution (30% of BH, $n = 200$, $\overline{\delta} = 32$)

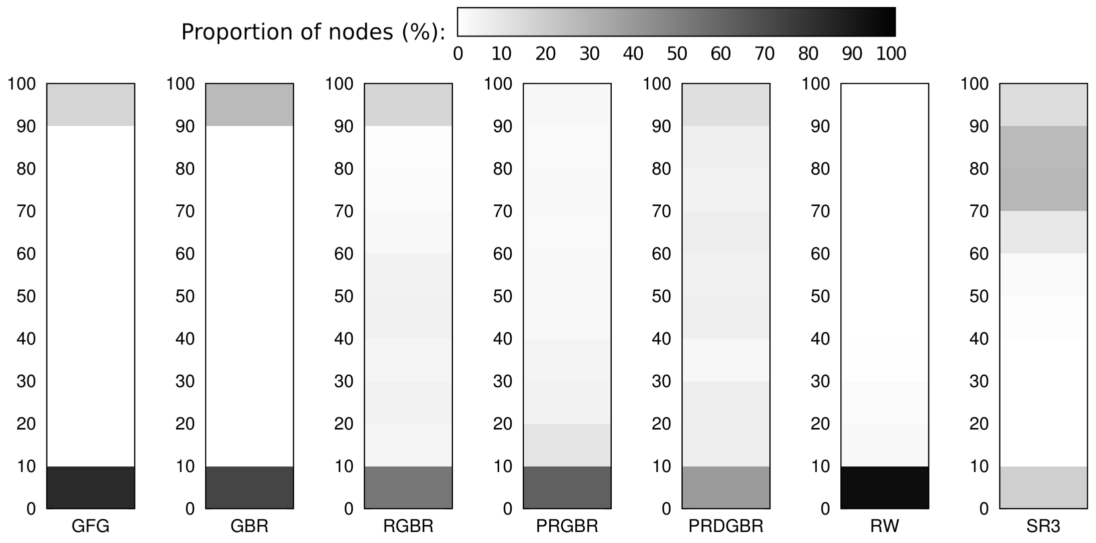


Figure 2.21: Average delivery rate distribution (30% of BH, $n = 200$, $\overline{\delta} = 8$)

cause to SR3, when compared to the other attackers presented in this section. In practice, the effect of these attacks strongly depend on how the radio link layer operates. However, our link layer model is too simple to handle realistically this type of attacks. We therefore approximate the impact of jamming attacks by considering that it affects nodes in either of two ways:

- In the first model, jamming a node totally disables its ability to communicate, *i.e.* we remove it completely from the simulation. We denote this model "Removed".
- In the second model, jamming a node only disable its ability to route messages, while its neighbors still consider it as a valid next hop, effectively turning it into a blackhole. We denote this model "BH".

There exists a lot of mitigation techniques for jamming attacks (see [LVHD+05, LKP07] for discussions of such countermeasures on WSNs). Since we are looking for an estimation of the potential damage for SR3, we do not consider any countermeasure here.

We slightly deviate from the experimental setting of the rest of this chapter to run these evaluations. As before, our topologies are random UDGs as described in Section 2.4.1, with intruders selected at random such that all honest nodes have an available safe route to the sink. Then, depending on the attack scenario we consider, we turn all 1- or 2-neighbors of intruders (except the sink, which is not influenced) into *jammed nodes*, which, depending on the chosen model, behave as blackholes, or are entirely disconnected from the network. At the end of this process, it is very likely that some non-jammed nodes end up being disconnected from the sink. Note that the jammed nodes are not considered as message sources. For each model, we generated 50 graphs of 200 nodes, expected degree 16, with merely 5% of attackers (that is, 10 nodes).

To give a better understanding of this attack's impact, we provide several measures for each simulation:

- The proportion of honest nodes which are turned into jammed nodes,
- The proportion without access to a safe route to the sink among the remaining honest nodes,
- The proportion of messages that reached the sink (denoted "delivered"),
- The proportion of messages transmitted to an attacker or a jammed node (denoted "intercepted"),
- The proportion of messages lost due to not being able to reach the sink at all (denoted "lost").

Simple Jamming

In the first scenario, we consider the impact of attackers who simply jam their neighbors' communications. Intuitively, this attack creates 10 holes in the network, each of them having a radius equal to the nodes' communication range. We present in Table 2.22 our results over 50 simulations in this experiment, with networks of $n = 200$, $\overline{\delta} = 16$, and 5% of initial attackers.

Scenario	Jammed nodes	Nodes with a safe route among the rest
Simple jamming	52.2%	67.9%

Scenario	Delivered	Lost	Intercepted
Simple jamming, Removed	67.9%	32.1%	0.0%
Simple jamming, BH	58.0%	0.0 %	42.0%

Figure 2.22: Simple jamming results (5% of jammers, $n = 200$, $\overline{\delta} = 16$)

We observe that even with only 5% of attackers, this attack manages to jam more than half of the nodes in the network on average. Of the remaining nodes, more than 30% are entirely disconnected from the sink on average, which leaves only around one third of all nodes able to send messages to the sink. When jammed nodes are considered to act as blackholes, the proportion of intercepted messages is only a little higher than the proportion of lost messages in the other case, as expected from the delivery rate results of SR3 against blackholes. Overall, the impact of this attack is strong, but this is understandable: as the average degree is high, each node will jam around 16 other nodes (minus the overlaps). This results in networks with large holes, which often disconnects entire parts of the network from the sink.

Acknowledgment Routing Loop

In the second scenario, the attackers use an amplification technique, specific to SR3, that uses the acknowledgment routing mechanism. To use it, an intruder i needs to be a neighbor of two interconnected nodes a and b. This attack uses the acknowledgment routing mechanism to create an ACK routing loop. For reference, we summarized the attack process in Figure 2.23.

First, the attacker needs to setup the attack. It consists in i sending a pseudo-message to a and b, making them believe the communication came from each other. This pseudo-message needs to be of the form $\langle C, h(N), s, b\rangle$ for a, with a N generated by the attacker, and arbitrary data for C and s (as these fields are not used by intermediate nodes). In the same way, i sends $\langle C, h(N), s, a\rangle$ to b. Thus, a thinks the message comes from b, and vice versa. Both nodes update their $L_{AckRouting}$ to reflect that belief.

The attacker then sends the corresponding pseudo-acknowledgment $\langle N, s\rangle$, which is going to loop between a and b until it gets lost or randomly routed if the entry in one of the $L_{AckRouting}$ gets removed. To further amplify that attack, i can send multiple copies of the pseudo-acknowledgment, or run multiple concurrent attacks. This attack allows the creation of routing loops in the immediate neighborhood of attacker nodes.

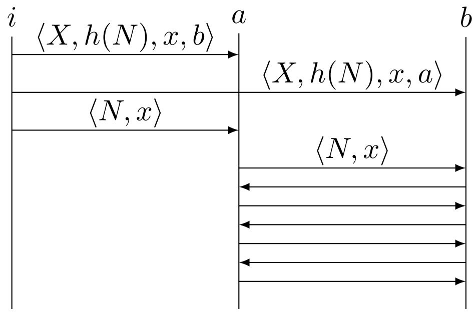

i
a
b
$\langle X, h(N), x, b\rangle$
$\langle X, h(N), x, a\rangle$
$\langle N, x\rangle$
$\langle N, x\rangle$

Figure 2.23: Availability attack on SR3 using ACKs

We provide the results we obtained in Figure 2.24, over 50 simulations in this experiment, with networks of $n = 200$, $\overline{\delta} = 16$, and 5% of initial attackers.

Scenario	Jammed nodes	Nodes with a safe route among the rest
Ack loop	85.8%	12.0%

Scenario	Delivered	Lost	Intercepted
Ack loop, Removed	12.0%	88.0%	0.0%
Ack loop, BH	10.2%	0.0%	89.8%

Figure 2.24: Acknowledgment loop jamming results (5% of jammers, $n = 200$, $\overline{\delta} = 16$)

The results are an amplified version of the previous experiment. With ten nodes operating a 2-neighborhood wide jamming, more than 85% of honest nodes in the network are jammed. We also observed that the few remaining nodes are very often isolated: only 12% of them have a safe route to the sink. As with the previous scenario, the difference between the two models of jammed nodes is relatively small. This attack is very efficient in our model for a simple reason: a 2-neighborhood encompasses a lot of nodes in a graph of average degree 16. For instance, a single attacker in the neighborhood of the sink effectively disconnects it from the rest of the network (as the attacker's 2-neighborhood contains all of the sink 1-neighborhood). More importantly, this attack has a very distinctive footprint: acknowledgments should not loop in SR3, and it is fairly easy to notice such loops or just the increase in quantity of received acknowledgments) with an intrusion detection system. Our conclusion is that such a system is a simple and necessary complement to SR3 in order to mitigate this weakness of our protocol.

2.5.4 Average Number of Hops

Here, we are only interested in the messages that are successfully delivered. So, we consider safe networks. Figures 2.26a, 2.26b and 2.26c show the average number of hops of data messages in networks of average degree respectively $\overline{\delta} = 8$, 16 and 32, where the size n varies from 50 to 400. Figures 2.27a, 2.27b and 2.27c show, on the same networks, the average route length expansion factor, *i.e.* the average of the route length for a node, divided by the optimal route length for that node.

First, note that we do not show results for RW in the figure because they are significantly worse than other protocols of the panel, *e.g.*, for 50 nodes and $\overline{\delta} = 16$, its average number of hops is 40, and for 400 nodes and $\overline{\delta} = 16$, its average number of hops is 529. Then, by definition, routes followed using GBR or RGBR are optimal, so we only include GBR. Finally, SR3 generates longer routes than the geographical and gradient-based protocols due to its lack of knowledge about the network. However, this length stays reasonable (*i.e.* we always observed lengths drastically smaller than n), and scales slowly with the number of nodes. Note that Greedy-Face-Greedy does not behave well in some low-degree graphs, this is due to the existence of dead ends in those graphs.

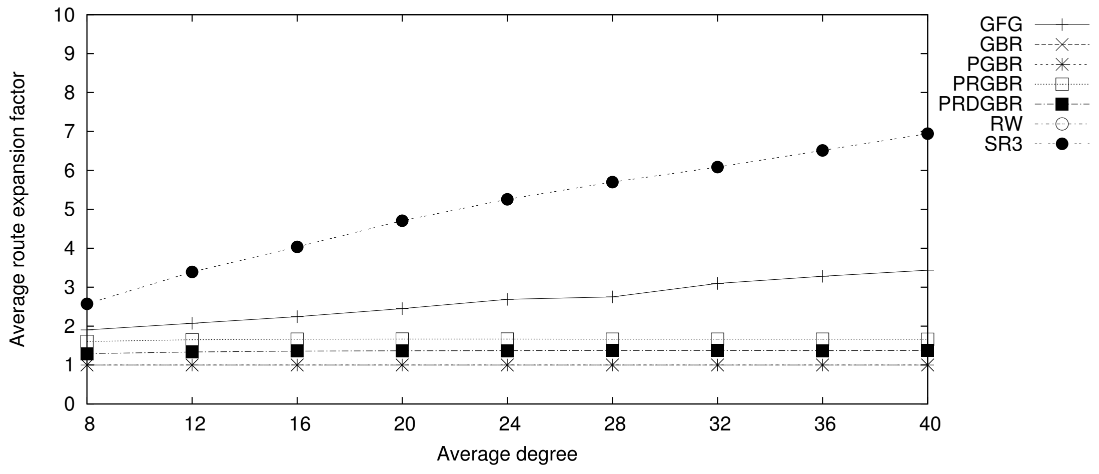
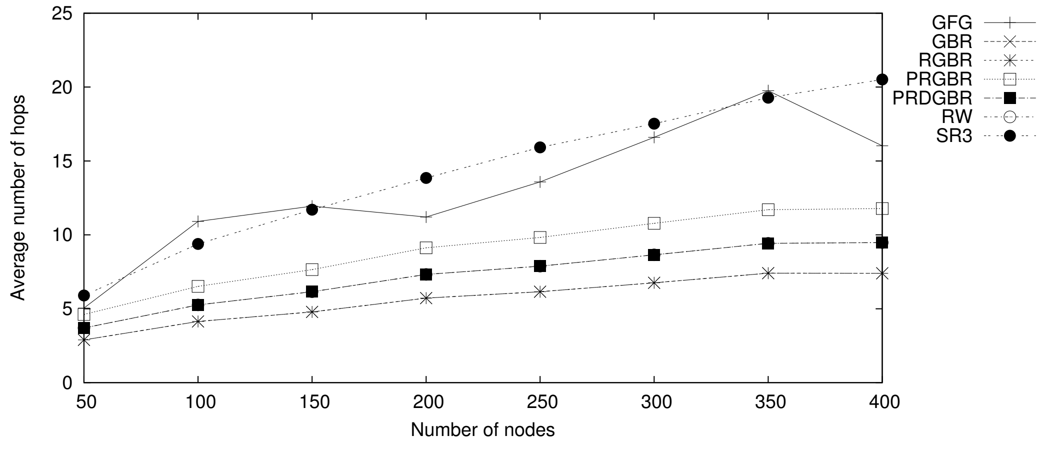


Figure 2.25: Average route extension factor in safe networks ($n = 200$)

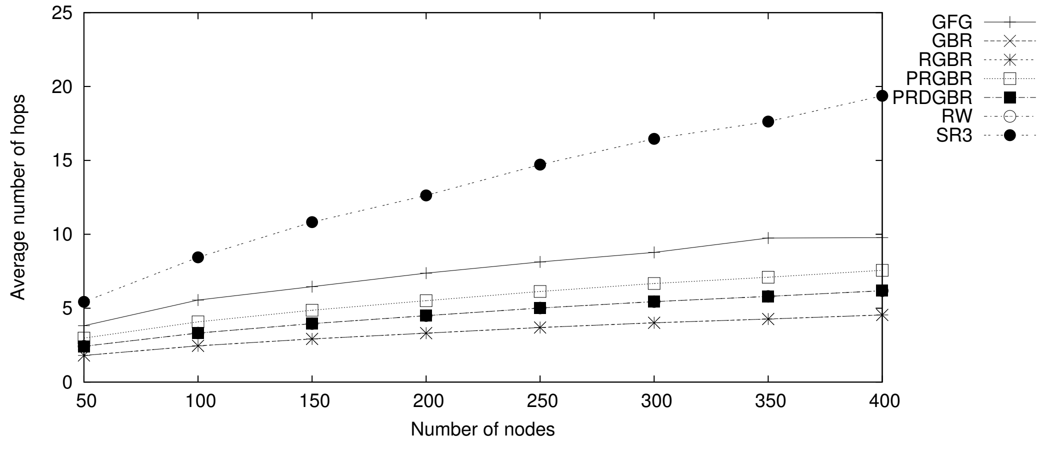
For each protocol from our panel, we also observed how the route expansion factor evolves depending on the expected average degree of the network. The results are provided in Figure 2.25. We observe that both PRGBR and PRDGBR have a nearly constant overhead, as the probability of not progressing towards the destination is fixed for each hop. More importantly, we observe that SR3 route expansion factor increases with the degree, but starts at a reasonable value of less than 3.

2.5.5 Self-Adaptivity

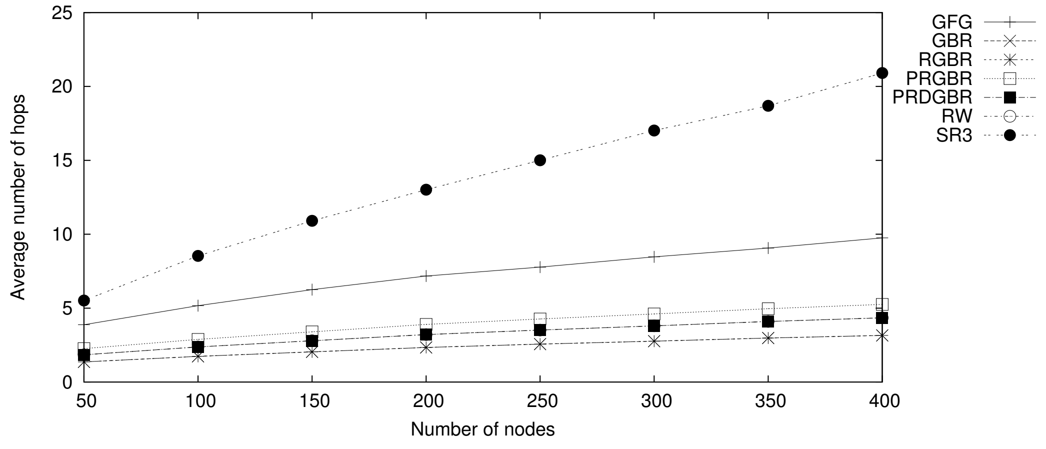
Thanks to its reputation mechanism, SR3 self-adapts to the variations of the hostile environment. To see this, consider the following scenario: in a network of $n = 200$ nodes with average degree $\overline{\delta} = 8$, we assume 5% of blackholes and 5% of wormholes/blackholes (WH/BH), that first behave as wormholes to attract the traffic, and then become



(a) Average number of hops in safe networks where $\overline{\delta} = 8$

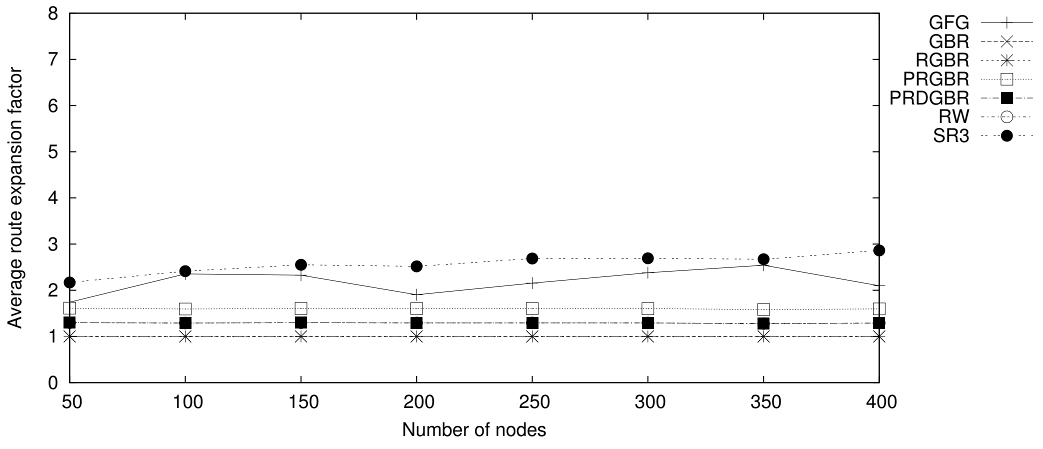


(b) Average number of hops in safe networks where $\overline{\delta} = 16$

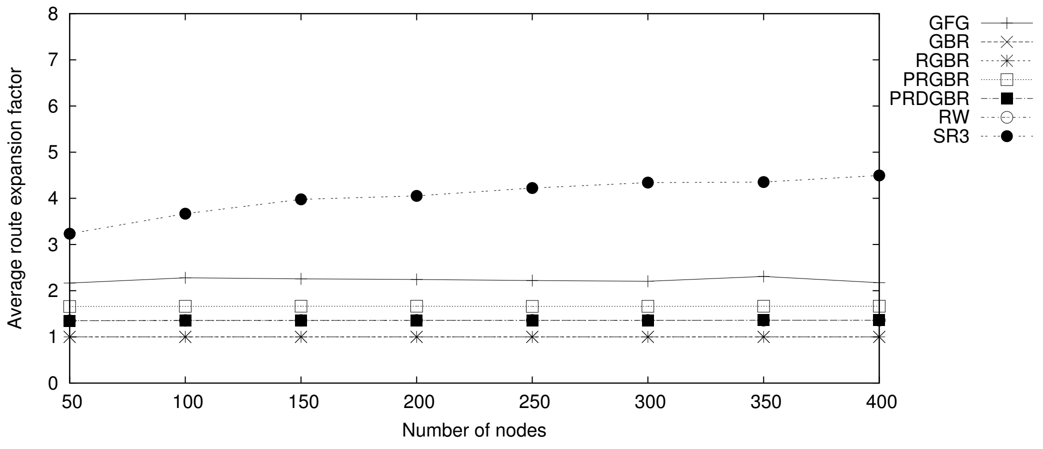


(c) Average number of hops in safe networks where $\overline{\delta} = 32$

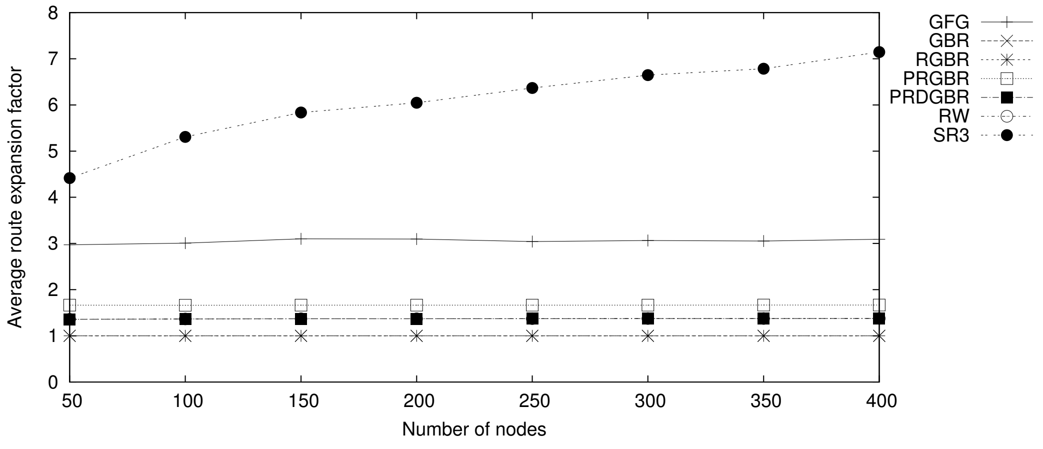
Figure 2.26: Average number of hops in safe networks



(a) Average route expansion factor in safe networks where $\overline{\delta} = 8$



(b) Average route expansion factor in safe networks where $\overline{\delta} = 16$



(c) Average route expansion factor in safe networks where $\overline{\delta} = 32$

Figure 2.27: Average route expansion factor in safe networks

blackholes after one third of the simulation. Such nodes appear more attractive to their neighbors because they allow delivering messages faster. Figure 2.28 shows the evolution of the delivery rates of each protocol: for each point (x, y) of the curves, y is the delivery rate computed over a window of 10 000 messages, from the $(x - 10000)^{th}$ to the x^{th} emitted message. Only SR3 recovers from this attack.

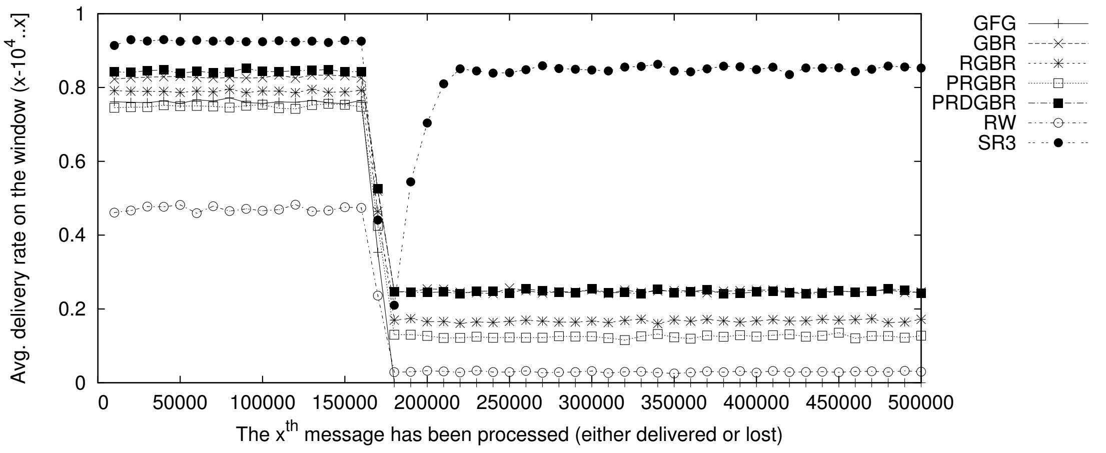


Figure 2.28: Average delivery rate (5% of WH/BH, 5% of BH, $n = 200$, $\overline{\delta} = 8$)

We show in Figure 2.29 the average delivery rates distribution in the same setting as previously (5% of WH/BH, 5% of BH, $n = 200$, $\overline{\delta} = 8$). We see that the deterministic protocols cause nodes to deliver either none, one third or all their messages. The randomization used in the GBR variants causes a more spread out distribution, but there is still a large concentration of nodes delivering between 30 and 40 percent of their messages. Finally, as observed before, SR3 is fair: most nodes deliver more than 70 percent of all their messages, whereas for the other protocols managing to deliver some data, the delivery rates of the nodes in this scenario strongly depend on their position in the network.

In this attack, the attacker is able to build up trust during the first third of the lifespan of our simulated networks. To better illustrate the resiliency of SR3, we observed the effects of the same attacker where attacker nodes switch back and forth between wormhole and blackhole behavior after various amounts of time. We tried switching after 100×2^p messages have been emitted, where p goes from 0 to 10, and on the same networks as previously (5% of WH/BH, 5% of BH, $n = 200$, $\overline{\delta} = 8$). The average delivery rates we obtained are available in Figure 2.30. Notice that an attacker switching behaviors every 25600 packets causes the highest reduction in delivery rate, but by an acceptable amount. We show the results over time of this attack strategy over three different intervals in Figure 2.31. We observe that a lower interval between behavior changes causes lower reductions in the delivery rate, since the attacker cannot gather as much trust at each blackhole phase. Furthermore, the fairness of SR3 stays good for

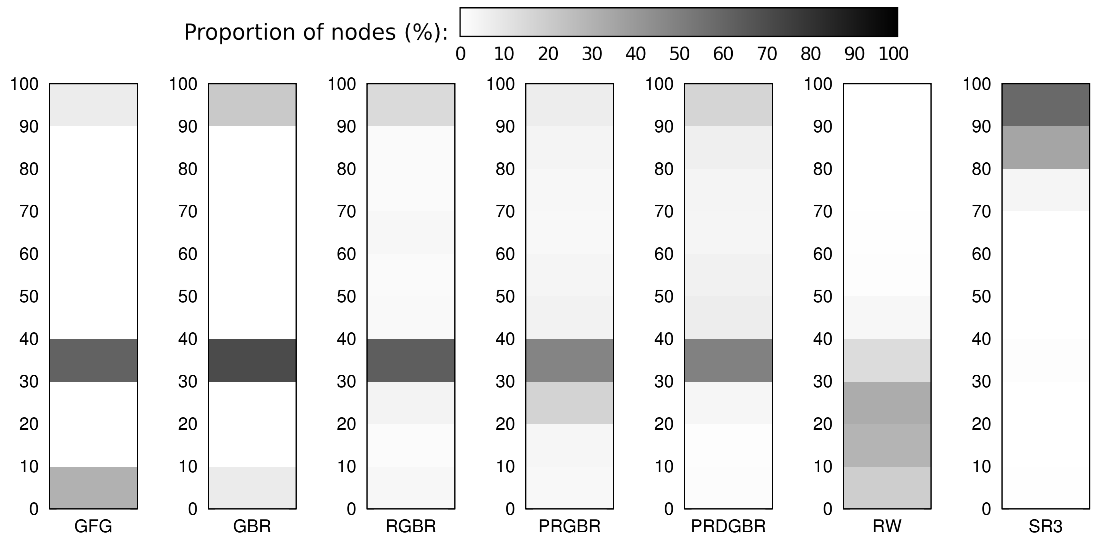


Figure 2.29: Average delivery rate distribution (5% of WH/BH, 5% of BH, $n = 200$, $\overline{\delta} = 8$)

every interval, *i.e.* most nodes deliver around 85% of their messages. Overall, a sinkhole attacker can increase its impact by choosing the right frequency of behavior change, but even with this optimization, the overall delivery rate reduction stays acceptable. On the other hand, this attacker strategy can be useful to strongly disrupt a network for a very short period of time, which can be useful depending on the context.

2.5.6 SIGF-Specific Smart Blackholes

As we described in Sections 2.1.3 and 2.4.2, SIGF is a geographical routing algorithm which is complemented with a reputation mechanism, in a way similar to SR3. However, SIGF uses local and untrustworthy knowledge to build the reputation of nodes. We implemented one of the attacks exploiting that weakness, where the intruders forwards messages to everyone except their destination, in order to appear reputable while actually dropping messages. We denote these intruder nodes *smart blackholes* (SBH). Such intruders, in our implementation, routes messages similarly to a node running SIGF with all neighbors' reputations equal to 1. However, instead of broadcasting messages, it sends the message to all neighbors but the message's destination. We chose to use the SIGF-2 variant as described in [WFSH06].

We present in Figure 2.32 the results for this scenario, and we compare them with how SIGF, SR3 and GFG behave when facing regular blackholes. We observe as expected that SIGF's reputation mechanism works very well against traditional blackholes, but the delivery rate is cut by half (near GFG's level) when facing these smart blackholes.We conjecture that if the reputation mechanism is subverted, SIGF acts as a somewhat randomized geographical routing protocol.

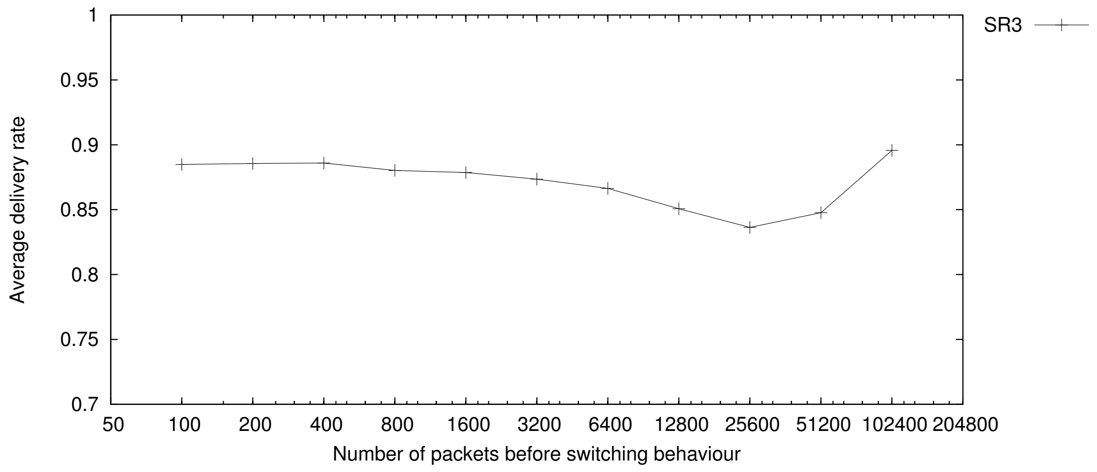


Figure 2.30: Average delivery rate depending on the behavior change interval of the attacker (5% of alternating WH/BH, 5% of BH, $n = 200$, $\overline{\delta} = 8$)

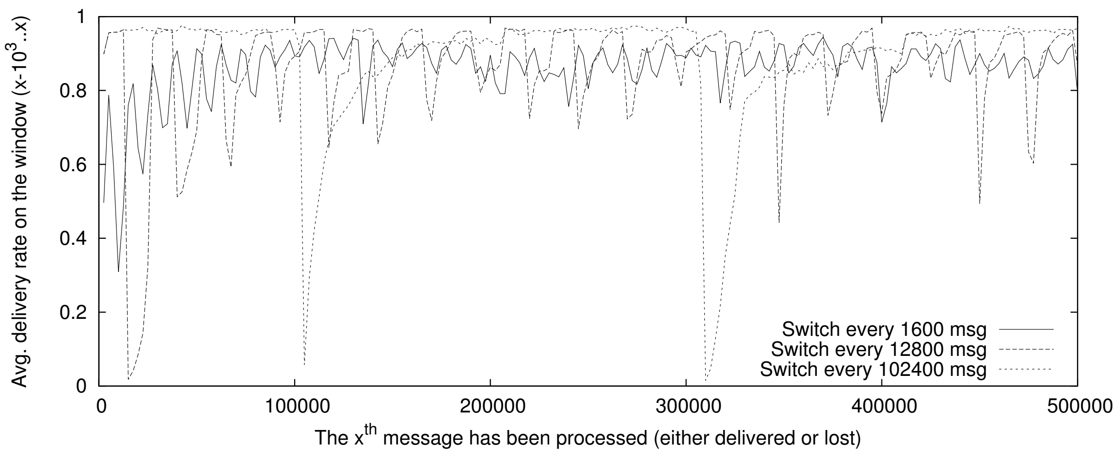


Figure 2.31: Delivery rate over time (5% of alternating WH/BH, 5% of BH, $n = 200$, $\overline{\delta} = 8$)

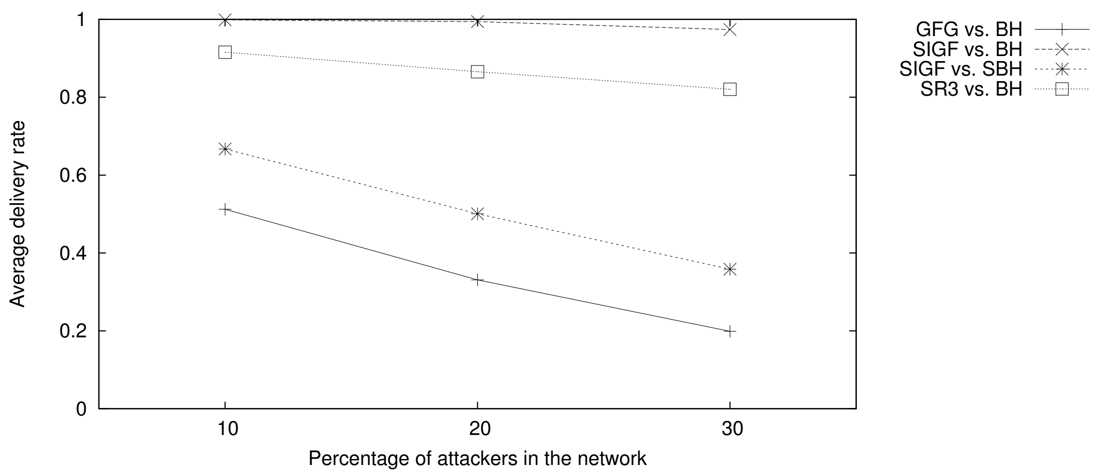


Figure 2.32: Average delivery rates depending on the proportion of attackers ($n = 200$, $\overline{\delta} = 16$)

2.6 Conclusion and Future Work

We proposed SR3, a secure and resilient algorithm for convergecast routing in wireless sensor networks. Using only lightweight cryptographic primitives (symmetric encryption, hashing, and secure random numbers), SR3 achieves data confidentiality and data packet unforgeability, which we formally proved with CryptoVerif and Scyther. Using simulations, we showed the resiliency of SR3 in various attack scenarios, including selective forwarding, blackhole, wormhole, and Sybil nodes. Our comparative study shows that the resiliency accomplished by SR3 is better than the one achieved by several routing protocols of the literature, even those whose targeted metric is resiliency.

Our algorithm's main strength is its adaptivity. Against static attackers, or a complex network, it manages to reach a correct infrastructure, which yields a good delivery rate and is globally fair between nodes. This adaptivity also allows operation with absolutely no knowledge about the topology, unlike geographical routing protocols. Furthermore, nodes do not need to trust any of their neighbors, which is an useful feature when using nodes which are not tamper-proof or reliable.

On the other hand, this adaptivity also causes some trouble when adversaries want to disrupt the network's operation for small bursts of time. Consider the case of a WSN emitting alerts when a burglar is detected in a building: our algorithm needs time to rediscover routes and refresh its reputation lists, during which the delivery rate of messages is going to be lower. Hence, SR3 is best-suited for networks who need long-term reliability and fairness over responsiveness to punctual events.

Future Work

An implementation of SR3 in a WSN testbed platform is currently being finalized by Orange Labs. The preliminary results are promising, and everything should be ready soon for real-world experimentation.

We noticed during this work that SR3 can also operate well in settings where messages are supposed to go to any of several destinations. Without any modification or configuration, both the message and acknowledgment routing work flawlessly, which may be useful in some use cases: for instance, adding a sink to an already-existing network may reduce the overall energy consumption of the network (see for instance [KSC+05]).

Similarly, as our protocol self-adapts to the topology, we expect that having nodes moving around will not change how SR3 operates. If nodes are able to detect that their neighbor gets out of range, then they can simply remove all traces of that neighbor from their lists, and the rest of the protocol will sort itself out by falling back on the reputation of remaining neighbors stored in $L_{Reputation}$. Similarly, upon reaching new neighbors, the moving node will progressively increase their reputation if they are reliably delivering messages. However, experimental evaluation is needed to evaluate which level of mobility the algorithm can withstand.

The recovery phase of SR3 is also something that could be improved. We did not specify any reaction to a high message loss rate, unlike CASTOR where such an event causes the next messages to be broadcasted. This kind of behavior may allow SR3 to recover faster from attacks and reduce message loss, at the cost of more traffic in the network. If by hypothesis, no attackers are present at the beginning of the network lifetime (because of a manual deployment for instance), it may be interesting to bootstrap reputations in SR3 using route discovery protocols right after nodes are deployed. Obviously, these changes would require a careful analysis of their security implications.

We proved three security properties of the protocol. Our proofs address data confidentiality, authenticity and integrity of the packets, and we have shown through simulations that the common routing-level attacks' impact is weak. Obviously, we cannot cover all possible routing-level attacks this way, and the work in Chapter 3 originated from this concern. However, we designed SR3 to route messages based on a single mechanism, which is as simple and trustworthy as feasible. Furthermore, the formal proof of the nonce confidentiality proves that our acknowledgment system verifies that no attacker can reliably create an acknowledgment before it is delivered.

We mentioned the acknowledgment routing loop attack, which is merely an availability concern. Such attacks are hard to address from SR3's standpoint as it would require a costly authentication of each node's messages by their neighbors. More generally, we feel that availability attacks will always exist, and that routing protocol designers should not go to unreasonable lengths to prevent them, as such availability attacks usually have very distinguishable footprints (here, an acknowledgment routing loop). Such footprints, if they do not happen in a safe environment, imply that the attack can be easily noticed by an intrusion detection system looking for such behaviors. We discuss this interaction between protocols and IDS further in Chapter 4.

Chapter 3

Incorruptibility of routing protocols

Analyses of routing protocols security are nearly always supported by simulations, which often evaluate the ability to deliver messages to a given destination. Several competing definitions for secure routing exist, but to our knowledge, they only address source routing protocols. In this chapter, we propose the notion of *corruptibility*, a quantitative computational definition for routing security based on the attacker's ability to alter the routes used by messages. We first define *incorruptibility*, and we follow with the definition of *bounded corruptibility*, which uses two routing protocols as bounds for the evaluated protocol. These definitions are then illustrated with several routing algorithms. Finally, we provide a variant of these notions for attackers who cannot craft or reroute packets until the challenge message.

Les analyses de la sécurité des protocoles de routage reposent presque toujours sur des simulations, qui évaluent la capacité du protocole à délivrer ses messages aux bons nœuds. Il existe plusieurs définitions différentes pour concevoir la sécurité du routage, mais à notre connaissance, elles considèrent seulement les protocoles de *source routing*, où les routes sont déterminées avant que le message ne soit envoyé. Nous proposons la notion d'*incorruptibilité*, une définition calculatoire et quantitative pour la sécurité du routage basée sur la capacité d'un attaquant à altérer les routes empruntées par un message. Nous illustrons ensuite ces définitions par plusieurs analyses de protocoles.

Contents

3.1 Introduction

Internet is made out of several independent entities controlling their own networks. To be routed, packets need to get through several networks until they reach their destination, and so the Internet can be seen as a large ad-hoc network. In this context, routing relies on several protocols, including the Border Gateway Protocol (BGP). This protocol ensures the dissemination of routing information between autonomous systems (AS), and is notoriously insecure [Mur06, ND04]. For instance, the AS 7007 incident caused an internet-wide outage in 1997 [Bon97], because this AS declared itself able to route to the whole Internet. This declaration was made in a way that ensured most networks would choose the AS as the preferred gateway to the rest of the Internet. This misconfiguration then propagated through the Internet, overloading the faulty AS, and causing huge packet losses.

Another example, still related to BGP, has been seen more recently in the wild (see [HCG13]). In this incident, China Telecom's subnetwork declared itself preferential for the routing to more than 50,000 IPs, including some strategic subnetworks for the United States of America. Unlike the previous example, the infrastructure of the problematic subnetwork still managed to route packets to their destination.

In the context of WANETs, attacks and misconfigurations do not attract the same public attention as in traditional networks. However, from a research standpoint, they are well-studied. For instance, in [WCWC07], the authors present some classical routing protocol vulnerabilities. In [KW03], the analysis is more specific to the security of routing protocols on wireless sensor networks, which are a subset of the WANET family with more limited resources and specific protocols.

Our intuition is that a secure routing protocol should guarantee that malicious parties cannot influence the routes a message will take, or at least that this influence is limited in a clearly stated way. We call such protocols *incorruptible*, and using an incorruptible routing protocol would have prevented both previously mentioned incidents. We do not consider confidentiality or integrity of the data, as they are properties which are not necessarily tied to the routing layer.

3.1.1 Contribution

We propose the first steps towards a computational notion for the security of routing protocols, based on the ability for an attacker to influence how messages are routed. We provide three measures. The first one quantifies the difference between routing protocols in a safe context. The second one, denoted *routing protocol corruption*, quantifies how much an attacker is able to change how messages are routed. The last definition allows one to prove that an attacker can only corrupt a protocol within some limits. We only consider protocols where the nodes memories do not evolve once the attack begins. For each of these definitions, we provide an analysis of routing protocols.

3.1.2 Related work

In [PH02], the authors proposed the Source Routing Protocol (SRP), which is an on-demand route discovery protocol. In on-demand routing protocols, route discovery is the process of building a valid path for a given data message. Using BAN logic [BAN89], they claimed that routes generated by this protocol are correct and their integrity is respected. An attack has been found later on that protocol by [Mar02], who argued that the results of the analysis are flawed because such an analysis is "*a misuse of BAN*", as this logic has been designed to study trust relationships, and not security notions.

The authors of [ABV06] provided a definition of provably secure on-demand route discovery. They assume the adversary has compromised a few nodes in the network which gave him full control of their actions and memories. To prove security of protocols, they use the *simulation paradigm*, which uses two models: the *real-world model*, and an *ideal-world model* where the protocol is idealized. This way, they can detect specific problems in the protocol, while putting asides the concerns inherent to such routing protocols in general. In their model, a protocol is considered secure if the executions set in the real-world model are indistinguishable from those set in the ideal-world version. This model was expanded in [BT10], where the authors provided an automated way to check protocols in this model.

The main differences between our model and theirs lie in the limitations on the evaluated protocols, and the property being evaluated. Regarding protocols, their model is able to track of the evolution of the internal states on nodes and to model broadcasts, while we do not consider these situations. Regarding the properties, the one we verify is universal to routing protocols, and could be applied to any, while their property of secure route discovery only makes sense on source routing protocols.

3.1.3 Outline

In Section 3.2, we model networks and protocols in our formalism, and in Section 3.3 we provide some routing protocols. We present our definitions of an incorruptible routing protocol in Section 3.4, along with the analysis of one of the protocols given in Section 3.3. Finally, we conclude and present the perspectives of this work in Section 3.5.

3.2 Definitions

To represent the network *topology*, we use a vertex-labeled directed graph named the *topology*, and denoted $T = \{V, E, f\}$, where vertices V represent network nodes, and edges E represent their connectivity. We consider only static networks, and we suppose that nodes cannot send messages to themselves. The function f associates labels to nodes, which are used to model pre-existing distinctions between nodes, such as sinks and sensors in the case of a wireless sensor network. We denote by $\mathcal{N}eig_v$ the set of neighbors of a node v (that is, the nodes at one hop of v). We notice that $v \notin \mathcal{N}eig_v$.

3.2.1 Routing protocols

To forward messages, all the nodes follow a routing protocol $\mathcal{P}$. This protocol must verify that all messages are routed independently at the time of the analysis. The path that a message will take should not be influenced by what other messages have been routed before. Note that acknowledgments can be modeled by considering them as the continuation of the route of the initial message.

We define K as the array of individual node memories, denoted $K[v]$ for any node $v \in V$. Once initialized, a node's memory is never modified again. This is a strong restriction, and it reduces the range of protocols that can be modeled. On the other hand, for a given message, these protocols generate routes in a way that do not depend on the past messages, which is an important property for the following security proofs. We discuss how to lift this restriction in the conclusion.

We denote by η_d the data size, and by η the security parameter for cryptographic functions. We write $a \xleftarrow{\$} X$ to denote that a is a random value obtained according the distribution represented by X. If X is a set, a is drawn at random using the uniform law on X. Similarly, if X is a probabilistic algorithm, a is drawn at random using the algorithm.

We define a routing protocol $\mathcal{P}$ as the set of four oracles $\{\mathcal{P}^I, \mathcal{P}^G, \mathcal{P}^R, \mathcal{P}^D\}$ which respectively model the initialization, message generation, routing and the depacketing phases. They are defined in Definitions 3.1 through 3.4.

Definition 3.1 (Initialization oracle)**.** *Let $T = \{V, E, f\}$ be a topology, with nodes $v_1 \dots v_n \in V$. The* initialization oracle *$\mathcal{P}^I(T, \eta)$ models the setup phase of $\mathcal{P}$ on the* *network represented by T, with security parameter η, which initializes the memories of the nodes. This oracle call returns K, an array associating to each node its memories.*

Definition 3.2 (Message generation oracle)**.** *Let $T = \{V, E, f\}$ be a topology with $o, d \in V$. Once some initial memories K has been defined, the* message generation oracle *$\mathcal{P}_K^G(o, d, \eta_d)$ models the generation of a new random data of length η_d to route by o, for d. This oracle call returns a message m. That call does not modify K.*

Definition 3.3 (Routing oracle)**.** *Let $T = \{V, E, f\}$ be a topology with $v \in V$. Once some initial memories K has been defined,* the routing oracle *$\mathcal{P}_K^R(v, m)$ models how v would route m given the initialization K. This oracle call returns either $\perp$ if no message is forwarded, or (w, m') if a message m' is forwarded to w (with $w \in \mathcal{N}eig_v$). That call does not modify K.*

Definition 3.4 (Depacketing oracle)**.** *Let $T = \{V, E, f\}$ be a topology with $d \in V$. Once some initial memories K has been defined,* the depacketing oracle *$\mathcal{P}_K^D(d, m)$ models how d would unpack the message m given the initialization K. This oracle call returns either the Data contained in the message if extractable, or $\perp$ if that operation is not possible. That call does not modify K.*

3.2.2 Message life cycle and routes

We now present how to model the natural life cycle of a message. First, the network needs to be set-up, in order to initialize the various items nodes need to route messages. This is done with the initialization oracle, by calling $K \overset{\$}{\leftarrow} \mathcal{P}^I(T, \eta)$. Then, a new message containing a random data is generated by a node o with destination d using the generation oracle $m_0 = \mathcal{P}_K^G(o, d, \eta_d)$. That message is first routed by the node o with the routing oracle $\mathcal{P}_K^R(o, m_0) = (h_1, m_1)$, assuming that message is not dropped. We then continue with its first hop, h_1, who acts depending on it: $\mathcal{P}_K^R(h_1, m_1) = (h_2, m_2)$. This process continues until the message finally reaches a node h_n such that $\mathcal{P}_K^R(h_n, m_n) = \perp$: at this point, the message is stopped. We refer to such a sequence $[h_0, \ldots, h_n] = \mathbf{R}$ as a *route*, which can be empty, in which case we denote it [ ].

Definition 3.5 (GenRoute($m_0, h_0, \mathcal{P}_K^R$))**.** *Given a message m_0, a node identifier h_0, and a routing oracle $\mathcal{P}_K^R$ initialized with K, we define* GenRoute*($m_0, h_0, \mathcal{P}_K^R$) the function that generates a route for m_0 starting at h_0. This function calls $\mathcal{P}_K^R$, first with arguments (h_0, m_0), and then with the pair (h_i, m_i) returned by the previous call, until the oracle returns $\perp$. The function returns the route $[h_1, \ldots, h_n]$.*

As $\mathcal{P}$ can be probabilistic, making several calls to GenRoute with the same arguments can result in different routes. However, since we require message routing independence, the probabilistic distribution of routes should stay the same, no matter what messages have been routed before. Finally, we define a predicate on routes which we denote Φ.

Definition 3.6 ($\Phi(\mathbf{R}, a, b)$)**.** *Given a route $\mathbf{R}$ and two nodes a, b, we define $\Phi(\mathbf{R}, a, b)$ as a function that returns true if and only if a route R contains a and that a appears before any occurrence of b.*

3.3 Examples of routing protocols

We now provide some routing protocols in our formalism. The $\mathcal{P}^D$ oracles are straightforward: given a message m and the initialized memories array K, they return the $Data$ that is contained in m.

Let $\mathcal{S}$ be a signature scheme. It provides a function $\textsc{GenAsymKeyPair}(\eta)$ which generates asymmetric key pairs given η the security parameter, $\textsc{Sign}(x, sk)$ which generates a signature of x using the key sk, and $\textsc{Verify}(x, pk, S)$ which verifies a signature S against the input i with key pk. We also define the function $\textsc{ShortRoute}(o, d, T)$ as the function that takes as input an origin, a destination, and a topology, and returns uniformly at random one of the shortest routes between the origin and destination. Finally, $\textsc{FindNext}(\mathbf{R}, v)$ is the function that returns the node identifier coming right after v in the route $\mathbf{R}$, or $\perp$ otherwise.

The null protocol $\mathcal{P}\emptyset$ is defined in Figure 3.1. It drops all messages, and adds no information in the packets it generates. The uniform random walk $\mathcal{RW}$ is defined in Figure 3.2. The following protocol is called $\mathcal{SI}$ (for Shortest-Insecure). It is described in Figure 3.3. That protocol stores routes in messages without cryptographic or algorithmic protections.

We define two other protocols, which stem from $\mathcal{SI}$, and use the signature scheme $\mathcal{S}$. First, $\mathcal{S}\emptyset$ (Figure 3.4) is a secured version of $\mathcal{SI}$, which prevents any alteration to the route stored in a message by using signatures. The route is signed by the message sender, and if that signature does not verify, then the message is discarded. The next protocol, $\mathcal{SR}$ (Figure 3.5), works in a similar way, but instead of discarding the message, it routes it randomly until the message finds the destination.

3.4 Secure routing

We now present how our notion of incorruptibility is formalized, and how it can be used to show the security of routing protocols. We begin by defining what is an attacker in our context, and follow with the measures of distance, incorruptibility, and bounded corruptibility.

```
Initialization $\mathcal{P}\emptyset^I(T,\eta)$:
1: Return $\emptyset$
Message generation $\mathcal{P}\emptyset^G_K(o,d,\eta_d)$:
1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: Return $Data$
Message routing $\mathcal{P}\emptyset^R_K(m,v)$:
1: Return $\perp$
```

Figure 3.1: Protocol $\mathcal{P}\emptyset$

```
Initialization $\mathcal{RW}^I(T,\eta)$:
1: Return $\emptyset$
Message generation $\mathcal{RW}^G_K(o,d,\eta_d)$:
1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: Return $next, (Data, d)$
Message routing $\mathcal{RW}^R_K(m,v)$:
1: $(Data, d) \leftarrow m$
2: if $v \neq d$ then
3:     $next \xleftarrow{\$} \mathcal{N}eig_v$
4:     Return $next, (Data, d)$
5: else
6:     Return $\perp$
7: end if
```

Figure 3.2: Protocol $\mathcal{RW}$

```
Initialization $\mathcal{SI}^I(T,\eta)$:
1: for all nodes $v$ in $T$ do
2:     $K[v] \leftarrow T$
3: end for
4: Return $K$
Message generation $\mathcal{SI}^G_K(o,d,\eta_d)$:
1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: $\mathbf{R} \leftarrow \text{ShortRoute}(o, d, K[o])$
3: Return $(Data, \mathbf{R}, d)$
Message routing $\mathcal{SI}^R_K(m,v)$:
1: if $v \neq d$ then
2:     $next \leftarrow \text{FindNext}(\mathbf{R}, v)$
3:     Return $(next, (Data, \mathbf{R}, d))$
4: end if
5: Return $\perp$
```

Figure 3.3: Protocol $\mathcal{SI}$

```
Initialization $\mathcal{S}\emptyset^I(T,\eta)$:
1: for all nodes $v$ in $T$ do
2:     $(pk[v], sk[v]) \leftarrow \text{GenAsymKeyPair}(\eta)$
3:     $K[v] \leftarrow (T, pk, sk[v])$
4: end for
5: Return $K$
Message generation $\mathcal{S}\emptyset^G_K(o,d,\eta_d)$:
1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: $\mathbf{R} \leftarrow \text{ShortRoute}(o, d, K[o])$
3: $S \leftarrow \text{Sign}((Data, \mathbf{R}, o, d), sk[o])$
4: Return $(Data, \mathbf{R}, o, d, S)$
Message routing $\mathcal{S}\emptyset^R_K(m,v)$:
1: $(Data, \mathbf{R}, o, d, S) \leftarrow m$
2: if $v \neq d$ then
3:     if $\text{Verify}((Data, \mathbf{R}, o, d), pk[o], S)$ then
4:         $next \leftarrow \text{FindNext}(\mathbf{R}, v)$
5:         Return $(next, (Data, \mathbf{R}, o, d, S))$
6:     end if
7: end if
8: Return $\perp$
```

Figure 3.4: Protocol $\mathcal{S}\emptyset$

```
Initialization $\mathcal{SR}^I(T,\eta)$:
1: for all nodes $v$ in $T$ do
2:     $(pk[v], sk[v]) \leftarrow \text{GenAsymKeyPair}(\eta)$
3:     $K[v] \leftarrow (T, pk, sk[v])$
4: end for
5: Return $K$
Message generation $\mathcal{SR}^G_K(o,d,\eta_d)$:
1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: $\mathbf{R} \leftarrow \text{ShortRoute}(o, d, K[o])$
3: $S \leftarrow \text{Sign}((Data, \mathbf{R}, o, d), sk[o])$
4: Return $(Data, \mathbf{R}, o, d, S)$
Message routing $\mathcal{SR}^R_K(m,v)$:
1: $(Data, \mathbf{R}, o, d, S) \leftarrow m$
2: if $v \neq d$ then
3:     if $\text{Verify}((Data, \mathbf{R}, o, d), pk[o], S)$ then
4:         $next \leftarrow \text{FindNext}(\mathbf{R}, v)$
5:     else
6:         $next \xleftarrow{\$} \mathcal{N}eig_v$
7:     end if
8:     Return $(next, (Data, \mathbf{R}, o, d, S))$
9: end if
10: Return $\perp$
```

Figure 3.5: Protocol $\mathcal{SR}$

3.4.1 Attacker

Our model deals with an attacker external to the network, who did not compromise any honest node. This entity controls the network links, and is able to intercept, create and manipulate messages. Its goal is to alter how a challenge message is routed. Notice that manipulating communications implies the possibility of forging messages to observe how nodes would react, and to observe which messages are generated in the network. However, since we suppose the attacker is external to the network, it does not have any direct access to the node's memories.

This adversary is modeled as a probabilistic polynomial-time Turing machine. It can query the oracles $\mathcal{P}^G_K$, $\mathcal{P}^R_K$ and $\mathcal{P}^D_K$ a polynomial number of times, but does not have direct access to the array of node memories K. None of its actions can modify K (by hypothesis on the protocol), but the adversary has its own memory, since it is a Turing machine. Note that we provide access to $\mathcal{P}^D_K$ to the adversary, as we are not concerned by confidentiality.

We denote by $\mathcal{A}_{safe}$ the trivial attacker that returns its given input message. We name it "safe" as it is a placeholder attacker that effectively does nothing.

3.4.2 Measuring how routing protocols operate

We define an experiment named $\mathbf{Expt}^{Rt}_{\mathcal{P}}$, which allows us to reason on an adversary's ability to influence how a message is routed.

Definition 3.7 ($\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$)**.** *Let $\mathcal{P}$ be a routing protocol. Let $\mathcal{A}$ be an adversary and $T = \{V, E, f\}$ a topology with $o, d, s, a, b \in V$. We define the experiment $\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ as:*

```
$\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ :
    $K \xleftarrow{\$} \mathcal{P}^I(T, \eta)$
    $m \xleftarrow{\$} \mathcal{P}^G_K(o, d, \eta_d)$
    $m' \xleftarrow{\$} \mathcal{A}^{\mathcal{P}^R_K, \mathcal{P}^G_K, \mathcal{P}^I, \mathcal{P}^D_K}(m, o, d, s, a, b, T)$
    **If** $\mathcal{P}^D_K(d, m') \neq \mathcal{P}^D_K(d, m)$
        $m' \leftarrow m$
    $\mathbf{R} \xleftarrow{\$} \textsc{GenRoute}(m', s, \mathcal{P}^R_K)$
    **Return** $\Phi(\mathbf{R}, a, b)$
```

This experiment runs as follows. First, initialization is done by calling $\mathcal{P}^I(T, \eta)$, which returns the array of node memories K that is used through the experiment. A challenge message m is generated using K, and given to the adversary. The adversary should then change m in a new message m', containing the same data as m. The experiment returns a value $\Phi(\mathbf{R}, a, b)$ with $\mathbf{R}$ a route generated for m' from the node s. Informally, this value is true when the route passes through a before b. We use the equality of depacketed messages as a way to prevent replay attacks: an attacker has no incentive in returning new messages containing random data, as their answer would get replaced by the challenge message, which they could have output in the first place.

For instance, the return value of $\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}_{safe}, T, o, d, s, a, b, \eta, \eta_d)$ models whether a random message m generated by o in destination of d gets routed by a before b when sent first from s, when all those nodes follow the routing protocol $\mathcal{P}$. When we use an arbitrary adversary $\mathcal{A}$, then $\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ represents the same observation, except that m has been tampered with by $\mathcal{A}$ before being routed by s. We remark that $\mathcal{P}\emptyset$ has an interesting property here: for any $\mathcal{A}, T$ and o, d, s, a, b, $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}\emptyset}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)] = 0$.

We then compare two such measures in order to define the *distance* between a tuple protocol, attacker and another.

Definition 3.8 (Distance)**.** *For a topology $T = \{V, E, f\}$ with $o, d, s, a, b \in V$, two adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, two protocols $\mathcal{P}_1$ and $\mathcal{P}_2$, we define the distance as*

$$Dist((\mathcal{P}_1, \mathcal{A}_1), (\mathcal{P}_2, \mathcal{A}_2), T, o, d, s, a, b, \eta, \eta_d) =$$

$$\left|Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}_1}(\mathcal{A}_1, T, o, d, s, a, b, \eta, \eta_d)] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}_2}(\mathcal{A}_2, T, o, d, s, a, b, \eta, \eta_d)]\right|$$

The notion of distance is a way to compare the routes being generated by $(\mathcal{P}_1, \mathcal{A}_1)$ and $(\mathcal{P}_2, \mathcal{A}_2)$, given T and o, d, s, a, b. We now present how to measure observable differences between routing protocols using this experiment.

3.4.3 Routing similarity

We begin by expressing the similarity of routing protocols using $Dist$. We recall that stating that a function $\mu(x) : \mathbb{N} \to \mathbb{R}$ is negligible in x means that for every positive polynomial P there exists an integer I such that for all $x > I$, $\mu(x) < |\frac{1}{P(x)}|$, as given in [Bel02].

Definition 3.9 (Routing protocols similarity)**.** *For a topology $T = \{V, E, f\}$, we say that two protocols $\mathcal{P}_1$ and $\mathcal{P}_2$* route messages similarly *on T if $\forall o, d, s, a, b \in V$,*

$$Dist((\mathcal{P}_1, \mathcal{A}_{safe}), (\mathcal{P}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d)$$

is negligible in η and in η_d.

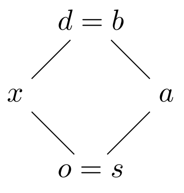


Figure 3.6: Topology T_s

Intuitively, two protocols route messages similarly on a topology if the routes generated for random messages are computationally indistinguishable. for all origins o, destinations d, and senders s. For instance, $\mathcal{RW}$ and $\mathcal{SI}$ are not similar for all topologies, as the latter generates distinguishably shorter routes. However, on a topology T_2 consisting of two connected nodes, they are similar, as messages are either routed to the neighbor if $o \neq d$ and not routed at all otherwise.

Consider for instance $\mathcal{S}\emptyset$ and $\mathcal{P}\emptyset$ on a topology T_s as described in Figure 3.6. We observe that $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe}, T_s, o, d, s, a, b, \eta, \eta_d)] = 0.5$, as there are two shortest routes from o to d and only one reaches a before b. We know that the probability $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}\emptyset}(\mathcal{A}_{safe}, T_s, o, d, s, a, b, \eta, \eta_d)] = 0$, since null routes will never reach any of the two nodes. Therefore, $Dist((\mathcal{S}\emptyset, \mathcal{A}_{safe}), (\mathcal{P}\emptyset, \mathcal{A}_{safe}), T_s, o, d, s, a, b, \eta, \eta_d) = 0.5$, which means that those two protocols are not similar on T_s, and we can conclude that $\mathcal{P}\emptyset$ and $\mathcal{S}\emptyset$ are not *similar on every topology*, as they differ on at least T_s. Notice that this definition does not include attackers: two protocols routing messages similarly may not behave in the same way in presence of an *active* adversary. This allows us to show that a secure version of a protocol is similar to its original counterpart. For instance, $\mathcal{SR}$, $\mathcal{S}\emptyset$ and $\mathcal{SI}$ are all similar.

3.4.4 Incorruptibility of a protocol

We propose a measure which evaluates whether an attacker can alter a message m into another message m' in order to make its routing distinguishably different. We call this measure the *incorruptibility* of a routing protocol, the related advantage is denoted $\mathbf{Adv}^{INC}_{\mathcal{P}}$, and we define it using $Dist$.

Definition 3.10 ($\mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$)**.** *For an adversary $\mathcal{A}$, a topology $T = \{V, E, f\}$ with five nodes $o, d, s, a, b \in V$, we define $\mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ as:*

$$\mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = Dist((\mathcal{P}, \mathcal{A}), (\mathcal{P}, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d)$$

Definition 3.11 (Incorruptible protocol)**.** *If for any adversary $\mathcal{A}$, any topology $T = \{V, E, f\}$, and any five nodes $o, d, s, a, b \in V$, $\mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ is negligible in η and in η_d, we say that $\mathcal{P}$ is* incorruptible*.*

Informally, a protocol is corruptible if the adversary's actions can result in a distinguishably different routing of messages from the untouched one. For instance, the $\mathcal{P}\emptyset$ protocol is incorruptible, as it always generates null routes. Similarly, $\mathcal{RW}$ is incorruptible: it is not influenced by any information contained in the messages, and so intuitively an attacker which can only alter the content of a message is not able to influence in any way how a message is routed.

However, this definition is too restrictive for some protocols that intuitively cannot be attacked, such as $\mathcal{S}\emptyset$. Most protocols whose behavior depends on the message contents can be corrupted, as an attacker can use that dependency in order to differ from the safe behavior of the protocol. We now provide an attacker for $\mathcal{S}\emptyset$ to illustrate this reasoning, and in the next subsection, we provide a generalization of the incorruptibility advantage to answer those concerns.

In order to show the corruptibility of $\mathcal{S}\emptyset$ (which is described in Figure 3.4), we use the adversary $\mathcal{A}_{zero}$ that takes as input the message $m = (Data, \mathbf{R}, o, d, S)$, and returns the altered $m' = (Data, \mathbf{R}, o, d, 0)$. Intuitively, this attacker destroys the signature S

of the message m, which forces the protocol to drop it at the next hop. This behavior differs significantly from how the original m would have been routed.

We recall the definition of $\mathbf{Adv}^{INC}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, T, o, d, s, a, b, \eta, \eta_d) =$

$$\left|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe}, o, d, s, a, b, \eta, \eta_d)]\right|$$

We omit T, o, d, s, a, b, and η, η_d from the parameters list when it is clear from the context. We are first interested in the left part of this subtraction. $\mathcal{A}_{zero}$ changes the signatures of messages it is given. Let us consider what happens with an altered message m'_{zero} (containing S_{zero}) and its corresponding $\mathbf{R}_{zero}$, generated in the $\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)$ experiment. We separate the case where S holds and its opposite. We have:

$$\begin{aligned}
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] = \\
&\qquad Pr[\Phi(\mathbf{R}_{zero}, a, b) | \textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \\
&\qquad Pr[\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] + \\
&\qquad Pr[\Phi(\mathbf{R}_{zero}, a, b) | \neg\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \\
&\qquad Pr[\neg\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})]
\end{aligned}$$

If we assume that $\mathcal{S}\emptyset$ uses a secure (UF-CMA in the sense of [GMR88]) signature scheme $\mathcal{S}$ of security parameter η, then we know that the probability ϵ of the signature being forged by an intruder (*i.e.* $\textsc{Verify}((Data, \mathbf{R}, o, d), pk[o], S)$ holds) becomes *negligible* in η. Therefore:

$$\begin{aligned}
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] = \\
&\qquad Pr[\Phi(\mathbf{R}_{zero}, a, b) | \textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \epsilon + \\
&\qquad Pr[\Phi(\mathbf{R}_{zero}, a, b) | \neg\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times (1 - \epsilon)
\end{aligned}$$

We first consider the case where the signature is invalid. Consider the oracle $\mathcal{S}\emptyset^R_K$ described in Figure 3.4. If the signature of the message is not valid, then the message is dropped. Therefore, all the routes generated for m' in this context are equal to the empty route $[\,]$. We know that $\Phi([\,], a, b)$ is always false for any a and b. We can therefore conclude that the experiment returns 0 with a probability $(1 - \epsilon)$.

$$\begin{aligned}
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] = \\
&\qquad Pr[\Phi(\mathbf{R}_{zero}, a, b) | \textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \epsilon + \\
&\qquad 0 \times (1 - \epsilon)
\end{aligned}$$

We denote p the probability of the experiment returning 1 when the signature is valid. Going back to the advantage, we have:

$$\mathbf{Adv}^{INC}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}) = \left|(0 + \epsilon \times p) - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]\right|$$

The first part of the subtraction is negligible in η, as p is a probability and ϵ is negligible in η. However, $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe}, T, o, d, s, a, b)]$ may not be negligible, depending on T and o, d, s, a, b (as we have shown in Figure 3.6). Intuitively, without attacker interference, $\mathcal{S}\emptyset$ actually routes messages to their destination, which ensures the existence of such nodes. Therefore, there exist some topologies T (T_s being one of them) where $\mathbf{Adv}^{INC}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, T, o, d, s, a, b, \eta, \eta_d)$ is not negligible in η and in η_d, and so $\mathcal{S}\emptyset$ is not *incorruptible on all topologies.*

3.4.5 Bounded corruptibility

We generalize the notion of corruptibility to a definition using two reference protocols.

We define another advantage, called $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}$. It follows the same principle as $\mathbf{Adv}^{INC}_{\mathcal{P}}$, but instead of considering how an attacker can force $\mathcal{P}$ to behave differently, we consider how it can be corrupted to the outside of a reference routing *interval*, defined by the safe execution of two protocols $\mathcal{B}_1$ and $\mathcal{B}_2$ on T, measured for the parameters o, d, s, a, b.

Definition 3.12 ($\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$)**.** *Let $\mathcal{A}$ be an attacker, and $T = \{V, E, f\}$ a topology with nodes $o, d, s, a, b \in V$. We consider a protocol $\mathcal{P}$, which is compared with two protocols $\mathcal{B}_1$ and $\mathcal{B}_2$. We define $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ as:*

$$\begin{aligned}
max(\; & Dist((\mathcal{B}_1, \mathcal{A}_{safe}), (\mathcal{B}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d),\\
& Dist((\mathcal{P}, \mathcal{A}), (\mathcal{B}_1, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d),\\
& Dist((\mathcal{P}, \mathcal{A}), (\mathcal{B}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d) \;)\\
) \; & - Dist((\mathcal{B}_1, \mathcal{A}_{safe}), (\mathcal{B}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d)
\end{aligned}$$

Informally, $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ is a measure of the maximal distance the behavior of $\mathcal{P}$ attacked by $\mathcal{A}$ can get from the outside of the interval determined by the safe behavior of $\mathcal{B}_1$ and $\mathcal{B}_2$. Remark that if the attacked protocol's behavior is in the interval, then the advantage is 0.

Definition 3.13 (Bounded corruptibility)**.** *If for any adversary $\mathcal{A}$, for any topology T, and for all o, d, s, a, b, $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ is negligible in η and in η_d, we say that $\mathcal{P}$'s corruptibility is bounded between $\mathcal{B}_1$ and $\mathcal{B}_2$.*

Remark that this definition has some interesting properties:

- $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = \mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_2,\mathcal{B}_1}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$:
  The bounds for bounded corruptibility are commutative.

- $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{P},\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = \mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$:
  Stating that a protocol is bounded between itself and itself is the same as stating its incorruptibility.

- $Dist(\mathcal{B}_1, \mathcal{B}_2, T, o, d, s, a, b, \eta, \eta_d) = 0 \Rightarrow$
  $\forall \mathcal{B}_3, \mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_3}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = \mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_2,\mathcal{B}_3}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$:
  If two protocols route messages identically, then those protocols are equivalent for bounding purposes.

Example: Bounded corruptibility of $\mathcal{S}\emptyset$

We try to bound $\mathcal{S}\emptyset$ using itself and $\mathcal{P}\emptyset$. We assume that $\mathcal{S}\emptyset$ uses a secure (UF-CMA in the sense of [GMR88]) signature scheme $\mathcal{S}$ of security parameter η. By definition, $\mathbf{Adv}^{BINC}_{\mathcal{S}\emptyset,\mathcal{S}\emptyset,\mathcal{P}\emptyset}(\mathcal{A}, T, o, d, s, a, b)$ equals:

$$\begin{aligned} max(&Dist((\mathcal{S}\emptyset, \mathcal{A}_{safe}), (\mathcal{P}\emptyset, \mathcal{A}_{safe})),\\ &Dist((\mathcal{S}\emptyset, \mathcal{A}), (\mathcal{S}\emptyset, \mathcal{A}_{safe})),\\ &Dist((\mathcal{S}\emptyset, \mathcal{A}), (\mathcal{P}\emptyset, \mathcal{A}_{safe})),\\ )-&Dist((\mathcal{S}\emptyset, \mathcal{A}_{safe}), (\mathcal{P}\emptyset, \mathcal{A}_{safe})) \end{aligned}$$

By using the fact that $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}\emptyset}(\mathcal{A}_{safe})] = 0$, we get the following:

$$\begin{aligned} max(&|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]|,\\ &|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})]|,\\ &|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})]|\\ )-&|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]| \end{aligned}$$

Probabilities are positive, which allows us to remove some of the absolute values. We also rewrite $|a|$ as $max(a, -a)$ in the last case, to remove all absolute values:

$$\begin{aligned} max(&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})],\\ &Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})],\\ &Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})],\\ &Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})]\\ )-&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] \end{aligned}$$

We include the subtraction in the maximum and simplify further:

$$\begin{aligned} max(0,\ &0 - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})],\\ &Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - 2Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})],\\ &Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]) \end{aligned}$$

We know that a probability is always between 0 and 1 included. Therefore, we know that $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - 2Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] \leq Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]$, so we can remove the right part of the inequation from the maximum.

$$\begin{aligned} max(0,\ &0 - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})],\\ &Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]) \end{aligned}$$

Similarly, $0 - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] \leq 0$. We can therefore simplify the maximum to:

$$max(0, Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})])$$

We want to prove that this is negligible in η and in η_d for all $T = \{V, E, f\}$ and $o, d, s, a, b \in V$. We reformulate this as $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] \leq \epsilon$, with ϵ negligible in η and in η_d. Looking at the experiment, this means that

$$Pr[\Phi(\textsc{GenRoute}(m', s, \mathcal{S}\emptyset^R_K), a, b))] - Pr[\Phi(\textsc{GenRoute}(m, s, \mathcal{S}\emptyset^R_K), a, b))] \leq \epsilon$$

The difference between these probabilities is null unless the routes generated from attacked messages m' and the routes generated from m are different. Looking at the route generation process, the only factors influencing the oracle $\mathcal{S}\emptyset_K^R$ (defined in Figure 3.4) are the validity of the signature, the contents of $\mathbf{R}$, and the identity of the receiver (which is not influencable by the attacker). Furthermore, all those solutions require $\mathcal{S}\emptyset_K^D(m') = \mathcal{S}\emptyset_K^D(m)$, as otherwise the experiment would have run as if the attacker output m.

We therefore know that the advantage is null unless the attacker made the signature invalid, or it altered the route stored in the message and the signature is still valid. In that first case, the invalid signature forces the message to be dropped. Consequently, the generated route is equal to the empty route $[\,]$. Since $\Phi([\,], a, b) = 0$, then the probability of the experiment returning true is null, and so this strategy does not provide an higher advantage. In the other case, the attacker managed to alter the route stored in the message, while keeping the same data, and keeping the signature valid. To have a valid signature for an altered message, the attacker has either forged it, or recovered it from $\mathcal{S}\emptyset_K^G(o, d)$. Note that it cannot create a valid signature for a key it created, as that key would not be present in any node's K.

- The attacker can try to forge the signature. We proceed by assuming it manages to forge or guess the signature with a probability p_F when running on T', o', d', s', a', b'. This adversary $\mathcal{A}$ could also be used to build an adversary $\mathcal{A}_S$ who breaks the UF-CMA experiment with probability p_F. $\mathcal{A}_S$ needs to emulate $\mathcal{S}\emptyset$ on T', creating its own initialization on the network except for the node o', who uses the challenge's key. As $\mathcal{A}_S$ does not know the keys of o', it should use the chosen-plaintext oracle and verification oracle provided in the UF-CMA experiment to simulate its knowledge. $\mathcal{A}$ will therefore be in the right simulated context for $\mathbf{Expt}_{\mathcal{S}\emptyset}^{Rt}(\mathcal{A}, T', o', d', s', a', b')$, and will be provided a message m originating from o' (which costs $\mathcal{A}_S$ one call to its chosen-plaintext oracle). By assumption, this adversary will therefore return a message m' containing a valid forged signature with probability p_F. In the end, given an adversary $\mathcal{A}$ for $\mathbf{Expt}_{\mathcal{S}\emptyset}^{Rt}$ who makes q_G queries to $\mathcal{S}\emptyset_K^G$ and q_R queries to $\mathcal{S}\emptyset_K^R$ to forge signatures with probability p_F, we built an adversary $\mathcal{A}_S$ making $q_G + 1$ queries ($\mathcal{A}$'s queries, plus one to create the setting) to the chosen-plaintext oracle, and q_R queries to the verification oracle such that $\mathbf{Adv}_S^{UFCMA}(\mathcal{A}_S, \eta) = p_F$. As we supposed the signature scheme $\mathcal{S}$ secure, we therefore know that p_F is negligible in η.

- The attacker can also try to obtain the signature without forging it. The only source of valid signatures for $\mathcal{A}$ is $\mathcal{S}\emptyset_K^G(o, d)$, but this oracle provides packets containing random data. Therefore, given the attacker does q_G queries, the probability of obtaining a valid packet m' verifying $\mathcal{S}\emptyset_K^D(m') = \mathcal{S}\emptyset_K^D(m)$ is $\left(\frac{2^{\eta_d}-1}{2^{\eta_d}}\right)^{q_G}$, which is negligible in η_d.

Summing all the possibilities before, we get:

$$Pr[\Phi(\textsc{GenRoute}(m', s, \mathcal{S}\emptyset_K^R), a, b))] - Pr[\Phi(\textsc{GenRoute}(m, s, \mathcal{S}\emptyset_K^R), a, b))]$$

$$\leq \mathbf{Adv}_{S}^{UFCMA}(\mathcal{A}_S, \eta) + \left(\frac{2^{\eta_d}-1}{2^{\eta_d}}\right)^{q_G}$$

Therefore, we can say that for all adversaries $\mathcal{A}$ making a polynomial number of queries to $\mathcal{S}\emptyset_K^G(o,d)$ and $\mathcal{S}\emptyset_K^R(m,v)$, $\mathbf{Adv}_{\mathcal{S}\emptyset,\mathcal{S}\emptyset,\mathcal{P}\emptyset}^{BINC}(\mathcal{A},T,o,d,s,a,b)$ is negligible in η and in η_d.

Protocols from Chapter 2

We presented a few protocols in the previous chapter. Gradient based routing (GBR) routes messages towards a sink, using a breadth-first tree. If we consider the tree building process is part of the initialization of the protocol, this protocol can be seen as a static routing scheme, where nodes routes all their messages towards a specific neighbor, no matter what the contents of these messages are. As we discussed before, such protocols are incorruptible, as the attacker can only try to influence the routing by changing messages. A geographical routing algorithm with a single, predetermined destination would be in the same case.

However, such a modelization does not analyze how the routing information is obtained, and this is precisely the weak point of these protocols. The analysis of these mechanisms, and of protocols such as SR3 would require to store memory which can be modified in the nodes, and our model does not allow this for now. This is further discussed in the conclusion of this chapter.

3.4.6 Covert attackers

We now consider an attacker who is not allowed to reroute or alter messages before the challenge. We denote such an attacker a *covert attacker*, as opposed to an *overt attacker* like the one previously described. We model that type of attacker with an experiment similar to the overt case. In our context, a covert attacker does not send any altered packets in the network before the challenge itself. Instead of being able to craft messages and reroute them, this attacker may only overhear what happens in the network without interfering.

Modelization of covert attackers

To model this, we replace in $\mathbf{Expt}_{\mathcal{P}}^{Rt}$ the accesses to $\mathcal{P}_K^R$ and to $\mathcal{P}_K^G$ with an access to GenRouteOracle$(\cdot,\cdot,\mathcal{P}_K^G,\mathcal{P}_K^R)$.

Definition 3.14 (GenRouteOracle$(o,d,\mathcal{P}_K^G,\mathcal{P}_K^R)$)**.** *Given two node identifiers o and d, a message generation oracle $\mathcal{P}_K^G$ and a routing oracle $\mathcal{P}_K^R$ both initialized with K, we define* GenRouteOracle$(o,d,\mathcal{P}_K^G,\mathcal{P}_K^R)$ *the function that creates a route for a message m_1 generated by o and intended for d. This function then operates similarly to* GenRoute*, and returns both the route generated for that initial message $[h_1,\ldots,h_n]$, together with the eventual successive alterations at each hop $[m_1,\ldots,m_{n-1}]$.*

This way, the attacker can observe what would happen in the network if everything goes according to the protocol, but cannot see how an arbitrary node reacts to an arbitrary message. We denote this new experiment $\mathbf{Expt}_{\mathcal{P}}^{Rt-P}$.

Definition 3.15 ($\mathbf{Expt}_{\mathcal{P}}^{Rt-P}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$)**.** *Let $\mathcal{P}$ be a routing protocol. Let $\mathcal{A}$ be an adversary and $T = \{V, E, f\}$ a topology with $o, d, s, a, b \in V$. We define $\mathbf{Expt}_{\mathcal{P}}^{Rt-P}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ as:*

```
$\mathbf{Expt}_{\mathcal{P}}^{Rt-P}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ :
    $K \xleftarrow{\$} \mathcal{P}^I(T, \eta)$
    $m \xleftarrow{\$} \mathcal{P}_K^G(o, d, \eta_d)$
    $m' \xleftarrow{\$} \mathcal{A}^{\textsc{GenRouteOracle}(\cdot,\cdot,\mathcal{P}_K^G,\mathcal{P}_K^R),\mathcal{P}^I,\mathcal{P}_K^D}(m, o, d, s, a, b, T)$
    **If** $\mathcal{P}_K^D(d, m') \neq \mathcal{P}_K^D(d, m)$
        $m' \leftarrow m$
    $\mathbf{R} \xleftarrow{\$} \textsc{GenRoute}(m', s, \mathcal{P}_K^R)$
    **Return** $\Phi(\mathbf{R}, a, b)$
```

Note that an adversary $\mathcal{A}$ in the overt version of the incorruptibility experiment $\mathbf{Expt}_{\mathcal{P}}^{Rt}$ has full access to $\mathcal{P}_K^R$, $\mathcal{P}_K^G$, and T. Consequently, it is able to emulate the oracle $\textsc{GenRouteOracle}(o, d, \mathcal{P}_K^G, \mathcal{P}_K^R)$. Therefore, for all attackers $\mathcal{A}_S$ for the covert incorruptibility experiment, there exists an attacker $\mathcal{A}_O$ for the overt incorruptibility experiment such that $\mathbf{Expt}_{\mathcal{P}}^{Rt-P}(\mathcal{A}_P, T, o, d, s, a, b, \eta, \eta_d) = \mathbf{Expt}_{\mathcal{P}}^{Rt}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ (*i.e.*, as intuitively expected, the covert version of the experiment is harder for the adversary than the overt version).

Initialization $\mathcal{S}\$^I(T, \eta)$:

```
1: **for** all nodes $v$ in $T$ **do**
2:     $(pk[v], sk[v]) \leftarrow \textsc{GenAsymKeyPair}(\eta)$
3:     $Seed \xleftarrow{\$} \{0,1\}^{\eta_d}$
4:     $K[v] \leftarrow (T, pk, sk[v], Seed)$
5: **end for**
6: **Return** $K$
```

Message generation $\mathcal{S}\$_K^G(o, d, \eta_d)$:

```
1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: $mid \xleftarrow{\$} \{0,1\}^{\eta_d}$
3: $E \leftarrow \textsc{AsymEncrypt}((Data), pk[d])$
4: **Return** $(E, d, mid)$
```

Message routing $\mathcal{S}\$_K^R(m, v)$:

```
1: $(E, d, mid) \leftarrow m$
2: **if** $v \neq d$ **then**
3:     $Candidates \leftarrow \textsc{NeighborsCloserTo}(d)$
4:     **if** $Candidates \neq \emptyset$ **then**
5:         $next \xleftarrow{\$(E,mid,Seed)} Candidates$
6:         $mid \xleftarrow{\$} \{0,1\}^{\eta_d}$
7:         **Return** $((E, d, mid), next)$
8:     **end if**
9: **end if**
10: **Return** $\perp$
```

Figure 3.7: Protocol $\mathcal{S}\$$

In the same way as before, we define $Dist_P$, $\mathbf{Adv}_{\mathcal{P}}^{INC-P}$ and $\mathbf{Adv}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}^{BINC-P}$. We refer to an attacker in the overt (resp. covert) version of the incorruptibility experiment as an *overt* (resp *covert*) attacker.

Example

To illustrate this security notion, we present a new protocol, called $\mathcal{S}\$$ (shortest-seeded). We first introduce some new notation, then an informal description of the algorithm, and finally the incorruptibility proof. For this protocol, we denote by $a \xleftarrow{\$(c)} b$ the selection of an element among the set b, based on a random oracle given the input c. The result of that selection is stored in a. Informally, this selection purely depends on c: the output is selected uniformly at random the first time it is called on a given

c, but then it always returns the same element a for the couple (b, c). We denote by $\textsc{AsymEncrypt}(Data, pk[d])$ the results of the asymmetric encryption of $Data$ using the public key of d.

The algorithm is described in Figure 3.7, and we provide an informal description of it. The protocol works as follows: messages contain a cryptographically protected destination field, and nodes route them at random to one of their neighbors closer to the destination. The specificity of $\mathcal{S}\$$ is that its random choices are based on three values: the message identifier mid, a node-specific random seed $Seed$, and an encrypted message field E. Those values determine which neighbor will be chosen. Furthermore, after each routing choice, mid is set to a fresh random value. E is specific to each message, but does not change during its lifetime, and $Seed$ is drawn at random by each node during the initialization phase.

We claim that $\mathcal{S}\$$'s corruptibility against a covert attacker is bounded between itself and $\mathcal{S}\emptyset$, but its corruptibility against an overt attacker is not. We first show the overt attacker that is able to corrupt the protocol, and then go on to give an intuition of its uncorruptibility against covert attackers.

Corruptible in the overt case Consider the topology T_s described in Figure 3.6, in the overt attack case. We recall that:

$$\mathbf{Adv}^{INC}_{\mathcal{S}\$}(\mathcal{A}, T_s, o, d, s, a, b, \eta, \eta_d)$$

$$= |Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\$}(\mathcal{A}, T_s)] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\$}(\mathcal{A}_{safe}, T_s)]|$$

In the case of $\mathcal{A}_{safe}$, the protocol selects uniformly at random either a or x to route the message, and the next node always choses b: therefore, the probability of going through a before b is $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\$}(\mathcal{A}_{safe}, T_s)] = 0.5$.

$$\mathbf{Adv}^{INC}_{\mathcal{S}\$}(\mathcal{A}, T_s) = |Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\$}(\mathcal{A}, T_s)] - 0.5|$$

Consider how an attacker may change the routes generated by $\mathcal{S}\$^R_K(m, v)$. The next hops are randomly determined at each hop, based on E, mid, and $Seed$. First of all, all the node's $Seed$ are never revealed, and so the attacker cannot learn it, whether covert or overt. However, both E and mid are known values for him.

Let us first look into E. The attacker does not control the random generation of $Data$. As both these intruders cannot obtain the secret key of a node $d \in V$, we know it is not possible for an intruder to recover $Data$. However, both covert and overt attackers can clearly modify mid. As the random selection of the next hop for a given node is dependent on mid, they can alter how the corresponding message is going to be routed.

An overt attacker can indeed, with a message (E, d, mid) and a next hop s, query $\mathcal{S}\$^R_K((E, d, mid'), s)$ with different values for mid' (while E and $Seed$ stay the same). The attacker thus obtains several messages, and knows their destinations (which are drawn randomly), based on the mid he chooses. This way, it obtains a pool of messages based on the number of routing queries made (we denote that number q), and therefore he is able to choose one of its altered messages depending on the destination he wants.

Now, with the algorithm described above, $\mathcal{A}$ is able to try various messages $m'_1..m'_n$ that will be independently routed. Let us suppose the adversary wants to send the message to a, and call that adversary $\mathcal{A}_a$. With q requests to $\mathcal{S}\$^R_K((E, d, mid'), s)$, $\mathcal{A}$ can produce a message m' such that this message will be sent to a. The probability of $\mathcal{A}_a$ not obtaining such a message in q queries is the same as only obtaining x-routed messages, that is, $1 - \frac{1}{2^q}$.Therefore:

$$\mathbf{Adv}^{INC}_{\mathcal{S}\$}(\mathcal{A}_a, T_s) = |Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\$}(\mathcal{A}_a, T_s)] - 0.5]| = 1 - \frac{1}{2^q} - 0.5 = 0.5 - \frac{1}{2^q}$$

and so, since that advantage is not negligible in η and η_d, the protocol $\mathcal{S}\$$ is not incorruptible against an overt attacker.

Incorruptible in the covert case However, for a covert attacker, this is not possible. The attacker does not have direct access to $\mathcal{S}\$^R_K$, and so cannot try how various mid' influence a challenge message's routing. This is also not observable using naturally generated messages using $\textsc{GenRouteOracle}(\cdot, \cdot, \mathcal{S}\$^G_K, \mathcal{S}\$^R_K)$: as each message only has a negligible probability of containing the same *Data*, E is going to be almost always different for all of them. Furthermore, the attacker may try changing E, which will only result in a message containing a random, unknown *Data*, which will cause the experiment to replace that answer message by the initial challenge m. Similarly, if the attacker changes the destination field d, then the decryption of E will also be something else than *Data*, also resulting in the replacement of the attacker's answer by m. Both these options are not beneficial for $\mathcal{A}$, in the sense that they do not result in a higher advantage.

We therefore conclude that in the covert case, the attacker has no way to knowingly influence the next hop of the message, but can force it to be dropped (by changing the message format). Such behavior would mean that the protocol's corruptibility is bounded between itself and $\mathcal{P}\emptyset$ (as we have shown for $\mathcal{S}\emptyset$ above). We therefore expect the $\mathcal{S}\$$ protocol's corruptibility to be bounded between itself and $\mathcal{P}\emptyset$ against a covert attacker.

3.5 Conclusion

In this chapter, we have presented a notion of routing security, named *incorruptibility.* Incorruptibility is a quantitative measure, based on the ability of an attacker to influence how messages are routed. We provide several example protocols in our modelization, and proved that some of them are indeed incorruptible. However, some protocols do not quite correspond to this definition: after showing why one of them is corruptible, we propose the notion of *bounded corruptibility*, a generalization of the previous measure. This more accommodating notion allows us to prove that some routing protocols can only be influenced between given limits, which are exprimed using routing protocols.

We finally provide a proof of the bounded corruptibility of one of our example routing protocols, and present a variant of those two notions for a covert attacker who do not alter or reroute messages before the challenge, as opposed to the overt attacks presented before.

Perspectives

There are several ways this work could be expanded.

We modeled routes as arrays: using trees would allow modelization of the emission of several concurrent messages, which in turn could model broadcasts. However, our opinion is that broadcasts by themselves will not be a significant improvement without allowing changes in the array of node memories K after initialization.

Modeling node state changes is possible, but this would in turn require more complex proof techniques to obtain results, as messages will influence how the following ones are routed. This would be an important step towards modeling SR3 or PRDGBR, for instance, and more generally to expand this model towards more complex protocols.

When considering bounded corruptibility, some bounds have specific properties. Proving that a protocol $\mathcal{P}$'s corruptibility is bounded by protocols $\mathcal{P}\emptyset$ and itself provides an insight in the protocol: this bound means that in any situation, an outside attacker can only cause $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A})]$ to be smaller than $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}_{safe})]$. This represents the fact that the attacker cannot *increase* the probability for a message to reach a target node before another, when compared to the safe behavior of $\mathcal{P}$. As a consequence of this, we know that it is not able to force messages to deviate from their route, which was the case in both the incidents we presented in the introduction.

Furthermore, one can wonder whether a protocol $\mathcal{P}\!\uparrow$ such that $\forall \mathcal{P}$, $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A})] \leq Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}\uparrow}(\mathcal{A})]$ exists ($\mathcal{P}\emptyset$ is the other extreme). Building such an upper bound is useful because showing that a protocol is bounded between itself and $\mathcal{P}\!\uparrow$ means that an adversary's interference can only increase the probability of a message reaching one given node before another. This property would allow one to show that a protocol guarantees the eventual delivery of messages, since the attacker's actions can only increase the probability of the message getting somewhere. We are still wondering if such a protocol can be written in our formalism.

Insider attacks on routing protocols suppose one or more nodes in the network are controlled by the attacker. To model this, our first intuition was to allow the attacker some degree of access to K, for instance $K[s]$ (as s represents the last hop before attacker alteration of m). However, this simple modification of the game does not work because of the $\mathcal{P}^D_K(d, m') \neq \mathcal{P}^D_K(d, m)$ check: if the attacker has enough access to K, most protocols get trivially broken in this model as the attacker can create a completely new message containing the data of its choice. We argue that this attack based on building from scratch another packet is not meaningful. An attacker as we model it could always replay another message and it will certainly be routed differently from m. Our goal is to ensure the attacker's m' is based on m, without blocking any legitimate alteration of the message that would change its route.

Finally, we chose to restrict this model to static topologies. It may be interesting to consider dynamic topologies, which correspond better to what may be actually found in a network, either because of varying wireless transmission quality, or because of an intruder actively disrupting the connections. To do this, the game could be modified so that an attacker is able to actively choose the topology used by the network during initialization and evaluation of the challenge message.

Chapter 4

Inputs of intrusion detection systems

To secure Wireless Ad-hoc Networks (WANET) against malicious behaviors, three components are needed: prevention, detection, and response. In this chapter, we focus on Intrusion Detection Systems (IDS) for WANET. We classify the different inputs used by the decision process of these IDS, according to their level of required cooperation, and the source of their data. We then propose InDICE, a decision aid which allows automated discovery of undetectable attacks for an IDS, according to the inputs in use. Finally we apply our framework to discover weaknesses in two existing IDS.

Pour sécuriser les réseaux ad hoc sans-fil (WANET) contre les comportements malicieux, trois composants sont nécéssaires : de la prévention, de la détection, et des mécanismes de réponse. Dans ce chapitre, nous étudions les systèmes de détection d'intrusions (IDS) pour WANET, et plus spécifiquement les sources de données utilisées pour leurs mécanismes de décision. Nous classifions celles-ci en fonction du niveau de coopération qu'elles requièrent, et en fonction de l'origine de leurs données. Nous proposons ensuite InDICE, un outil d'aide à la décision qui étant donné un IDS, permet de découvrir automatiquement quelles attaques seront indétectables par les sources de données qu'utilise cet IDS. Enfin, nous utilisons cet outil pour découvrir deux vulnérabilités dans des IDS de la littérature.

Contents

4.1 Introduction

4.1.1 Context

In the previous chapters, we presented a secure routing protocol and a model to verify neighborhood detection protocols. Both these components exist to *prevent* attacks. However, when hardening a wireless network, one may discover flaws which cannot be repaired immediately, and thus prevention becomes impossible. For instance, Wi-Fi has such a flaw since more than ten years. Wi-Fi (IEEE 802.11) is a combination of a medium access control and a link layer protocol, that allows nodes to communicate after some mutual authentication. When a client wants to deauthenticate, it sends to his access point a special frame which effectively disconnects the client. In 2003, an attack on this process was discovered [BS03], which uses the fact that this frame is not authenticated. Thus, an attacker can spoof the client's MAC address and send deauthentication frames on its behalf, effectively disconnecting the client against its will. For a more durable effect, the attacker can resend this frame each time the client authenticates, which prevents the client from maintaining authentication. Note that this attack is now well-known, and very easy to run since all the software required is publicly available on the internet[1].

Hardening a network against this attack is very difficult, as there was no official fix for this until 2009[2]. But this solution is only partial, as some Wi-Fi enabled devices may not implement it, and replacing all of them may not be a viable solution because of the cost. On the other hand, that attack has a recognizable footprint: clients are not expected to repeatedly break connection just after establishing it. The recommended course of action to counter such an attack is to automatically detect it using a network intrusion detection system (IDS), locate the device emitting the deauthentication frames, and then prevent the attacker's emissions from reaching the AP, either by electromagnetically shielding it, or by recovering the attacker's device. This example illustrates that to improve the security of ad-hoc wireless networks, one needs to add mechanisms that *react* to attacks.

Similarly, the acknowledgment routing attack against SR3 we described in Section 2.5.3 would be very costly to correct for the routing protocol, as it requires authentication of each neighbor of a node (and therefore, establishment of shared secrets). However, if we take a step back from the routing layer, the easiest solution to this problem is to add a simple IDS running on each node. It is both easy and cheap to detect anomalous acknowledgment routing loops, and to communicate some sort of alert to the sink using SR3's message routing.

In this chapter, we consider *intrusion detection systems* (IDS), which are software designed to detect malicious behavior, and in some cases trigger an *intrusion response system* after a detection. The goal of these systems is to detect attackers and then mitigate their effects, either through alerting an human operator, or with automated countermeasures. Then, the traces produced by an IDS can be used to investigate the systems which were used to perform the attack, which should be hardened to prevent further damage.

The evaluation of IDS is usually done experimentally, by using replayed packets which are either handcrafted or observed in the wild, and observing which of the attacks are detected. For instance, in [LFG$^+$00], the authors present a DARPA-mandated evaluation of several IDS from 1998, and they also provide the corresponding dataset. This evaluation and the subsequent one that happened the following year were then critiqued in [McH00], which raises several concerns about this particular evaluation. In this critique, the author argues that the use of a specific handcrafted dataset may not be representative of the real performances of the IDS, and that the metric which has been used for the evaluation artificially increases the scores of some IDS. Although this critique was specifically targeted at the DARPA evaluation, the author also argues that

[1]The `aircrack-ng` package is the best-known software for all WiFi-related attacks, including deauthentication.

[2]The IEEE 802.11w amendment has been approved in 2009. This amendment authenticates the problematic frames and consequently prevents the attack. However, it is not yet commonly deployed.

such concerns are valid for most experimental evaluations of intrusion detection systems.

Furthermore, an experimental evaluation has implicit limits, as it does not specify how the intruder node becomes part of the network, what is the attacker achieving with that attack, nor what exactly the IDS is trying to prevent. This lack of precision can lead to undiscovered flaws in the network. It is important to formulate the properties that the IDS tries to achieve in a clear way, to define the intruder model, to model the protocols and to state the network assumptions, in a formal framework. Then, we can use formal methods, in order to systematically find flaws in an IDS. This process is similar to what has been done in the last years in cryptographic protocol analysis [BCM11], and our goal is to provide the first components to enable the use of such methods in the context of IDS.

4.1.2 Contribution

Our contribution is threefold.

- We survey different inputs that an IDS decision process can use. We base our analysis on two axis: the degree of cooperation required to use that input, and what is being monitored. We also give examples of mechanisms from existing IDS to illustrate our classification.
- We develop a formal model to evaluate such systems, based on *anomalies*, which are the results of attacker behavior. We propose some deduction rules (around 40) expressing how the combination of certain anomalies allows an attacker to build more complex attacks. These rules model the logical steps needed for constructing a specific attack. Then, depending on inputs used by an IDS, we determine whether an attacker can mount an attack without being detected. We also provide a prototype of our formal framework, named InDICE.
- Using InDICE, we analyze two IDS from the literature, [dSMR$^+$05] and [OM05b]. We show that it is not possible to fool the first IDS in presence of restricted intruders. However, by relaxing some of the hypothesis, we discover some weaknesses in the IDS. For the second IDS, we found undetectable attacks using intruder nodes which are able to control the directivity of their wireless signals. Finally, we propose some modifications for these IDS in order to prevent these attacks. Overall, using our approach, it is easy to compare the inputs of an IDS to others from the literature, and to find its flaws and possible improvements.

4.1.3 Related work

Previous classification of IDS

Intrusion detection systems are usually classified with two main characteristics, which evolved from the seminal work in [DDW99]. The first characteristic expresses what is being observed, and contains two categories: *host-based IDS* monitor the internal workings of one node, while *network-based IDS* search for signs of malicious activity in the network. The second characteristic is based on the method used to detect intrusions: *signature-based* detection uses a database of known patterns, *anomaly-based* detection compares the behavior to a known normal behavior, and *specification-based* IDS deal with the compliance of nodes to a given specification. This second aspect is not considered in this chapter. Our classification can be seen as a refinement of the first category. Instead of having two broad categories, we separate the different inputs that an IDS uses and we determine the cooperation level and the sources of the data being monitored.

Analyzing anomalous events to build IDS

In [HL04], the authors provide a specification-based IDS for the AODV routing protocol, based on an extended finite state automaton for modeling the protocol. In their paper, attacks cause various *anomalous basic events*, which are defined as the segments of a routing process that do not follow the routing specification. These events are then classified in two categories: those that can be detected directly, and those that require statistical analysis. They also give a correspondence between some classical attacker models and the related anomalous basic events. Finally, they propose an IDS which is built to detect all of the anomalous basic events they identified.

Our work is different from theirs in two ways. First of all, instead of building an IDS specifically for AODV, we focus on the evaluation of any IDS for a wireless ad-hoc network. To achieve this, we built a model which is not protocol dependent and adjustable depending on the assumptions. The second important difference is our concept of anomaly. It is based on their anomalous events, but instead of isolating them, we add a model of their dependencies. This allows us to describe attacks taking into account the whole process, instead of just taking the end results of the attack into account.

Attack graphs for vulnerability analysis

In [PS98], the authors describe a system for risk analysis that uses *attack graphs* in order to find the most likely attacks to reach a given goal. These results show the weak points to prioritize when hardening the system. Several other uses of attack graphs can be found in the literature, and a review listing several applications in network vulnerability analysis is available in [LI05]. To give an example of an application on IDS, consider [WLJ06], where the authors describes a method to correlate IDS alerts and transform them into complete attack traces, using such attacker graphs. To our knowledge, attacker graphs have not yet been used to evaluate the coverage of an IDS against the various attacker behaviors encoded in the graph.

The attack graph described in [PS98] is close to our system of anomalies and rules (which describe a structure close to a directed graph), although different in some key points. In their graph, each vertex is a possible attack state describing a combination of machines and access levels for the attacker, and effects of the attack. Then, a library of known attacks is used to build transitions between those states. The analysis of the resulting graph provides paths from the vertices that are considered true to attacker goals. This analysis also considers the probability of success of each attack.

The first major difference with our approach is that we focus on the whole network at once, without distinguishing between individual network nodes and compromise levels. This allows us to evaluate the relevance of IDS inputs from a higher-level standpoint. Also, we do not consider the probability of success of attacks since our objective is to find *undetectable* attacks using IDS inputs only, without looking into the detection mechanisms.

4.1.4 Outline

In section 4.2, we provide our classification of IDS inputs and illustrate it through several examples from the literature. In section 4.3, we formally define our model, which we apply on two existing IDS using our prototype in section 4.4.

4.2 Inputs of IDS for WANETs

We propose our new classification. We then provide a review of the inputs used by several IDS found in the literature.

4.2.1 Classification

IDS build their decision process over a multitude of *inputs* that we classify along two axis. The first one is made of three categories, that express the level of cooperation needed to use an input:

a. **Local** inputs, that are accessible by a node, without help from their neighbors.

b. Inputs requiring **k-neighborhood-wide cooperation** (*i.e.* cooperation within a bounded distance).

c. Inputs requiring **global cooperation** (*i.e.* cooperation between arbitrarily distant nodes).

This distinction allows us to see clearly what infrastructure is needed for the IDS. The first category does not require anything per se, the second one only requires passing messages to neighboring nodes (so broadcasting may be enough), whereas the third one requires a full routing protocol to operate.

The other axis, denoted the *data source* axis, corresponds to how the IDS collect their inputs. We identified five categories. The first category is independent on the network, then the next categories depend on the protocols.

1. **Offline** inputs can be created even if the node is not part of a network. They are neither application-related nor network-related.

2. **Topological** inputs, related to the positions of the nodes and their ability to communicate.

3. **Radio** inputs, linked to the medium access control protocol.

4. **Routing** inputs, related directly or indirectly to the routing protocol.

5. Inputs extracted from the **application data**, as opposed to all the previous categories which analyze how the nodes and the network behave.

Using the data source axis, we now describe and illustrate with references each of those inputs according the level of involvement or cooperation needed to use an input, followed by a summary of this classification.

4.2.2 Offline inputs

Offline inputs do not depend on the network: the object of their monitoring is internal to a node.

a) The first family of offline inputs we identified are **local**, and are usually called host-based IDS. This family of IDS are looking for a partial compromise of the node running the detection, for instance through viruses or vulnerable applications (with for instance [WS02]). This family of inputs contains a large number of IDS inputs, which are by definition not network-centered.

b) To allow compromise detection by third-party nodes, we need **neighborhood cooperation**, as these IDS usually require the cooperation of the suspect node. For instance, in [YWZC07], the authors proposed a scheme where free memory in nodes is pre-loaded with random noise, the knowledge of which is shared among the node's neighbors. They make the hypothesis that a compromised node would have deleted some of that noise to include its rogue algorithms. Then, to detect whether a suspected node has been compromised, each of the neighbors challenges that node. To prove its integrity, the suspect node needs to produce the requested random noise.

c) Finally, we did not find any **global** offline inputs. We conjecture that network-wide cooperation would not add anything significantly more useful than local or neighborhood-scale analysis in this category, as these inputs are not related to the network by definition.

4.2.3 Inputs Based on the Network Topology

This category contains mechanisms that use distance measurements and neighborhoods.

a) First, this analysis can be **local** to a node. For instance, in the IDS described in [dSMR$^+$05], nodes have a list of neighbors, so that messages coming from other sources trigger alerts. This prevents attackers with very strong emission capabilities from broadcasting a message to the whole network (for instance, this is a stepping stone towards *hello floods*). Signal strength monitoring is a very common input, which is for instance found in [DS06, OM05a, OM05b]. To operate, it makes nodes remember the strength of the received signal for the last transmissions from each neighbor. As the signal strength is related to distance, this allows to detect node movement, changes in environmental conditions, and impersonations. Some variants can be found: for instance, [PJdPFWL04], instead of using previous measurements of signal strength to detect anomalies, the expected value is deduced from distance data. However, this measure is not trustworthy, as received signal strength also depends on the emission equipment and environment, and so an intruder that adjusts its emission power to match the expected received signal strength may be able to fool this IDS input.

b) The IDS described in [SXZC07] requires **neighborhood-wide cooperation** to measure distances between nodes in a static network, which allows to detect unexpected changes in localization that are characteristic of a node being manipulated. A similar neighborhood-wide effort can identify Sybil intruders (single nodes that use multiple identities) through signal strength, as described in [DS06]. By computing the ratios of the signal strength of different nodes, the authors show that it is feasible to triangulate the position of other nodes, allowing detection of nodes that share the same exact position and who may be Sybil intruders. Note that this measure may be fooled by an intruder which is able to actively change its emission pattern, as the authors mentioned in the paper.

c) Finally, such an analysis can be done from a **global** point of view. In [DLL$^+$11], the authors argue that under some hypothesis on the underlying network, it is feasible to detect wormholes (two distant intruder nodes, linked with a special communication channel) using only topological data. In [CDPMM07], the authors propose a mechanism to detect nodes sharing identities in the network. It is a global algorithm which reports a suspect node's identity and location to a specific third-party node, using the suspect identity. Then, if that third-party node receives several reports containing different locations for a single identity, an alert is raised.

4.2.4 Inputs from the Medium Access Control (MAC) Protocol

This category contains all the inputs which use data originating from the wireless transmission medium.

a) This category is well-suited to **local** analysis, but increasing the cooperation will not significantly improve these techniques. By looking at indicators in the MAC such as collision rate or the number of retransmissions requests, a node may be able to tell if there is a perturbation in the radio medium, whether they are environmental or caused by an attacker. In all of [BG06, dSMR$^+$05, PPJ06, TBMS05], there is a part of the intrusion detection being done based on the collision rate. This mechanism is close to the definition of specification-based IDS, where monitors observe whether neighbors behave as expected given a protocol. Another approach based on the radio layer is to use characteristic features of the radio emitters to identify them. This technique is called *fingerprinting*, and is found in [BRC07, HBK04]. Such a technique, depending on the precision of the analysis, allows detection of Sybil attackers (as all identities will have the same fingerprint) and impersonations (as the impersonated node will have several fingerprints).

4.2.5 Data Based on the Routing Layer and Traffic

This category is based on the analysis of how messages are routed through the network. Some of the IDS mechanisms we present here are tied to a specific routing protocol, which allows them to monitor the protocol compliance of suspect nodes at the cost of genericity (these are called specification-based IDS). On the other hand, some of them are built upon generic properties that are common to most routing protocols, such as timely packet retransmission or each node using only one public identity. This genericity may reduce the precision of the analysis if the observed protocol has a more complex specification than what the IDS is monitoring.

a) A typical **local** input is nodes monitoring the variations in the volume of their incoming traffic. If they notice a significant difference when compared to some reference measure, the nodes will raise an alert. This mechanism is seen frequently in the literature, with several variants such as separating traffic streams per message type [CANT10], per neighbor [dSMR$^+$05], or depending on the messages source and destination [LLM05]. Some IDS also monitor the intervals between reception of messages of different types [CANT10]. Finally, instead of looking at the traffic flow, the authors of [BG06] suggest observing the application data types which are expected on each route, assuming the network is used to route more than a single type. Another category of local inputs require using promiscuous listening, so that a monitor node can observe its neighbor's behavior. By designing an IDS more specific to the routing protocol, it becomes possible to detect deviations from the routing protocol, using an analysis that can span from simple rules or features to a complete specification-based monitoring. Both [dSMR$^+$05] and [TBMS05] propose some IDS which uses promiscuous listening. The IDS described in [TSB$^+$06, TBK$^+$03] monitor the routing process of neighboring nodes using finite state machines, respectively built for AODV and OLSR (which we presented in section 2.1.3). Another specification-based IDS built on top of an extended finite state machine for AODV is described in [HL04].

b) By extending to a **k-neighborhood-wide** effort, specification-based IDS can be more efficient and accurate. For instance, in [TWKL06], the authors present DEMEM, a specification-based IDS with new messages overlayed on the routing protocol. These messages allow nodes to verify the claims of their neighbors, to detect more attacks on the routing protocol.

c) Finally, the **global** scale inputs can be illustrated with Lipad [AT04]. This IDS does traffic analysis in a centralized way, which allows to locate precisely where the anomaly occurred, and so under certain hypothesis on the number of intruder nodes.

4.2.6 Inputs Based on the Application Data

The last category concerns all the inputs which use the application data, and assumptions about it.

a) A good illustration of **local** application data analysis can be found in [KTK02]. This IDS is designed for general-purpose networks, and operates by statistical analysis of the application data being transmitted in the captured packets, with a different model for each application. The IDS described in [LPR10] also fits this category. This data source is generated from the various applicative requests the nodes make, which are then categorized. With some training data or pre-the detection system rests its case based on the number of requests of each category.

b) Using **neighborhood** collaboration, the authors of [DA03] present an IDS based on the analysis with hidden Markov models of the data delivered to the application. Such a data analysis allows them to detect altered data, depending on what is being monitored. This example is on a global scale, and is the only one we found to illustrate such an analysis for WANET in the literature. We conjecture that this is because such techniques are strongly tied to the application, as this technique could also be used with a lower cooperation level. For instance, in a network monitoring earthquakes, neighborhood-scale cooperation makes sense to detect malicious data insertion. On the other hand, if the measurements are completely unrelated between each other, cooperative analysis of non-overlapping data may not be useful.

4.2.7 Summary

Our classification allows us to categorize the various inputs a network-based IDS uses in its decision process. In Figure 4.1, we recapitulate our classification, and we provide a list of the different inputs we identified from existing IDS, plus the categories in which they belong. Note that the same IDS can use different inputs, and will therefore be in several different categories, but all inputs fit in one cell each. We also include a few mechanisms built to detect specific intruders such as [CDPMM07, DS06], as they are compatible with our definition of IDS input.

We denote by a 'X' the categories which do not bring any new significant information when compared to the lower cooperation levels. Since the MAC protocol is by essence local to a link, nodes have little interest in relying on the declarations of distant nodes to detect intruders. A similar situation appears for global offline inputs. We also denote with a '?' the categories of data analysis where we did not find any example in the literature, but that combination can be relevant to detect malicious nodes.

4.3 Vulnerabilities Discovery Model

4.3.1 Definitions and assumptions

We define an *identity* as the different items a node needs to join and operate legitimately in the network, and a node in possession of a valid identity is considered *associated*. In this context, we denote as *valid* the messages whose data would be delivered to the application if they reach their destination.

The model described in this section is built to be generic, and would fit most networks. However, we specifically chose the rules to model a wireless network, and therefore some of the rules will not be appropriate for wired networks. We consider that some nodes generate data messages, which are then routed to one or several destinations. We provide a way to specify whether attackers have access to identities or not. Our model

Data source	Offline	Topology	MAC	Routing	Data
Local	[WS02]	[dSMR$^+$05] [DS06] [OM05a] [OM05b] [PJdPFWL04]	[BG06] [BRC07] [dSMR$^+$05] [HBK04] [PPJ06] [TBMS05]	[BG06] [CANT10] [dSMR$^+$05] [HL04] [LLM05] [NKJ$^+$09] [TBMS05] [TSB$^+$06] [TBK$^+$03]	[KTK02] [LPR10]
Neighborhood	[YWZC07]	[DS06] [SXZC07]	X	[TWKL06]	[DA03]
Global	X	[CDPMM07] [DLL$^+$11]	X	[AT04]	?

Figure 4.1: Classification of IDS data sources

does not take into account movement of nodes, and assumes that the neighbors of a node know its declared identity.

We assume that attackers emit data with a constant emission pattern and constant transmission power. This assumption can be relaxed when explicitly stated.

4.3.2 Anomalies and facts

We now describe formally our model, beginning with the notion of facts.

Facts

We define *facts* as various assumptions that are selected in order to refine the model.

Definition 4.1 (Facts)**.** *Assumptions about the network, protocols and attackers are called facts. They are represented by keywords in italic. We denote the set of all facts* **F**.

We now detail all facts contained in **F**, ordered by category. The first category contains facts related to the attacker, its capabilities and initial knowledge.

- *CompromisableNodes*: An attacker is able to take full control of legitimate nodes, and recover their identity and knowledge. This makes them insider intruder nodes.
- *TxPowAdjust*: Intruder nodes are able to adjust their radio transmission power. This allows them to knowingly influence the corresponding received signal strength.
- *TxDirAdjust*: Intruder nodes are able to knowingly change their emission pattern (for instance, using a steerable antenna). In our model, this means that these intruder nodes are able to choose which subset of their neighbors receive their transmissions.
- *CanImpersonate*: The attacker is able to impersonate honest nodes such that transmissions appear to come from them. However, this fact does not give the attacker the ability to create valid messages.

The second category contains the facts related to the protocols in use in the network.

- *SimpleValidity*: The protocol does not guarantee the unforgeability of messages, and therefore an attacker can alter or create a message and keep it valid.
- *ValidityCheckedEachHop*: The protocol guarantees the unforgeability and the freshness of messages, and therefore, an attacker cannot alter one of them in a way that keeps it valid, and appearing to originate from an uncompromised node. The validity of messages is checked at each hop.
- *NoConfidentiality*: The protocol does not provide any confidentiality, and therefore any passive listener is able to recover the contents of messages in transit.

- *HopByHopConfidentiality*: Only the nodes on a message's route (or anyone with their identity) can recover its contents. This is hop-by-hop confidentiality.
- *EndToEndConfidentiality*: Only the source and destination(s) of a message (or anyone with their identity) can recover its contents.
- *OpenNetwork*: The network is by design open, requiring little to no knowledge to join. Therefore, nodes have access to new valid identities at will, and any node can associate, regardless of initial knowledge or pre-existing trust relationships.

Factual relationships Some of these facts are related. We define that a fact F_1 is more restrictive for an attacker than F_2 when any possible attack when F_1 holds is also possible when F_2 holds.

Definition 4.2 (Factual relationships $\mathcal{F}$)**.** *If a fact $F_1 \in \mathbf{F}$ is more restrictive for an attacker than a fact $F_2 \in \mathbf{F}$, we express that relation using the following rule named (FR):*

$$(FR)\frac{F_2}{F_1}$$

We denote by $\mathcal{F}$ the set of rules (F-Conf1),(F-Conf2) and (F-VC1).

We now describe the contents of $\mathcal{F}$. Having validity checks at each hop blocks any invalid message before it gets retransmitted by an honest node, which only limits what messages an attacker can usefully send. Therefore, the fact *ValidityChecksAtEachHop* is more restrictive for an attacker than *SimpleValidity.*

$$(F-VC1)\frac{SimpleValidity}{ValidityChecksAtEachHop}$$

Regarding the confidentiality-related facts, the relationship is similar. *HopByHopConfidentiality* strictly restricts attacker knowledge when compared to *NoConfidentiality*, as it only prevents listeners outside of the route from being able to read the data. Therefore, *HopByHopConfidentiality* is more restrictive for an attacker than *NoConfidentiality.*

$$(F-Conf1)\frac{NoConfidentiality}{HopByHopConfidentiality}$$

If an attack is possible when considering the *EndToEndConfidentiality* fact, then removing the confidentiality aspect for intermediate nodes does not change that possibility, as it merely adds more possibilities for the intruder. Therefore, we say that *EndToEndConfidentiality* is more restrictive for an attacker than *HopByHopConfidentiality*, and the rule, named (F-Conf2), is written as follows:

$$(F-Conf2)\frac{HopByHopConfidentiality}{EndToEndConfidentiality}$$

The set of facts we want to use for the analysis is denoted $\mathbf{F}_I \subseteq \mathcal{F}$. Using this set of facts, we build a set of axioms, named the selected hypothesis.

Definition 4.3 (Selected hypothesis $Hyp(\mathcal{H}, \mathbf{F}_I)$)**.** *Let $\mathcal{H}$ be the set of all possible axioms deducing a fact from $\mathbf{F}$, and let $\mathbf{F}_I$ be a subset of $\mathbf{F}$ we want to assume for the verification. The set of selected hypothesis is a set of axioms, denoted by $Hyp(\mathcal{H}, \mathbf{F}_I)$, and defined by:*

$$Hyp(\mathcal{H}, \mathbf{F}_I) = \left\{ (AF)\frac{}{F} \middle| F \in \mathbf{F}_I \right\}$$

Anomalies

We define anomalies as objects that are used to describe the different steps in an attack, such as the attacker getting recognized as legitimate or message suppression. Anomalies are linked together using rules. We first define them, and then we present the contents of $\mathbf{A}$, separated in categories.

Definition 4.4 (Anomalies)**.** *Anomalies are the results of the attacker's behavior. We denote the set of all anomalies $\mathbf{A}$.*

The first subset of anomalies we present are related to impersonation.

- *OmniImpersonation*: An intruder node transmits packets as if they were emitted by an honest neighbor, but the message can be received by any neighbor, and the received signal strength may differ from the one from legitimate transmissions.
- *DirImpersonation*: An intruder node transmits packets in a directed fashion, such that only the attacker and the receiver know the transmission happened. The signal strength of this transmission may however be different from what is expected from the impersonated node.
- *TxPowImpersonation*: An intruder node impersonates an honest node, while adjusting its transmission power so that the signal strength at the receiver corresponds to the signal strength which would be expected from the impersonated node.
- *DirTxPowImpersonation*: An intruder node transmits packets in a directed fashion, such that only the attacker and the receiver know the transmission happened. Furthermore, the intruder adjusted its transmission power so that the signal strength at the receiver corresponds to the signal strength which would be expected from the impersonated node.
- *Impersonation*: The attacker can communicate with any node, as if the communication happened from an honest neighbor. This anomaly is a generic version of the previous ones.

The next category are anomalies related to node compromise and identities.

- *VirusCompromise*: The attacker compromises some part of a node, whose identity is now compromised. The attacker is also able to control this node.
- *TotalCompromise*: The attacker takes full control of a node, whose identity is now compromised.
- *AttackerAssociated*: The attacker uses intruder nodes which are associated. This fact should be selected when modeling insider attacks.
- *RoutingMisbehavior*: Intruder nodes deviate from the routing protocol.

Finally, the last category of anomalies are related to messages, what happens to them, and to the application data.

- *ApplicationDataAltered*: The attacker alters the data which is delivered to the application, either by adding, altering or subtracting data.
- *Snooping*: The attacker reads some of the application data going through the network.
- *Alteration*: The attacker alters data in messages.
- *ValidAlteration*: The attacker alters data in messages, while keeping them both valid and appearing to be from an emitter whose identity is uncompromised.
- *ImmediateAlteration*: An intruder node is able to alter the data in a message it received.
- *NeighborVisibleAlteration*: An intruder node alters the data in a message it received, in a way that can be overheard by neighbors.
- *NeighborVisibleValidAlteration*: An intruder node alters the data in a message it received, while keeping them both valid and appearing to be from an emitter whose identity is uncompromised, in a way that can be overheard by neighbors.
- *ImmediateValidAlteration*: An intruder node is able to alter the data in a message it received, while keeping it both valid and appearing to be from an emitter whose identity is uncompromised.
- *NeighborVisibleSuppression*: The attacker drops data messages, which prevents them from being delivered to their destinations. This action can be overheard by the node's neighbors.
- *Suppression*: The attacker drops data messages at some point in the network, which prevents them from being delivered to their destinations.
- *ImmediateSuppression*: An intruder node is able to drop valid data-bearing messages.

- *Insertion*: The attacker creates valid data messages.
- *NeighVisibleInsertion*: An intruder node creates valid data messages, which may be overheard by neighbors.
- *ImmediateInsertion*: An intruder node has the ability to create valid data messages.

Notice that we separated the three last anomalies. This is needed to model intruders able to transfer data to a single node, instead of their whole neighborhood. Such an attacker would be able to bypass promiscuous monitoring by its neighbors.

4.3.3 Rules and attacker progression

Anomalies follow a logical progression, with certain anomalies and facts being prerequisites to an attacker reaching some other anomalies. To model these dependencies, we use rules.

Definition 4.5 (Rules). *For $i \in [0..n]$, let $T_i \in \mathbf{A} \cup \mathbf{F}$, and $A \in \mathbf{A}$. We denote the* rule *(R) which allows an attacker to reach A given that all T_i hold by*

$$(R)\frac{T_0 \quad \dots \quad T_n}{A}$$

We denote by $\mathcal{R}$ the set of all our rules.

The set $\mathcal{R}$ contains 39 rules, given in Figure 4.2, and described below. We now explain how those anomalies and rules are used to search for undetectable attacks.

We now present all the rules we consider. These are abitrarily ordered by theme: first, the rules related to node compromises and attacker association, then impersonations, application data alterations, and finally confidentiality.

Rules related to compromises

$$(CompV)\frac{CompromisableNodes}{VirusCompromise}, (CompT)\frac{CompromisableNodes}{TotalCompromise}$$

The *CompromisableNode* fact determines whether nodes are considered tamper-proof for an attacker. If they are not, an attacker can compromise them in two different ways: either through physically taking control of the node, or by execution of malicious code.

$$(VAssoc)\frac{VirusCompromise}{AttackerAssociated}, (TAssoc)\frac{TotalCompromise}{AttackerAssociated}$$

An attacker which manages to execute malicious code on an honest node, or reprogram an honest node effectively controls part or totality of that node, and we therefore consider it an intruder node which is associated.

$$(Open)\frac{OpenNetwork}{AttackerAssociated}$$

Name	**Prerequisite(s)**	**Conclusion**
CompV	*CompromisableNodes*	*VirusCompromise*
CompT	*CompromisableNodes*	*TotalCompromise*
VAssoc	*VirusCompromise*	*AttackerAssociated*
TAssoc	*TotalCompromise*	*AttackerAssociated*
Open	*OpenNetwork*	*AttackerAssociated*
Misbehave	*AttackerAssociated*	*RoutingMisbehavior*
OmnI	*CanImpersonate*	*OmniImpersonation*
DirI	*TxDirAdjust* $\wedge$ *CanImpersonate*	*DirImpersonation*
PowI	*TxPowAdjust* $\wedge$ *CanImpersonate*	*TxPowImpersonation*
DirPowI	*TxPowAdjust* $\wedge$ *TxDirAdjust* $\wedge$ *CanImpersonate*	*DirTxPowImpersonation*
OtoI	*OmniImpersonation*	*Impersonation*
DtoI	*DirImpersonation*	*Impersonation*
TtoI	*TxPowImpersonation*	*Impersonation*
DTtoI	*DirTxPowImpersonation*	*Impersonation*
IStoS	*ImmediateSuppression*	*NeighVisibleSuppression*
DirIStoS	*ImmediateSuppression* $\wedge$ *TxDirAdjust*	*Suppression*
IIntoNVIn	*ImmediateInsertion*	*NeighVisibleInsertion*
AssoIns	*AttackerAssociated*	*ImmediateInsertion*
ImpInser	*Impersonation* $\wedge$ *SimpleValidity*	*ImmediateInsertion*
IAlt	*AttackerAssociated* $\wedge$ *RoutingMisbehavior*	*ImmediateAlteration*
VIAlt	*ImmediateAlteration* $\wedge$ *SimpleValidity*	*ImmediateValidAlteration*
VIaltIAlt	*ImmediateValidAlteration*	*ImmediateAlteration*
ValtAlt	*ValidAlteration*	*Alteration*
NVIaltAlt	*ImmediateAlteration*	*NeighVisibleAlteration*
NVIValtVAlt	*ImmediateValidAlteration*	*NeighVisibleValidAlteration*
NVIaltAlt	*ImmediateAlteration* $\wedge$ *TxDirAdjust*	*Alteration*
NVIValtVAlt	*ImmediateValidAlteration* $\wedge$ *TxDirAdjust*	*ValidAlteration*
AssoISup	*AttackerAssociated* $\wedge$ *RoutingMisbehavior*	*ImmediateSuppression*
InSupAlt	*ImmediateSuppression* $\wedge$ *ImmediateInsertion*	*ImmediateAlteration*
S	*NeighVisibleSuppression*	*Suppression*
DirIIntoIn	*ImmediateInsertion* $\wedge$ *TxDirAdjust*	*Insertion*
NVIntoIn	*NeighVisibleInsertion*	*Insertion*
IaltAlt	*NeighVisibleAlteration*	*Alteration*
IValtVAlt	*NeighVisibleValidAlteration*	*ValidAlteration*
SApp	*Suppression*	*ApplicationDataAltered*
AApp	*ValidAlteration*	*ApplicationDataAltered*
IApp	*Insertion*	*ApplicationDataAltered*
HopSnoop	*AttackerAssociated* $\wedge$ *HopByHopConfidentiality*	*Snooping*
ConfSnoop	*NoConfidentiality*	*Snooping*

Figure 4.2: List of all the rules

The definition of an open network specifies that any node can associate. Therefore, in this context, intruder nodes can associate freely.

$$(Misbehave)\frac{AttackerAssociated}{RoutingMisbehavior}$$

An intruder node who is associated can choose to deviate from the protocols used in the network.

Rules related to impersonations

$$(OmnI)\frac{CanImpersonate}{OmniImpersonation}$$

Simple impersonation of a node does not require any specific equipment besides what is expected of the intruder. We therefore suppose that if impersonations are possible for an intruder, this rough form of it is possible.

$$(DirI)\frac{TxDirAdjust \qquad CanImpersonate}{DirImpersonation}$$

An intruder with access to directionnal antennas which allow to selectively choose the recipients of emissions, plus the required knowledge for impersonations is able to impersonate a node without monitors being able to overhear the communication.

$$(PowI)\frac{TxPowAdjust \qquad CanImpersonate}{TxPowImpersonation}$$

An intruder able to change its emission power, and able to impersonate nodes, is able to impersonate an honest node while matching that target node expected received signal strength.

$$(DirPowI)\frac{TxPowAdjust \qquad TxDirAdjust \qquad CanImpersonate}{DirTxPowImpersonation}$$

Having both the ability to change its emission pattern and its transmission power allow an intruder node to impersonate an honest node with the expected received signal strength for the destination node, and without any other monitor node overhearing the transmission.

$$(OtoI)\frac{OmniImpersonation}{Impersonation}, (DtoI)\frac{DirImpersonation}{Impersonation}$$

$$(TtoI)\frac{TxPowImpersonation}{Impersonation}, (DTtoI)\frac{DirTxPowImpersonation}{Impersonation}$$

We regroup the four different forms of impersonation as a generic act of impersonation, as the specific details of the impersonation do not matter anymore past this point.

Rules related to application data alteration

$$(IStoS)\frac{ImmediateSuppression}{NeighVisibleSuppression}$$

An intruder node able to drop messages, or able to make another node drop messages is able to cause data loss in a way that is obvious to its neighboring monitors.

$$(DirIStoS)\frac{ImmediateSuppression \qquad TxDirAdjust}{Suppression}$$

An intruder node able to control who receives its messages can forward a message to its neighboring monitors only, and not its destination. Thus, the message gets dropped, and monitors do not overhear any breach of protocol.

$$(IIntoNVIn)\frac{ImmediateInsertion}{NeighVisibleInsertion}$$

An intruder node which simply inserts a message can be overheard by neighboring monitors.

$$(AssoIns)\frac{AttackerAssociated}{ImmediateInsertion}$$

An attacker controlling an associated intruder node is able to send new data from this node.

$$(ImpInser)\frac{Impersonation \qquad TrivialValidity}{ImmediateInsertion}$$

An intruder node able to do impersonations can insert new data, but only if it is able to make it valid. Remark that this rule does not require the attacker to be associated.

$$(IAlt)\frac{AttackerAssociated \qquad RoutingMisbehavior}{ImmediateAlteration}$$

An attacker which uses associated intruder nodes can use them to change the application data that is being forwarded by that intruder node, but this is a deviation from the routing protocol.

$$(VIAlt)\frac{ImmediateAlteration \qquad TrivialValidity}{ImmediateValidAlteration}$$

An intruder node able to alter a message when their validity is easy to obtain ends up with an altered valid message.

$$(VIaltIAlt)\frac{ImmediateValidAlteration}{ImmediateAlteration}, (ValtAlt)\frac{ValidAlteration}{Alteration}$$

By definition, a valid alteration is an alteration.

$$(NVIaltAlt)\frac{ImmediateAlteration}{NeighVisibleAlteration},$$

$$(NVIValtVAlt)\frac{ImmediateValidAlteration}{NeighVisibleValidAlteration}$$

An associated node able to alter messages (valid or not) can do it in a way that will be visible for any neighbor monitoring that action.

$$(NVIaltAlt)\frac{ImmediateAlteration \qquad TxDirAdjust}{Alteration},$$

$$(NVIValtVAlt)\frac{ImmediateValidAlteration \qquad TxDirAdjust}{ValidAlteration}$$

By using its directionnal emission capabilities, an intruder node can send the unaltered version of a message to the monitor nodes and the alterated version to the next hop, resulting in a stealthy alteration.

$$(AssoISup)\frac{AttackerAssociated \qquad RoutingMisbehavior}{ImmediateSuppression}$$

An intruder node which is associated can drop data messages by deviating from the routing protocol.

$$(InSupAlt)\frac{ImmediateSuppression \qquad ImmediateInsertion}{ImmediateAlteration}$$

An intruder node which is both able to delete messages and add messages can do both at the same time, which would appear as the retransmission of an altered message.

$$(S)\frac{NeighVisibleSuppression}{Suppression}$$

A suppression visible by neighbors is still a supression.

$$(DirIIntoIn)\frac{ImmediateInsertion \qquad TxDirAdjust}{Insertion}$$

An intruder node with directionnal emission capabilities can send a new message directly to its destination, and not to any third-party node, making that action invisible to its neighbors.

$$(NVIntoIn)\frac{NeighVisibleInsertion}{Insertion}$$

An insertion visible by neighbors is still an insertion.

$$(IaltAlt)\frac{NeighVisibleAlteration}{Alteration},$$

$$(IValtVAlt)\frac{NeighVisibleValidAlteration}{ValidAlteration}$$

An alteration visible by neighbors is still an alteration.

$$(SApp)\frac{Suppression}{ApplicationDataAltered},(AApp)\frac{ValidAlteration}{ApplicationDataAltered},$$

$$(IApp)\frac{Insertion}{ApplicationDataAltered}$$

Suppressions, insertions, and valid alterations end up modifying the data sent to the application.

Rules related to confidentiality

$$(HopSnoop)\frac{AttackerAssociated \qquad HopConfidentiality}{Snooping}$$

If the routing protocol guarantees confidentiality of the application data only when it is transmitted, having an associated intruder node allows the attacker to recover some application data. If the attacker was not able to snoop, this would mean that either the hops cannot read the data, or the attacker does not control an intermediate node to begin with.

$$(ConfSnoop)\frac{NoConfidentiality}{Snooping}$$

If the data messages contain readable application data, an attacker can simply overhear the message to recover it.

4.3.4 IDS Inputs and attack detection

An IDS has several inputs, each of these noticing a certain set of anomalies. To know if an IDS is adequate to prevent an attacker from reaching a given goal, we need to check if there is a way to reach a target anomaly from accepted facts, in a way that do not use any of the anomalies covered by the set of this IDS's inputs. To model this, given an IDS, we remove all the rules going to anomalies detected by that IDS, leaving only anomalies which do not trigger detection.

Definition 4.6 (IDS)**.** *An IDS is modeled by the set of anomalies it monitors $\mathbf{A}_I \subseteq \mathbf{A}$. We define the set of rules $IDS(\mathcal{R}, \mathbf{A}_I)$ which is the allowed set of rules for the attacker given the base rules. This set is defined as:*

$$IDS(\mathcal{R}, \mathbf{A}_I) = \left\{ (R)\frac{T_0 \quad \dots \quad T_n}{A} \in \mathcal{R} \;\middle|\; A \notin \mathbf{A}_I \land \forall i, T_i \notin \mathbf{A}_I \right\}$$

To build $\mathbf{A}_I$, one should examine which are the inputs the IDS uses, and for each of them, which are the anomalies that may be detected by such an input. For instance, message addition or subtraction can be detected by traffic analysis. Then, the set $IDS(\mathcal{R}, \mathbf{A}_I)$ is used to build the set of rules which will be used by the analysis.

Definition 4.7 (Setting)**.** *The set of rules obtained by the union of selected hypothesis, the rules allowed given a specific IDS, and factual relationships is called a setting. We denote it by : $\mathcal{S} = IDS(\mathcal{R}, \mathbf{A}_I) \cup Hyp(\mathcal{H}, \mathbf{F}_I) \cup \mathcal{F}$*

Once the setting is determined, we can search for undetectable attacks, by looking at which anomalies are reachable using the rules.

Definition 4.8 (Undetectable attack)**.** *Let $\mathcal{S}$ be a setting describing an IDS together with assumptions about the network, protocol and attacker. Let $G \in \mathcal{A}$ be an anomaly. We say that there exists an attack resulting in G which cannot be detected by the IDS described in $\mathcal{S}$ if G is reachable using $\mathcal{S}$.*

To summarize, in our model, an attack is a chain of anomalies, linked together by rules. Starting from facts, the attacker mounts his attack using only the rules in the setting (*i.e.* the rules that allow him to stay undetected). Each of those rules allow him to progress to further anomalies. Therefore, analyzing whether the attacker can reach a certain anomaly in a specific setting allows us to know whether an undetectable attack is possible against that IDS. Also, we focus only on attacks which change the data to the application, impersonations, and node compromise. All considerations linked to the performances and availability of the network are not captured by our model, and constitute a natural future extension of this work.

4.4 Analysis of existing IDS

We now use the inputs and the intruder model previously described to evaluate two existing IDS, [OM05b] and [dSMR+05], and show the weaknesses and the possible improvements we discovered.

4.4.1 A Real-Time Node-Based Traffic Anomaly Detection Algorithm [OM05b]

In [OM05b], Ilker Onat and Ali Miri present an anomaly-based IDS based on two inputs, received signal strength, and packet arrival rates. They make several hypothesis: the routing protocol is based on a tree (such as GBR), nodes are static and can uniquely identify neighbors, all nodes use the same hardware and software, and all nodes use constant transmission power. Our model does not take into account movement of nodes, and assumes that neighbors can be identified. Thus, the only assumption we need to transpose is the constant transmission power. This is modeled by not adding *TxPowAdjust* to the hypothesis set.

We now build the set $\mathbf{A}_I$ of anomalies the IDS can detect. The first input used by this IDS is a packet arrival rate analysis. This input is able to discover any attacker that stops or inserts more messages than expected from an honest node. The corresponding anomalies in our model are *Suppression* and *Insertion*. Each node running this IDS also observe the received signal strength, which will detect the anomalies *OmniImpersonation* and *DirImpersonation*. We therefore have our set of anomalies $\mathbf{A}_I = \{$*Suppression, Insertion, OmniImpersonation, DirImpersonation*$\}$.

The next step is to set the attacker's goal. In their paper, the authors address node impersonation, and resource depletion by excessive generation of traffic. As the latter is not covered by our model, we focus on impersonation first, and then consider a more general anomaly, *ApplicationDataAltered*.

In order to find if an attacker in our model is able to impersonate honest nodes, we need to choose the facts modeling the attacker. The IDS supposes that attackers have the ability to impersonate honest nodes (fact *CanImpersonate*), and we also suppose that they have adjustable antenna patterns (fact *TxDirAdjust*) as no assumptions were made about this in the paper. We therefore set $\mathbf{F}_I = \{$*CanImpersonate, TxDirAdjust*$\}$.

From the facts and the IDS description, we compute the set of rules $\mathcal{S}$, and search for a way to reach *Impersonation* using $\mathcal{S}$. We see that in this case, the tool cannot apply any rules, and so our system did not find weaknesses on this aspect of the IDS.

However, the assumption that the attacker cannot modify its intruder nodes's transmission power is strong, as such hardware is readily available. If we relax that assumption by removing *TxPowAdjust* from $\mathbf{F}_I$, we find that the attacker can now reach *Impersonation* by doing an impersonation with adjusted transmission power, effectively bypassing the IDS input. To prevent this, the IDS would need a way of detecting this behavior.

We can also consider other intruder goals. For instance, we wonder if an attacker would be able to alter the data going to the application (*ApplicationDataAltered*). Let us assume that nodes can be compromised by an attacker (fact *CompromisableNodes*), and that validity is easy to fake for the attacker (fact *SimpleValidity*). As we assumed that nodes can be compromised, and there are no protections against this in the IDS, the intruder can take control of some nodes, and make them alter the data they retransmit. As we assumed that an attacker can fake the validity of a message, the results of that alteration is valid, thus the altered data will get delivered without any alert.

This attack path uses the fact that traffic flow analysis does not protects against message alterations. To prevent this, the IDS would need countermeasures preventing message alterations, such as choosing the right protocols to have integrity and authenticity of the messages, or using more IDS inputs such as the ones used by [dSMR+05]. The other way to prevent this attack would be to prevent the compromise of nodes, either through specific inputs (see for instance [YWZC07] or [SXZC07]).

4.4.2 Decentralized Intrusion Detection [dSMR+05]

In [dSMR+05], the authors propose an IDS for WSNs based on promiscuous listening. The network contains monitoring nodes, which observe their neighbor's behavior. If their neighbors break one of a series of rules, an alert is raised. To avoid confusion with the rules from our model, we call the rules from this IDS *behavior rules.* They are the following: A node must receive messages regularly (the interval rule), neighbors must retransmit packets quickly (the delay rule), without altering them, nor repeating them. Also, transmissions must come from a sensible distance, and there must not be too many collisions.

We first build the set of monitored anomalies $\mathbf{A}_I$. The interval rule detects *Suppression* and *Insertion*, as both addition or suppression of messages alters downstream traffic flows. The integrity rule detects *NeighVisibleAlteration* and *NeighVisibleValidAlteration*, as they are based on promiscuous monitoring. The delay rule detects *NeighVisibleSuppression.* The last three behavior rules are not considered in our model, as we do not model any sort of distance measurements for the range rule, nor availability-related anomalies regarding the collision rule. We therefore have our set of anomalies $\mathbf{A}_I = \{$*Suppression, Insertion, NeighVisibleAlteration, NeighVisibleValidAlteration, NeighVisibleSuppression*$\}$.

There are no specific hypothesis about the nodes in the paper. Regarding facts, we include *CompromisableNodes* to allow the attacker to compromise nodes. We also add *TxDirAdjust* to model the attacker's access to advanced hardware. Regarding the protocols, we add *EndToEndConfidentiality* to model a protocol ensuring that the data stays confidential, and *SimpleValidity* to be able to find an attack. Thus, we have $\mathbf{F}_I = \{$*CompromisableNodes, TxDirAdjust, EndToEndConfidentiality, SimpleValidity*$\}$. For the intruder goal, we select *ApplicationDataAltered* as it encompasses most of what this IDS aims to prevent. Our model shows that there is an undetectable attack.

Similarly to the previous IDS, this attack stems from the assumption that the attacker can compromise honest nodes, as there are no countermeasures regarding these anomalies. With an associated intruder node, the attacker can therefore modify the data being routed, as we assumed intruder nodes can alter data while keeping packets valid. However, the IDS makes honest nodes monitor their neighbors for such a behavior. This is where we use the adjustable antenna patterns: with these, the intruder node is able to send the altered packet to its destination, while sending the initial version of that packet to the monitors. This way, the attacker can alter data, while appearing to satisfy the behavior rules triggering the intrusion detection. Then, that altered packet will be forwarded to its destination and delivered, effectively altering application data, which is the goal we set.

The straightforward way to prevent this specific attack in our model is to use a protocol that guarantees the validity of the transmitted message. Indeed, when removing the *SimpleValidity* fact, our tool was not able to find an undetectable attack. Alternatively, one may try to prevent any association of the intruder, through for instance tamperresistant nodes and secure software. This way, attackers will not be able to alter the traffic going through the network, and this would also prevent an attacker from reaching *ApplicationDataAltered* in our model.

4.5 InDICE : INtrusion Detection Inputs Coverage Evaluation

We built a prototype, called InDICE (for INtrusion Detection Inputs Coverage Evaluation) which automatically goes through all the reachable anomalies, given an IDS and facts. It is available online, along with a manual, in [JL13c]. The examples in the previous section were analyzed using that tool, and the analysis took less than a second for each of them on a regular laptop. We now describe succinctly that tool. w

Tool usage First, the command-line arguments allow to determine the setting used in the following analysis. For each of those arguments:

- The character '+' followed by a fact adds it to $\mathbf{F}_I$.
- The character '-' followed by an anomaly adds it to the IDS-observed set $\mathbf{A}_I$ which is forbidden to the attacker.
- The character '%' followed by an anomaly marks it as the target. In this case, the tool outputs whether the anomaly is reachable for an intruder.

The output of the tool is made of two parts. The first one is the description of the setting, and the second part describes which rules are applied, and what are the resulting anomalies. For instance, when the tool outputs:

```
...  Reaching TxPowImpersonation using rule PowI from (TxPowAdjust and
CanImpersonate)
```

This means that since the anomalies/facts *TxPowAdjust* and *CanImpersonate* hold, the tool used rule (PowI) to conclude the anomaly *TxPowImpersonation.* If no such lines are displayed, this means that no rules could be applied.

Example analysis We now present the tool usage and interpretation we used for the analysis of the first IDS, a Real-time Node-based IDS, which analysis is available in Section 4.4.1.

In this IDS, the monitored anomalies are the following: *Suppression, Insertion, OmniImpersonation* and *DirImpersonation.* The facts we chose for the basic analysis, following the description of [OM05b], are *CanImpersonate* and *TxDirAdjust.*

The first analysis for this IDS follows the hypothesis of the paper. We target *Impersonation*, to evaluate whether their countermeasures are enough to cover all the cases.

We found that given their hypothesis, the IDS is indeed secure in our model, as we could not reach any target anomaly.

```
[rjamet@dinah InDICE]$ ./indice.pl +CanImpersonate +TxDirAdjust -Suppression
-Insertion -OmniImpersonation -DirImpersonation %Impersonation
[?]  Proto :  usage ./indice.pl [%TargetAnomaly] +Fact1 +Fact2 -Anomaly1
-Anomaly2
[+] Fact :  CanImpersonate
[+] Fact :  TxDirAdjust
[+] IDS forbids :  Suppression
[+] IDS forbids :  Insertion
[+] IDS forbids :  OmniImpersonation
[+] IDS forbids :  DirImpersonation
[+] Opening rules...
[+] Read 42 rules and factual relationships, running the analysis
[+] Finished :  did not reach Impersonation, no undetectable attack using our
model.
```

In the second phase, we add the *TxPowAdjust* fact to signify that the attacker can now adjust its transmission power. Note that this goes against one of the hypothesis of the paper. With this setting, we find that an attacker can bypass the transmission power countermeasure, and therefore reach the *Impersonation* anomaly.

```
[rjamet@dinah InDICE]$ ./indice.pl +CanImpersonate +TxDirAdjust +TxPowAdjust
-Suppression -Insertion -OmniImpersonation -DirImpersonation %Impersonation
[?]  Proto :  usage ./indice.pl [%TargetAnomaly] +Fact1 +Fact2 -Anomaly1
-Anomaly2
[+] Fact :  CanImpersonate
[+] Fact :  TxDirAdjust
[+] Fact :  TxPowAdjust
[+] IDS forbids :  Suppression
[+] IDS forbids :  Insertion
[+] IDS forbids :  OmniImpersonation
[+] IDS forbids :  DirImpersonation
[+] Opening rules...
[+] Read 42 rules and factual relationships, running the analysis
...  Reaching TxPowImpersonation using rule PowI from (TxPowAdjust and
CanImpersonate)
...  Reaching DirTxPowImpersonation using rule DirPowI from (TxPowAdjust and
TxDirAdjust and CanImpersonate)
...  Reaching Impersonation using rule TtoI from (TxPowImpersonation)
[+] Finished :  reached Impersonation, there is an undetectable attack using
our model.
```

4.6 Conclusion and future work

We presented a characterization and modeling of the different data sources used in network-based intrusion detection systems, focusing on wireless ad-hoc networks. We found that a lot of IDS from the literature base their decisions on a small set of distinct inputs, which we have listed in the first part of this chapter. We then used those inputs to build a decision aid in order to help IDS designers to locate some of the oversights of their algorithms, depending on the protocols used in their network. Finally, we provided InDICE, a prototype implementation of this model.

In the future, we would like to further refine that model and take into account various families of protocols, especially when looking at the low levels of the protocol stack, such as the medium access protocols. Also, we would like to extend our model to include availability attacks.

It would also be interesting to consider the accuracy of the detection processes, instead of only considering how an attacker can bypass defenses. This way, if we know that some aspect of the IDS does not reliably detect some attacks, and that this unreliability can be estimated, then the model could be used to discover the weak points of the system. This information will also indicate where additional security would benefit the most. The downside of such a system when compared to our actual model is that it requires taking into account the detection mechanisms of IDS, adding a fair amount of analysis in order to obtain results.

Chapter 5

Conclusion

Contents

5.1 Summary

We introduced wireless ad-hoc networks (WANETs) and wireless sensor networks (WSNs) in Chapter 1, along with the field of formal verification of cryptographic protocols.

In Chapter 2, we presented SR3, a Secure, Resilient, Reputation-based Routing algorithm built for resiliency and fairness [ADJL13b, ADJL13a]. This protocol is a convergecast routing protocol designed to use only low-cost primitives and to have a small memory footprint. SR3 is therefore well-suited for WSNs, where all these restrictions apply. SR3 builds its routing from scratch, by randomly routing its initial messages and by learning from their behavior. All nodes keep track of recently acknowledged messages for their neighbors, while obsolete data get naturally replaced. This allows our protocol to be self-adaptive. We formally proved three security properties of this protocol using two automated tools: *CryptoVerif* [Bla08], which uses the computational model, and *Scyther* [Cre08], which uses the symbolic model. SR3's messages are confidential, unforgeable (this property implies their integrity and authenticity), and the acknowledgment for a given message can only be built by an adversary if that message reached the sink. Our reputation system rests upon this assumption. SR3 is built to provide a good delivery rate, even in presence of internal attackers. We evaluated this ability using simulations, using *Sinalgo* [Dis08], a discrete events network simulator. We compared SR3 to several other algorithms from the literature, when facing various attacker scenarios: blackholes, selective forwarding attackers, sybil nodes, and wormholes. In all of these cases, SR3 is able to keep a good delivery rate, especially when the degree of the network is low. Furthermore, SR3 can recover from attacks where the behavior of intruders evolves over time. Our implementation of SR3 in Sinalgo is available online at [ADJL13c], along with the security proofs of the algorithm (which are also available in Appendix A).

Our second contribution is presented in Chapter 3. Routing security is a notion encompassing multiple properties, and we propose a formalization of one of these, which we named *incorruptibility*. A routing protocol is incorruptible if there is no way for an attacker to influence how messages are routed by altering them. To formalize this idea, we propose a measure of distance between how two protocols route messages. This measure is then used in a computational definition of the incorruptibility of a routing protocol, when facing an external attacker. The core idea of this measure is to observe the probability of the message m reaching a node a before a node b, compared to the same probability for an attacker-modified message m'. We provide some examples to illustrate this measure. In the next part, we generalized this notion to bounded corruptibility, where a protocol, when attacked, behaves somewhere between two bounds formalized by routing protocols. We also provide variants of these two notions to model attackers who do not alter or redirect messages, and then we supply example protocols and proofs.

The third contribution deals with intrusion detection systems (IDS) for wireless ad-hoc networks [JL13a]. Their decision procedures are well-studied, so we chose to focus on their sources of data. In a first time, we do a survey of which data sources are used by IDS for WANETs. We propose a classification for these inputs, based on the level of collaboration between nodes they require, and which protocol level do they rest on. In a second time, we propose InDICE (Intrusion Detection Inputs Coverage Evaluation) a model to automatically discover flaws in IDS, based only on their inputs. We use a rule-based model centered on anomalies in order to represent attacker progression towards certain goals. Then, we discover which anomalies in the graph may trigger one of the IDS inputs, and finally, based on certain assumptions (*facts*) depending on the attacker, topology and protocols, the analysis propagates through all anomalies which are guaranteed not to trigger detection. The model then concludes whether the attacker is able to reach its objectives without detection. We implemented this model in a tool, which is freely available online at [JL13c].

5.2 Perspectives

We have several leads on how our study of SR3 could be expanded. First of all, we believe SR3 is well-suited for networks with some mobile nodes, and maybe also for networks where there is global slow mobility. Algorithmically, only a small change is required in SR3: when a node disappears from the neighbors, the current node should remove all items relevant to that node from its lists. We could not evaluate SR3 in these cases due to technical reasons, as Sinalgo supports mobility only when in its synchronous mode of operation, and we use the asynchronous mode for our model. Another obvious perspective is to experiment the algorithm on a real wireless sensor network. A testbed implementation and test campaign is currently on hold. The protocol itself could also be expanded. Our first idea is to adapt the algorithm to support variable-length data packets. This would likely require a block cipher chaining mode, but the security proofs would need to accommodate this. Furthermore, an analysis using a network model which takes into account transmission times would be required. We also remark that SR3 is well suited to applications requiring a good average delivery rate over the whole lifespan of the network, but some specific attacks may strongly reduce the delivery rate for small amounts of time. We would like to investigate mechanisms that react to attacks (maybe drawing some inspiration from CASTOR [GPP+10]).

Regarding the incorruptibility notion, there are several points that could be improved upon in our definition of an incorruptible protocol. Overall, our definition needs to be extended in order to be able to accommodate more complex routing protocols. First of all, we are aware that the restriction on the modification of K during the game is a very restricting working hypothesis. However, this allows us to assume messages are routed independently of each other for several protocols, and this assumption is critical to our proofs. In a second time, our definition could be upgraded to take into account internal intruders, which have full read access over one or more node's $K[v]$. This would require to change a check in the game definition with something to ensure the message is indeed altered (and not merely re-created) in order to find meaningful attacks. Finally, the bounded corruptibility notion is quite hard to explain, and using routing protocols as bounds is not intuitive either. A graphical representation of the probabilities of the experiments succeeding would be useful. However, there are a lot of parameters, which complexifies the representation and require interactivity. We therefore would like to build such an interactive representation of the incorruptibility bounds for a set of sample protocols.

Our work on intrusion detection systems could be expanded in several ways. First, the model could be reused for IDS which are not focused on WANETs, such as host-based IDS for instance, and one can also imagine a broader model for the analysis of any IDS. Regarding the domain expansion, we avoided any availability concern on purpose, at it depends a lot on the link layer protocols. We think that there are opportunities for the improvement of the model on this aspect for an expert of that layer. There is also the possibility to add some sort of detection probabilities to the model. This would bring it one step closer to the attack tree model, increasing its precision, but at the cost of complexity. Finally, our model relies on user-provided assumptions to model the properties of the protocol stack used, such as data confidentiality or a guarantee of message unforgeability checked at each hop. However, a lot of protocols have claimed to provide such guarantees, while they actually do not. Ideally, a greater trust in the system's conclusions could be achieved by providing the means to analyze protocols together with the InDICE model, so that no ambiguities remain on the meaning of our assumptions.

Bibliography

[ABV06] Gergely Ács, Levente Buttyán, and István Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *Transactions on Mobile Computing*, 5(11):1533–1546, 2006.

[ADJL13a] Karine Altisen, Stéphane Devismes, Raphaël Jamet, and Pascal Lafourcade. Routage sécurisé et résilient pour réseaux de capteurs sans fil. In *Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel'13)*, pages 1–4, Pornic, France, 2013.

[ADJL13b] Karine Altisen, Stéphane Devismes, Raphaël Jamet, and Pascal Lafourcade. SR3: Secure resilient reputation-based routing. In *Distributed Computing in Sensor Systems (DCOSS'13)*, pages 258–265. IEEE, 2013.

[ADJL13c] Karine Altisen, Stéphane Devismes, Raphaël Jamet, and Pascal Lafourcade. SR3 supplementary material, available at `http://www-verimag.imag.fr/~rjamet/SR3/`, December 2013.

[AFP13] Nadhem J. Al Fardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *Security and Privacy (SP'13)*, pages 526–540. IEEE, 2013.

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM, 1997.

[AGS11] Michael Adeyeye and Paul Gardner-Stephen. The Village Telco project: a reliable and practical wireless mesh telephony infrastructure. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):1–11, 2011.

[AKL$^+$79] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Annual Symposium on Foundations of Computer Science (FOCS'79)*, pages 218–223, 1979.

[AR00] Martın Abadi and Phillip Rogaway. Reconciling two views of cryptography. *IFIP International Conference on Theoretical Computer Science (IFIP TCS'00)*, 1872:3–22, 2000.

[AT04] Farooq Anjum and Rajesh Talpade. Lipad: lightweight packet drop detection for ad hoc networks. In *Vehicular Technology Conference (VTC'04-Fall)*, volume 2, pages 1233–1237. IEEE, 2004.

[BAN89] Michael Burrows, Martin Abadi, and Roger M. Needham. A logic of authentication. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 426(1871):233–271, 1989.

[BCM11] David Basin, Cas Cremers, and Catherine Meadows. Model checking security protocols. *Handbook of Model Checking*, 2011.

[BDJR97] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Foundations of Computer Science (FOCS'97)*, pages 394–403, 1997.

[Bel02] Mihir Bellare. A note on negligible functions. *Journal of Cryptology*, 15(4):271–284, 2002.

[BFK$^+$13] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, and Pierre-Yves Strub. Implementing TLS with verified cryptographic security. In *Security and Privacy (SP'13)*, pages 445–459. IEEE, 2013.

[BG06] Vijay Bhuse and Ajay Gupta. Anomaly intrusion detection in wireless sensor networks. *Journal of High Speed Networks*, 15(1):33–51, 2006.

[BHvR05a] Rimon Barr, Zygmunt J. Haas, and Robbert van Renesse. JiST: An efficient approach to simulation using virtual machines. *Software: Practice and Experience (SPE'05)*, 35(6):539–576, 2005.

[BHVR05b] Rimon Barr, Zygmunt J. Haas, and Robbert Van Renesse. Scalable wireless ad hoc network simulation. *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*, pages 297–311, 2005.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In *Advances in Cryptology (ASIACRYPT'11)*, pages 344–371. Springer, 2011.

[Bla08] Bruno Blanchet. A computationally sound mechanized prover for security protocols. *Transactions on Dependable and Secure Computing*, 5(4):193–207, 2008.

[BMSU01] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[Bon97] V. J. Bono. 7007 explanation and apology. *Appears in NANOG mailing list*, 1997.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *conference on Computer and communications security (CCS'93)*, pages 62–73. ACM, 1993.

[BRC07] Kasper Bonne Rasmussen and Srdjan Capkun. Implications of radio fingerprinting on the security of sensor networks. In *International Conference on Security and Privacy in Communication Networks (SecureComm'07)*, pages 331–340. IEEE, 2007.

[BS03] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: real vulnerabilities and practical solutions. In *USENIX Security Symposium (USENIX Security'14)*, pages 15–28, 2003.

[BT10] Levente Buttyán and Ta Vinh Thong. Formal verification of secure ad-hoc network routing protocols using deductive model-checking. In *Wireless and Mobile Networking Conference (WMNC'10)*, pages 1–6. IEEE, 2010.

[CANT10] Jordi Cucurull, Mikael Asplund, and Simin Nadjm-Tehrani. Anomaly detection and mitigation for disaster area networks. In Somesh Jha, Robin Sommer, and Christian Kreibich, editors, *Recent Advances in Intrusion Detection (RAID'10)*, volume 6307 of *LNCS*, pages 339–359. Springer, 2010.

[CDPMM07] Mauro Conti, Roberto Di Pietro, Luigi Vincenzo Mancini, and Alessandro Mei. A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In *International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'07)*, pages 80–89. ACM, 2007.

[CFH06] Guillaume Chelius, Antoine Fraboulet, and Elyes Ben Hamida. WSNet: a modular event-driven wireless network simulator. `http://wsnet.gforge.inria.fr`, 2006.

[CHVV03] Brice Canvel, Alain Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In *Advances in Cryptology (CRYPTO'03)*, pages 583–599. Springer, 2003.

[CJ03] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[CLN09] Cas J. F. Cremers, Pascal Lafourcade, and Philippe Nadeau. Comparing state spaces in automatic security protocol analysis. In *Formal to Practical Security*, pages 70–94. Springer, 2009.

[CMM13] Mickaël Cazorla, Kevin Marquet, and Marine Minier. Survey and benchmark of lightweight block ciphers for wireless sensor networks. In *International Conference on Security and Cryptography (SECRYPT'13)*, pages 543–548, 2013.

[CON02] James Cowie, Andy Ogielski, and David M. Nicol. The SSFNet network simulator. `http://www.ssfnet.org/homePage.html`, 2002.

[Cre08] Cas J. F. Cremers. The Scyther tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification (CAV'08)*, pages 414–418. Springer, 2008.

[DA03] Sarjoun S. Doumit and Dharma P. Agrawal. Self-organized criticality and stochastic learning based intrusion detection system for wireless sensor networks. In *Military Communications Conference (MILCOM'03)*, volume 1, pages 609–614. IEEE, 2003.

[DDW99] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[Dis08] Distributed Computing Group at ETH Zurich. Sinalgo, simulator for network algorithms. `http://disco.ethz.ch/projects/sinalgo`, 2008.

[DLL$^+$11] Dezun Dong, Mo Li, Yunhao Liu, Xiang-Yang Li, and Xiangke Liao. Topological detection on wormholes in wireless ad hoc and sensor networks. *IEEE/ACM Transactions on Networking (ToN'11)*, 19(6):1787–1796, 2011.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[DR11] Thai Duong and Juliano Rizzo. Here come the $\oplus$ ninjas. *Unpublished manuscript*, 2011.

[DR12] Thai Duong and Juliano Rizzo. The CRIME attack. In *Presentation at Ekoparty Security Conference*, 2012.

[DS06] Murat Demirbas and Youngwhan Song. An RSSI-based scheme for sybil attack detection in wireless sensor networks. In *World of Wireless, Mobile and Multimedia (WoWMoM'06)*, pages 564–570. IEEE Computer Society, 2006.

[dSMR$^+$05] Ana Paula R. da Silva, Marcelo H. T. Martins, Bruno PS Rocha, Antonio AF Loureiro, Linnyer B Ruiz, and Hao Chi Wong. Decentralized intrusion detection in wireless sensor networks. In *ACM International Workshop on Quality of Service & Security in Wireless and Mobile Networks (Q2SWinet'05)*, pages 16–23. ACM, 2005.

[DY83] Danny Dolev and Andrew Yao. On the security of public key protocols. *Information Theory*, 29(2):198–208, 1983.

[EK07] T. Eisenbarth and S. Kumar. A survey of lightweight-cryptography implementations. *Design & Test of Computers*, 24(6):522–533, 2007.

[EOKMV12] O. Erdene-Ochir, A Kountouris, M. Minier, and F. Valois. A new metric to quantify resiliency in networking. *Communications Letters, IEEE*, 16(10):1699–1702, October 2012.

[EOMVK10a] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois, and Apostolos Kountouris. Resiliency of wireless sensor networks: Definitions and analyses. In *Telecommunications (ICT'10)*, pages 828–835. IEEE, 2010.

[EOMVK10b] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois, and Apostolos Kountouris. Toward resilient routing in wireless sensor networks: gradient-based routing in focus. In *International Conference on Sensor Technologies and Applications (SENSORCOMM'10)*, pages 478–483. IEEE, 2010.

[EOMVK11] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois, and Apostolos Kountouris. Enhancing resiliency against routing layer attacks in wireless sensor networks: Gradient-based routing in focus. *International Journal On Advances in Networks and Services*, 4(1 and 2):38–54, 2011.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In *Advances in Cryptology (CRYPTO'01)*, pages 260–274. Springer, 2001.

[FV07] Kevin Fall and Kannan Varadhan. The network simulator (ns-2). `http://www.isi.edu/nsnam/ns`, 2007.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[GMP$^+$08] Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally composable security analysis of TLS. In *Provable Security (ProvSec'08)*, pages 313–327. Springer, 2008.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[GPP$^+$10] Wojciech Galuba, Panos Papadimitratos, Marcin Poturalski, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. Castor: scalable secure routing for ad hoc networks. In *International Conference on Computer Communications (INFOCOM'10)*, pages 1–9. IEEE, 2010.

[GS69] K.R. Gabriel and R.R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969.

[GSCL$^+$13] Paul Gardner-Stephen, Romana Challans, Jeremy Lakeman, Andrew Bettison, Dione Gardner-Stephen, and Matthew Lloyd. The serval mesh: A platform for resilient communications in disaster & crisis. In *Global Humanitarian Technology Conference (GHTC'13)*, pages 162–166. IEEE, 2013.

[Han09] Robert C. Hansen. *Phased array antennas*, volume 213. John Wiley & Sons, 2009.

[HBK04] Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis. Enhancing intrusion detection in wireless networks using radio frequency fingerprinting. In *Communications, Internet, and Information Technology (CIIT'04)*, pages 201–206. IASTED, 2004.

[HCG13] Rahul Hiran, Niklas Carlsson, and Phillipa Gill. Characterizing large-scale routing anomalies: A case study of the China Telecom incident. In *Passive and Active Measurement*, pages 229–238. Springer, 2013.

[HL04] Yi-an Huang and Wenke Lee. Attack analysis and detection for ad hoc routing protocols. In *Recent Advances in Intrusion Detection (RAID'04)*, pages 125–145. Springer, 2004.

[HPJ05] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1-2):21–38, 2005.

[HSD$^+$05] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In *Conference on Computer and Communications Security (CCS'05)*, pages 2–15. ACM, 2005.

[JL13a] Raphaël Jamet and Pascal Lafourcade. Discovering flaws in IDS through analysis of their inputs. In *Fundations and Practice of Security (FPS'13)*, pages 391–407, 2013.

[JL13b] Raphaël Jamet and Pascal Lafourcade. Formal model for (k)-Neighborhood discovery protocols. In *Advances in Network Analysis and its Applications*, volume 18 of *Mathematics in Industry*, pages 181–207. Springer Berlin Heidelberg, 2013.

[JL13c] Raphaël Jamet and Pascal Lafourcade. Supplementary material for indice, available at `http://www-verimag.imag.fr/~rjamet/IDS/`, December 2013.

[JM96] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[Kar01] Brad Karp. Challenges in geographic routing: Sparse networks, obstacles, and traffic provisioning. In *Presentation at the DIMACS Workshop on Pervasive Networking*, 2001.

[KGSW05] Frank Kargl, Alfred Geiß, Stefan Schlott, and Michael Weber. Secure Dynamic Source Routing. In *Hawaii International Conference on System Sciences (HICSS'05)*, Hawaii, USA, 01/2005 2005. IEEE, IEEE.

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[KSC+05] Haeyong Kim, Yongho Seok, Nakjung Choi, Yanghee Choi, and Taekyoung Kwon. Optimal multi-sink positioning and energy-efficient routing in wireless sensor networks. In *Information Networking. Convergence in Broadband and Mobile Networking (ICOIN'05)*, pages 264–274. Springer, 2005.

[KTK02] Christopher Krügel, Thomas Toth, and Engin Kirda. Service specific anomaly detection for network intrusion detection. In *Symposium On Applied Computing (SAC'02)*, pages 201–208. ACM, 2002.

[KW03] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2-3):293–315, 2003.

[LFG+00] Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, et al. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition (DISCEX'00)*, volume 2, pages 12–26. IEEE, 2000.

[LI05] Richard Paul Lippmann and Kyle William Ingols. An annotated review of past papers on attack graphs. Technical report, DTIC Document, 2005.

[LKP07] Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *International Conference on Computer Communications (INFOCOM'07)*, pages 1307–1315. IEEE, 2007.

[LLM05] Yu Liu, Yang Li, and Hong Man. Short paper: A distributed cross-layer intrusion detection system for ad hoc networks. In *International Conference on Security and Privacy in Communication Networks (SecureComm'05)*, pages 418–420. IEEE, 2005.

[LLWC03] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and scalable simulation of entire tinyos applications. In *International conference on embedded networked sensor systems (SenSys'03)*, pages 126–137. ACM, 2003.

[LN08] An Liu and Peng Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Information Processing in Sensor Networks (IPSN'08)*, pages 245–256, 2008.

[Low95] Gavin Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information processing letters*, 56(3):131–133, 1995.

[Low97] Gavin Lowe. A hierarchy of authentication specifications. In *Computer Security Foundations Workshop (CSFW'97)*, pages 31–43. IEEE, 1997.

[LPR10] Adrian P. Lauf, Richard A. Peters, and William H. Robinson. A distributed intrusion detection system for resource-constrained devices in ad-hoc networks. *Ad Hoc Networks*, 8(3):253–266, 2010.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[LVHD$^+$05] Yee Wei Law, Lodewijk Van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. In *Workshop on Security of Ad hoc and Sensor Networks (SASN'05)*, pages 76–88. ACM, 2005.

[LZW$^+$09] Qing Li, Meiyuan Zhao, Jesse Walker, Yih-Chun Hu, Adrian Perrig, and Wade Trappe. SEAR: a secure efficient ad hoc on demand routing protocol for wireless networks. *Security and Communication Networks*, 2(4):325–340, 2009.

[Mar02] John Marshall. An analysis of SRP for mobile ad hoc networks. In *International Multiconference in Computer Science (IMECS'02)*, pages 18–21, 2002.

[McH00] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. *ACM transactions on Information and system Security*, 3(4):262–294, 2000.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology (CRYPTO'85)*, volume 218, pages 417–426, 1986.

[MRR80] John M. McQuillan, Ira Richer, and Eric Rosen. The new routing algorithm for the ARPANET. *Transactions on Communications*, 28(5):711–719, 1980.

[MSW08] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. A modular security analysis of the TLS handshake protocol. In *Advances in Cryptology (ASIACRYPT'08)*, pages 55–73. Springer, 2008.

[Mur06] Sandra Murphy. BGP Security Vulnerabilities Analysis. RFC 4272 (Informational), January 2006.

[MVdZVD+10] Nirvana Meratnia, Berend Jan Van der Zwaag, Hylke W Van Dijk, Dennis JA Bijwaard, and Paul JM Havinga. Sensor networks in the low lands. *Sensors*, 10(9):8504–8525, 2010.

[ND04] Ola Nordström and Constantinos Dovrolis. Beware of BGP attacks. *ACM SIGCOMM Computer Communication Review*, 34(2):1–8, 2004.

[NKJ+09] Hidehisa Nakayama, Satoshi Kurosawa, Abbas Jamalipour, Yoshiaki Nemoto, and Nei Kato. A dynamic anomaly detection scheme for AODV-based mobile ad hoc networks. *Vehicular Technology (VT'09)*, 58(5):2471–2481, 2009.

[OM05a] Ilker Onat and Ali Miri. An intrusion detection system for wireless sensor networks. In *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'05)*, volume 3, pages 253–259. IEEE, 2005.

[OM05b] Ilker Onat and Ali Miri. A real-time node-based traffic anomaly detection algorithm for wireless sensor networks. In *Systems Communications*, pages 422–427. IEEE, 2005.

[Pau99] Lawrence C. Paulson. Inductive analysis of the internet protocol TLS. *Transactions on Information and System Security (TISSEC'99)*, 2(3):332–351, 1999.

[PBM+04] Jonathan Polley, Dionysus Blazakis, Jonathan McGee, Daniel Rusk, and John S. Baras. ATEMU: a fine-grained sensor network simulator. In *Sensor and Ad Hoc Communications and Networks (SECON'04)*, pages 145–152. IEEE, 2004.

[PH02] Panos Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'02)*, pages 193–204, 2002.

[PJdPFWL04] Waldir Ribeiro Pires Jr, Thiago H. de Paula Figueiredo, Hao Chi Wong, and Antonio A. F. Loureiro. Malicious node detection in wireless sensor networks. In *International Parallel and Distributed Processing Symposium (IPDPS'04)*, page 24. IEEE, 2004.

[Plu82] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (INTERNET STANDARD), November 1982. Updated by RFCs 5227, 5494.

[PPJ06] Jim Parker, Anand Patwardhan, and Anupam Joshi. Cross-layer analysis for detecting wireless misbehavior. In *Consumer Communications and Networking Conference (CCNC'06)*, pages 6–9. IEEE, 2006.

[PR99] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100. IEEE, 1999.

[PS98] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *Workshop on New security paradigms*, pages 71–79. ACM, 1998.

[PST$^+$02] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[RSA78] Ronald L. Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[SHK$^+$06] Ahmed Sobeih, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Honghai Zhang, Wei-Peng Chen, Hung-Ying Tyan, and Hyuk Lim. J-Sim: a simulation and emulation environment for wireless sensor networks. *Wireless Communications*, 13(4):104–119, 2006.

[SKA04] Sameer Sundresh, Wooyoung Kim, and Gul Agha. SENS: A sensor, environment and network simulator. In *Symposium on Simulation*, page 221. IEEE Computer Society, 2004.

[SS01] Curt Schurgers and Mani B. Srivastava. Energy efficient routing in wireless sensor networks. In *Military Communications Conference (MILCOM'01)*, volume 1, pages 357–361, 2001.

[SXZC07] Hui Song, Liang Xie, Sencun Zhu, and Guohong Cao. Sensor node compromise detection: the location perspective. In *International Wireless Communications and Mobile Computing Conference (IWCMC'07)*, pages 242–247. ACM, 2007.

[TBK$^+$03] Chinyang Tseng, Poornima Balasubramanyam, Calvin Ko, Rattapon Limprasittiporn, Jeff Rowe, and Karl Levitt. A specification-based intrusion detection system for AODV. In *Workshop on Security of Ad hoc and Sensor Networks (SASN'03)*, pages 125–134. ACM, 2003.

[TBMS05] Geethapriya Thamilarasu, Aruna Balasubramanian, Sumita Mishra, and Ramalingam Sridhar. A cross-layer based intrusion detection approach for wireless ad hoc networks. In *International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05)*, pages 7–pp. IEEE, 2005.

[Thu12] P. Thubert. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552 (Proposed Standard), March 2012.

[TSB$^+$06] Chinyang Tseng, Tao Song, Poornima Balasubramanyam, Calvin Ko, and Karl Levitt. A specification-based intrusion detection model for OLSR. In *Recent Advances in Intrusion Detection (RAID'06)*, pages 330–350. Springer, 2006.

[TWKL06] Chinyang Tseng, Shiau-Huey Wang, Calvin Ko, and Karl Levitt. Demem: Distributed evidence-driven message exchange intrusion detection model for manet. In *Recent Advances in Intrusion Detection (RAID'06)*, pages 249–271. Springer, 2006.

[V$^+$01] András Varga et al. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, volume 9, page 185. sn, 2001.

[VMWS12] Jan Von Mulert, Ian Welch, and Winston K. G. Seah. Security threats and solutions in MANETs: A case study using AODV and SAODV. *Journal of Network and Computer Applications*, 35(4):1249–1259, 2012.

[WCWC07] Bing Wu, Jianmin Chen, Jie Wu, and Mihaela Cardei. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless Network Security*, pages 103–135. Springer, 2007.

[WFSH06] A.D. Wood, L. Fang, J.A. Stankovic, and T. He. SIGF: a family of configurable, secure routing protocols for wireless sensor networks. In *Workshop on Security of ad hoc and sensor networks (SASN'06)*, pages 35–48, 2006.

[WLJ06] Lingyu Wang, Anyi Liu, and Sushil Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer communications*, 29(15):2917–2933, 2006.

[WS02] David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Conference on Computer and Communications Security (CCS'2002)*, pages 255–264. ACM, 2002.

[WTB+12] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), March 2012.

[YWZC07] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. Distributed software-based attestation for node compromise detection in sensor networks. In *Symposium on Reliable Distributed Systems (SRDS'07)*, pages 219–230. IEEE, 2007.

[Zap02] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector routing. *Mobile Computing and Communications Review*, 6(3):106–107, 2002.

Appendix A

Cryptographic modeling of SR3

We provide in this annex the modeling files we used with Scyther and CryptoVerif in Chapter 2.

A.1 CryptoVerif

A.1.1 Data confidentiality

```
(************** SR3 FG modelization **************)
(*
              SR3 is a routing protocol for wireless sensor
              networks. Packets look like :

                     ( E_{k_src}(Data || N), H(N), src )

              Here's the FG game, UF in the other file.
*)

proof {
       crypto prp(enc); (* We've got to start with that, CV is stuck otherwise *)
       auto;
       success
}


(************** Parameters **************)
(* Our number of queries to the packet-generation oracle *)
param qGen [noninteractive].
param qHash [noninteractive].


(************** Basic types **************)
type nonce [large, fixed]. (* also models nonces *)
type data [large, fixed].
type block [large, fixed].
```

```
fun concat(data, nonce):block [compos].

param N [noninteractive]. (* useless and necessary *)

(*
       Important and non-trivial : a random
       block is undistinguishable from the
       concatenation of a random data and
       a random nonce.
*)

equiv
!N new b:block; a() := b
              <=(0)=> [manual]
!N new d:data; new n:nonce; a() := concat(d,n).

(************** Block cipher **************)
type keyseed [large,fixed].
type key [large,fixed]. (* for the cipher *)

proba PRPProba. (* advprp(enc) *)
expand PRP_cipher(keyseed, key, block, kgen, enc, dec, PRPProba).

(************** Hash function **************)
type hashkey [large,fixed].
(* input type : nonces *)
type hashout [large, fixed].

(* fun hash(hashkey, nonce):hashout . *)
expand ROM_hash(hashkey, nonce, hashout, hash).

(************** Queries **************)
(* the standard FG stuff *)
query secret b.

(*************** Packet generation oracle ******)
channel OGenIn, OGenOut.
let OGen =
       !qGen
       in (OGenIn, (DCPA:data));
       new n_oracle_cpa : nonce;
       let p_oracle_cpa = ( enc(concat(DCPA, n_oracle_cpa), k), hash(hk,n_oracle_cpa))
           in
       out (OGenOut, (p_oracle_cpa,n_oracle_cpa)).

channel OHashIn, OHashOut.
let OHash =
       !qHash
       in(OHashIn, hash_query:nonce);
       out(OHashOut, hash(hk,hash_query)).

(*************** The game itself ****************)
```

```
channel keyinitin, keyinitout, gamein, gameout.
process (
       in(keyinitin, ());
       new hk : hashkey;
       new ks : keyseed;
       new n_challenge : nonce;
       new b : bool;
       let k = kgen(ks) in
       out(keyinitout,());
       (
              in(gamein,(D0:data,D1:data));
              let cc = concat( (if b then D0 else D1) , n_challenge ) in
              let p_challenge = ( enc(cc, k), hash(hk,n_challenge) ) in
              out( gameout, (p_challenge, n_challenge) )

       ) | OHash | OGen

)
```

A.1.2 Nonce confidentiality

```
(************** SR3 nonce secrecy modelization **************)
(*
              SR3 is a routing protocol for wireless sensor
              networks. Packets look like :

                     ( E_{k_src}(Data || N), H(N), src )

*)

(************** Parameters **************)
(* Our number of queries to the packet-generation oracle *)
param qGen [noninteractive].
param qHash [noninteractive].
param qAnswer [noninteractive].

(************** Basic types **************)
type nonce [large, fixed]. (* also models nonces *)
type data [large, fixed].
type block [large, fixed].
type nodeId [large, fixed].

fun concat(data, nonce):block [compos].

param N [noninteractive]. (* useless and necessary *)

(*
       Important and non-trivial : a random
       block is undistinguishable from the
       concatenation of a random data and
       a random nonce.
*)

equiv
```

```
!N new b:block; a() := b
   <=(0)=> [manual]
!N new d:data; new n:nonce; a() := concat(d,n).

(************** Block cipher **************)
type keyseed [large,fixed].
type key [large,fixed]. (* for the cipher *)

proba PRPProba. (* advprp(enc) *)
expand PRP_cipher(keyseed, key, block, kgen, enc, dec, PRPProba).

(************** Hash function **************)
type hashkey [large,fixed].
(* input type : nonces *)
type hashout [large, fixed].

expand ROM_hash(hashkey, nonce, hashout, hash).

(************** Queries **************)
query event bad ==> false.

(*************** Packet generation oracle ******)
channel OGenIn, OGenOut. let OGen =
       !qGen
       in (OGenIn, (DCPA:data));
       new n_oracle_cpa : nonce;
       let p_oracle_cpa = ( enc(concat(DCPA, n_oracle_cpa), k), hash(hk,n_oracle_cpa))
           in
       out (OGenOut, (p_oracle_cpa,n_oracle_cpa)).

(**************** Hash oracle *******************)
channel OHashIn, OHashOut.
let OHash =
  !qHash
  in(OHashIn, hashOHash:nonce);
  out(OHashOut, hash(hk,hashOHash)).

(* win oracle *)
event bad().
channel WIn, WOut.
let OWin = !qAnswer
  in(WIn,( n:nonce));
  if (n = n_challenge) then
             event bad;
       out(WOut, ()).

(*************** The game itself ****************)
channel keyinitin, keyinitout, gamein, gameout.
process (
       in(keyinitin, ());

       new hk : hashkey;
       new n_challenge : nonce;
```

```
        new ks : keyseed;
        let k = kgen(ks) in

        out(keyinitout,());
        (
                (
                        in(gamein,(D:data));
                        let cc = concat( D , n_challenge ) in
                        let p_challenge = ( enc(cc, k), hash(hk,n_challenge) ) in
                        out( gameout, (p_challenge) )

                ) | OGen | OHash | OWin
        )
)
```

A.1.3 Packet unforgeability

```
(************** SR3 UF modelization **************)
(*
                SR3 is a routing protocol for wireless sensor
                networks. Packets look like :

                        ( E_{k_src}(Data || N), H(N), src )
*)


(************** Parameters **************)
(* Our number of queries to the packet-generation oracle *)
param qGen [noninteractive].
param qVerif [noninteractive].
param qHash [noninteractive].

(************** Basic types **************)
type nonce [large, fixed]. (* also models nonces *)
type data [large, fixed].
type block [large, fixed].
type nodeId [large, fixed].

fun concat(data, nonce):block [compos].

param N [noninteractive]. (* useless and necessary *)

(*
        Important and non-trivial : a random
        block is undistinguishable from the
        concatenation of a random data and
        a random nonce.
*)

equiv
!N new b:block; a() := b
                <=(0)=> (*undistinguishable*)
!N new d:data; new n:nonce; a() := concat(d,n).
```

```
(************** Block cipher **************)
type keyseed [large,fixed].
type key [large,fixed]. (* for the cipher *)

proba SPRPProba. (* advprp-cca(enc) *)
expand SPRP_cipher(keyseed, key, block, kgen, enc, dec, SPRPProba).

(************** Hash function **************)
type hashkey [large,fixed].
(* input type : nonces *)
type hashout [large, fixed].

expand ROM_hash(hashkey, nonce, hashout, hash).

(************** Queries **************)
table queries(data,nonce).
event bad().
query event bad() ==> false.

(*************** Packet generation oracle ******)
channel OGenIn, OGenOut.
let OGen =
       !qGen
       in (OGenIn, (DGen:data));
       new NonceGen : nonce;
       insert queries(DGen,NonceGen);
       let c = enc(concat(DGen, NonceGen), k) in
       let h = hash(hk,NonceGen) in
       out( OGenOut, (c, h, NonceGen)).

(*************** Packet verification oracle ******)
channel OVerifIn, OVerifOut.
let OVerif =
       !qVerif
       in (OVerifIn, (cVerif:block,hVerif:hashout));
       let concat(dVerif:data,nVerif:nonce) = dec( cVerif, k ) in
       let ( verifSuccess:bool ) = ( hash( hk, nVerif ) = hVerif ) in
       out(OVerifOut, (verifSuccess)).

(**************** Hash oracle *******************)
channel OHashIn, OHashOut.
let OHash =
       !qHash
       in(OHashIn, hashOHash:nonce);
       out(OHashOut, hash(hk,hashOHash)).

(*************** The game itself ****************)
channel SetupIn, SetupOut, ChallengeIn, ChallengeOut.
process (
       in(SetupIn, ());
       new hk : hashkey;   (*public, used for the nonces *)
       new ks : keyseed;
       let k = kgen(ks) in
```

```
	out(SetupOut,());
	(
		in (ChallengeIn, (c:block,h:hashout));
		let concat(dV:data,nV:nonce) = dec(c,k) in
		get queries(d2,n2) suchthat (nV=n2 && d2=dV) in
			out(ChallengeOut, ())
		else
			if (h = hash(hk,nV)) then (
				event bad();
				out(ChallengeOut, ())
			) else
				out(ChallengeOut, ())
	) | OHash | OGen | OVerif
)
```

A.2 Scyther

```
hashfunction H;
usertype Key;

protocol sr3(V,S)
{

	role V
	{
		fresh Nv: Nonce;
		fresh D:Ticket;
		send_1(V,S,{D,Nv}k(V,S),H(Nv),V);
		recv_2(S,V,Nv,V);

		claim_V1(V, Niagree);
		claim_V2(V, Secret, D);
		claim_V3(V, Secret, k(V,S));
		claim_V4(V, Nisynch);
		claim_V5(V, Secret, Nv);
	}

	role S
	{
		var D:Ticket;
		var Nv:Nonce;
		recv_1(V,S,{D,Nv}k(V,S),H(Nv),V);
		send_2(S,V,Nv,V);

		claim_S1(S, Niagree);
		claim_S2(S, Secret, D);
		claim_S3(S, Secret, k(V,S));
		claim_S4(S, Nisynch);
		claim_S5(S, Secret, Nv);
	}

}
```

Appendix B

Introduction en français

Contents

B.1 Introduction

Les réseaux sans-fil deviennent omniprésents. Téléphones, systèmes de paiement, infrastructure urbaine, ordinateurs : de plus en plus d'objets utilisent une forme ou une autre de communication sans-fil pour former ou rejoindre des réseaux, qu'ils soient par essence locaux ou globaux (Internet).

Dans cette thèse, nous nous intéressons aux réseaux ad hoc sans-fil (WANET). Un réseau ad hoc est un réseaux dans lequel tous les *nœuds* routent les messages, sans s'appuyer sur une infrastructure dédiée. Les communications dans ces réseaux demandent la coopération des nœuds intermédiaires si l'émetteur n'est pas dans le voisinnage direct du destinataire.

Ces réseaux semblent de plus en plus la meilleure réponse à de nombreux problèmes de collecte et de dissémination de données. Par exemple, les auteurs de [AGS11] utilisent un WANET pour partager une connection centrale à Internet parmi les habitants d'une petite ville. Ceci permet de connecter tous les habitants de cette ville, géographiquement étendue, avec des coûts raisonnables comparés au câblage chaque habitation. On peut aussi illustrer ceci avec un autre système, décrit dans [GSCL$^+$13]. Le projet Serval est construit sur le système d'exploitation Android pour téléphones mobiles. Il permet la création d'un WANET et fournit des services l'exploitant entre un groupe de personnes équipées de smartphones en se servant de leurs capacités de communication sans-fil. L'intérêt de cette méthode est d'avoir une connectivité même en l'absence d'infrastructure, par exemple après une catastrophe naturelle. Dans cette situation, les

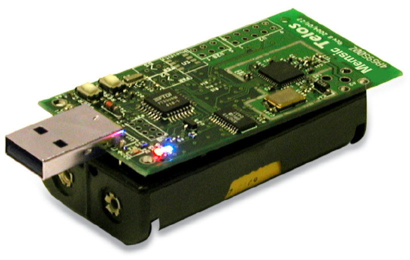


Figure B.1: Une *mote* TelosB

utilisateurs de ce système peuvent se joindre avec des appels sécurisés, faire remonter des informations, et diverses autres applications.

Les réseaux de capteurs sans fil (WSN) sont une sous-famille des réseaux ad hoc sans-fil. Ce sont des réseaux constitués de capteurs et de répéteurs interconnectés, construits pour fournir une connectivité à prix plus bas que les réseaux filliaires. Les capteurs dans ce contexte sont typiquement des objets fonctionnant sur batterie, qui génèrent des données à propos de leur environnement (*e.g.* température) pour des services spécifiques (*e.g.* surveiller d'éventuels feux de forêt). Par exemple, les *motes* TelosB (Figure B.1) sont équipées avec un microcontrolleur de 8Mhz et 10KB de RAM, qui limite leur capacité à utiliser des algorithmes cryptographiques complexes. Ces *motes* sont équipées de radios aux normes ZigBee.

Les utilisations de ces types de réseaux sont très variées : par exemple, une étude des WSN aux Pays-Bas [MVdZVD$^+$10] a montré plusieurs utilisations de ces réseaux en production. L'un d'entre eux, nommé GuArtNet[1], est utilisé pour surveiller les objets de valeur dans un musée. Plusieurs capteurs sont placés sur les objets individuels, et surveillent leur mouvement. Une alerte est levée et propagée au reste du système, à travers des répéteurs, quand un objet est déplacé : l'information remonte à un nœud central, le *puits*, qui réagira ensuite. Le puits n'est généralement pas limité en puissance ou batterie, et peut donc collecter les informations (et les *heartbeats*) des capteurs, de manière à lever une alerte dès que des problèmes de brouillage ou de batterie apparaissent.

Spécificités des WANET

Tous les WANET ont des spécificités liées à leur nature sans-fil. Leurs communications sont fortement liées aux antennes, qui peuvent être omnidirectionnelles ou avoir différents schémas d'émission directionnelles, qui émettent plus fort dans certaines directions. De plus, certaines techniques permettent aux émetteurs de diriger activement leurs émissions, en se servant par exemple de *phased array antennas* [Han09]. De plus, un lien

[1]`http://www.sownet.nl/index.php/en/products/guartnet`

peut être symétrique si les communications sont bidirectionnelles, ou asymétrique dans le cas contraire. Finalement, la qualité d'un lien fluctue au fil du temps, étant donné qu'ils sont influencés par de nombreux facteurs externes au réseau. Ces spécificités doivent être prises en compte dans la conception de protocoles pour WANET.

De plus, certaines difficultés sont liées à la nature ad hoc du réseau. Ces réseaux sont souvent collaboratifs, ce qui permet aux nœuds malicieux d'avoir plus d'effet, qu'ils soient construits par la compromission de nœuds, ou parce que le réseau est ouvert par design. De plus, certains réseaux sont dynamiques et/ou mobiles par nature, ce qui doit être pris en compte par les protocoles. Enfin, le matériel utilisé par les nœuds peut avoir des caractéristiques importantes, comme par exemple des limitations fortes sur la capacité des batteries, la puissance de calcul, le réseau peut être hétérogène, les communications très peu fiables ou à bande passante faible, etc... Tous ces facteurs sont importants et doivent être considérés.

Propriétés de sécurité et systèmes sécurisés

Les utilisateurs de ces systèmes ont des exigences de sécurité. L'exemple de GuArtNet l'illustre : intuitivement, un opérateur s'attend à ce que les alertes soient authentiques, à ce que la présence de *heartbeat* implique la présence du nœud, et que si une alerte est levée, elle atteindra le puits rapidement. La plupart des applications construites sur des réseaux ad-hoc impliquent ce type d'attentes, si ce n'est toutes. Elles correspondent à des propriétés de sécurité, comme par exemple la confidentialité ou l'authentificiation.

Pour prouver qu'un protocole satisfait ces propriétés, deux modèles principaux sont utilisés : le modèle calculatoire, et le modèle symbolique. Ces deux modèles ont été créés pour prouver des constructions cryptographiques, et même s'ils ont évolué séparément à leurs débuts, les liens entre eux ont été établis dans [AR00].

- Le modèle *calculatoire* [GM84, GMW91, BDJR97, BKR00] représente l'attaquant comme une machine de Turing probabiliste, avec un temps d'exécution polynomial et une mémoire polynomiale. Ce modèle permet d'obtenir une borne haute sur la probabilité qu'un attaquant arbitraire casse la propriété analysée. Par exemple, il a permis d'obtenir une preuve de sécurité pour le système RSA-OAEP [FOPS01].
- En ce qui concerne le modèle *symbolique* [DY83, BAN89, AG97], les données sont représentées par des symboles, qui peuvent être combinés pour obtenir d'autres symboles. Dans ce modèle, l'attaquant est capable de jouer des nouvelles sessions du protocole, et de créer des nouveaux messages dérivés de ses connaissances. Ce modèle a été utilisé pour découvrir une attaque sur le protocole de Needham-Schroeder [Low95].

Champ d'examen d'une preuve et sécurité en profondeur

Les preuves de sécurité garantissent que dans le cadre d'un modèle spécifique, le protocole analysé garantit la propriété de sécurité observée. Ces preuves peuvent être automatisées, auquel cas les étapes de la preuves sont clairement identifiées, évitant ainsi toute erreur possible dans le cas de preuves manuelles. Cependant, comme avec tous les modèles, il se peut qu'il y aie des différences avec la réalité, et les attaques hors du champ d'examination ne seront pas détectées.

Pour illustrer ceci, on peut mentionner l'attaque récente (Mars 2014)[2] sur la suite de protocoles TLS, couramment utilisée sur Internet (et adoptée par l'IETF [DR08]). TLS a un dispositif nommé *three-way handshake*, utilisé pour créer des canaux sécurisés entre les participants.

TLS propose plusieurs protocoles durant ce handshake, et un attaquant peut se servir de trois d'entre eux afin de rediriger des messages entre deux participants Alice et Bob, de manière à ce que les deux pensent communiquer directement entre eux. Cependant, la vulnérabilité permet à l'attaquant d'injecter des données dans ce que Bob pense avoir reçu d'Alice. C'est une brèche d'authentification et d'intégrité, dûe au mécanisme de secret maître de TLS permettant de réutiliser des valeurs déja vues. Dans notre contexte, l'intérêt est de voir que TLS a été formellement prouvé de nombreuses fois dans divers modèles de sécurité [Pau99, HSD$^+$05, GMP$^+$08, MSW08, BFK$^+$13]. On peut aussi mentionner que plusieurs vulnérabilités ont été précédemment trouvées dans TLS, autant au niveau protocole que dans ses implémentations (par exemple, [CHVV03, DR11, DR12, AFP13]). Cette vulnérabilité dans TLS montre que même si des preuves formelles prouvent la sécurité d'un système dans un modèle donné, certaines attaques peuvent ne pas être dans le champ d'examination. Dans ces cas là, avoir de la sécurité en profondeur est importante : par exemple, un système de détection d'intrusion peut être capable de détecter ce type d'attaques, soit par leur signature (un comportement caractéristique d'une attaque), soit plus généralement une anomalie (un comportement inhabituel du système). Ensuite, un système de réponse aux intrusions pourra réagir, par exemple en alertant un humain pour une analyse subséquente.

Évolution des besoins

Dans certains cas, des systèmes qui n'étaient pas critiques au moment de leur création sont devenus critiques. Des exemples de ce types abondent dans les protocoles utilisés sur Internet, comme ce réseau reposait initialement sur la confiance mutuelle entre ses acteurs. Cet état d'esprit a fortement influencé l'infrastructure logicielle sous-jacente à ce réseau.

Par exemple, le protocole ARP (*Address Resolution Protocol*, standardisé par l'IETF dans [Plu82]) est fréquemment utilisé dans les réseaux locaux pour faire le lien entre les adresses logiques (IP) et physiques (MAC). Schématiquement, un nœud en cherchant un autre broadcaste un paquet demandant "qui a cette IP", requête à laquelle répondra le possesseur de l'IP. Ce protocole n'inclut cependant pas de mécanisme d'authentification, et un intrus peut se servir de ceci pour usurper une addresse IP sur son réseau local. Sur des réseaux filiaires, ce risque semble assez faible, à condition de contrôler qui est câblé. Cependant, ce protocole est utilisé aussi dans le cadre de réseaux WiFi, ou la menace est plus sérieuse. Pire, si le réseau est ouvert, n'importe quel intrus peut usurper une

[2]`https://secure-resumption.com/`

addresse IP, facilitant les attaques de l'homme du milieu (MITM). On peut aussi citer BGP, un autre protocole dans cette situation, qui décide la manière dont les paquets seront routés les paquets entre entités autonomes dans Internet. Des incidents arrivent encore régulièrement depuis une vingtaine d'années, ou une entité déclare des informations erronées voire malicieuses, qui influent la manière dont sont routés les messages légitimes. Cette influence peut être utilisé à des fins de surveillance, dégrader artificiellement la qualité de service d'une entreprise, par exemple. Pour répondre à ce type de problème, nous proposons dans le Chapitre 3 une mesure quantitative de l'influence possible d'un attaquant sur les routes utilisées par un message, et nous définissons la propriété correspondante (nommée *incorruptibilité*).

B.2 Contributions et organisation

Cette thèse est construite en trois grands chapitres. Notre première contribution est SR3, un protocole Sécurisé de Routage, qui est Résilient et à base de Réputation. Ce protocole est construit pour le routage convergent dans les WSN, que nous présentons dans le Chapitre 2. Nous avons conçu ce protocole comme une marche aléatoire qui est renforcée à l'aide d'un mécanisme de réputation, qui est alimenté à l'aide d'informations sûres. Nous avons prouvé trois propriétés de sécurité dans le modèle calculatoire à l'aide de *CryptoVerif* [Bla08], et dans le modèle symbolique avec *Scyther* [Cre08]. Nous nous sommes ensuite servis de *Sinalgo* [Dis08] pour l'évaluation expérimentale du protocole via simulations, et nous l'avons comparé à plusieurs autres protocoles de la littérature : Greedy-Face-Greedy [BMSU01], la marche aléatoire uniforme, SIGF [WFSH06], et plusieurs variantes du routage par gradient [EOMVK11]. Cette expérimentation nous a permis de montrer la résilience, l'efficacité et l'équité de SR3.

Notre seconde contribution a aussi trait au routage. Dans le Chapitre 3, nous présentons une définition pour la sécurité des protocoles de routage que nous nommons *incorruptibilité*. Les autres mesures de la sécurité du routage que nous avons trouvé sont seulement applicable aux protocoles de *source routing*: la nôtre mesure la capacité pour un attaquant à altérer significativement la manière dont les messages seront routés dans le réseau. Nous présentons en premier la formalisation de cette notion, suivie par une généralisation que nous nommons *corruptibilité bornée*. Nous fournissons plusieurs protocoles d'exemple, ainsi que les preuves associées, et nous présentons les possibles extensions requises à notre notion pour accomoder une plus grande variété de protocoles.

Notre troisième contribution a trait aux systèmes de détection d'intrusion (IDS) pour réseaux ad hoc sans-fil. Ces systèmes sont critiques pour la sécurité de ces réseaux, et malgré ceci, leur évaluation est souvent expérimentale. Nous proposons donc dans le Chapitre 4 deux résultats pour leur amélioration. Nous avons donc choisi de nous centrer sur leurs sources de données. Dans un premier temps, nous avons effectué un état de l'art de ces sources de données, et nous avons fourni une classification de celles-ci basé sur le niveau de collaboration entre les nœuds qu'elles nécéssitent, ainsi que sur la couche réseau sur lequel elles reposent. Dans un second temps, nous avons proposé InDICE (*Intrusion Detection Inputs Coverage Evaluation*), un modèle pour trouver automatiquement des failles dans les IDS, en s'appuyant uniquement sur leurs sources de données. InDICE utilise un modèle basé sur des règles et des anomalies afin de représenter la progression de l'attaquant en direction de buts prédéterminés. Nous appliquons ensuite cet outil à deux IDS de la littérature ([dSMR$^+$05] and [OM05b]).

B.3 Publications

L'essentiel du travail original présenté dans ce manuscrit a été publié ou est en cours de publication dans des conférences internationales. Le protocole SR3, et les résultats associés présentés dans le Chapitre 2, ont été présentés à DCOSS'13 [ADJL13b], ainsi qu'à la conférence nationale Algotel'13 [ADJL13a]. Une version journal est également en cours de jugement. Tout le travail présenté dans le Chapitre 4 a été publié à FPS'13 [JL13a], et le travail du Chapitre 3 a été accepté à FPS'14 et sera publié sous peu au moment de l'écriture de ces lignes.

Enfin, nous avons construit un modèle pour vérifier les protocoles de découverte de voisinnage dans les réseaux sans-fil. Ce travail a résulté en un modèle capable de prendre en compte les mouvements de nœuds et le temps, et nous l'avons utilisé pour vérifier un protocole simple de la littérature. Nous avons également proposé des mécanismes pour qu'un protocole puisse déterminer le $n+1$-ou-moins voisinnage d'un nœud, à partir de la connaissance des n-voisinnages. Tout ce travail a été publié dans un livre [JL13b], et nous ne l'avons pas inclus dans ce document.

Appendix C

Perspectives en français

Nous avons plusieurs pistes sur des extensions possibles pour notre étude de SR3. Pour commencer, SR3 semble adapté aux réseaux qui présentent quelques noeuds mobiles, et également pour les réseaux avec une mobilité plus faible de tous les noeuds simultanément. Algorithmiquement, seuls un changement mineur est requis : quand un noeud disparaît d'un voisinnage, son identité doit être purgée des différentes listes. Pour des raisons techniques, nous ne pûmes pas évaluer SR3 dans ces conditions, car Sinalgo ne supporte la mobilité que dans les simulations synchrones. Une autre perspective directe est de tester l'algorithme dans un banc d'essai (*testbed*). Une campagne de test sur capteurs réels est actuellement en suspens. Le protocole lui même peut aussi être étendu. Notre première idée est d'adapter l'algorithme afin de gérer des paquets de données de taille variable. Cette modification requiert vraisemblablement l'utilisation d'un mode de chainage pour le chiffrement, et par conséquent les preuves de sécurité devront être revues. Une autre extension possible serait de prendre en compte les temps de transmission dans le modèle d'analyse. Enfin, nous avons remarqué que SR3 est adapté aux applications demandant un taux de livraison en moyenne bon au fil de la vie du réseau, mais certaines attaques ciblées peuvent réduire fortement ce taux de livraison de manière ponctuelle. Nous aimerions donc chercher des mécanismes qui réagiraient directement aux attaques, par exemple en s'inspirant de CASTOR [GPP$^+$10].

Pour la notion d'incorruptibilité, nous voyons plusieurs points pouvant être peaufinés dans les définitions. Globalement, notre définition doit être étendue afin de pouvoir gérer des protocoles plus complexes : principalement, l'interdiction des modifications de K pendant le jeu est une hypothèse de travail très limitante. Cependant, elle est aussi critique pour nos preuves. Dans un second temps, notre définition pourrait être étendue afin d'accomoder des attaquants internes au réseau, qui auraient un accès total en lecture à un ou plusieurs $K[v]$. Ceci nécéssiterait de changer des vérifications dans la définition du jeu afin de s'assurer qu'un message est effectivement altéré (et pas juste recréé), dans l'optique de trouver des attaques significatives. Enfin, la corruptibilité bornée est une notion assez obtuse, et l'utilisation de protocoles comme bornes est peu intuitive. Une représentation graphique des probabilités que l'*experiment* réussisse serait une grande aide à la compréhension. Cependant, la quantité de paramètres à prendre en compte rend plus complexe la représentation, et nécéssitera peut-être de l'interactivité. Nous aimerions donc pouvoir construire une telle représentation.

Notre travail sur les détection d'intrusions pourrait être étendu de plusieurs manières. Premièrement, le modèle peut être réutilisé pour les IDS qui ne sont pas centré sur les WANET, comme par exemple les *host-based* IDS. On peut également imaginer un modèle plus large, pour l'analyse de n'importe quel IDS. Nous avons évité toute considération d'attaques par déni de service, parce qu'elles dépendent fortement de la couche liaison du réseau. Nous pensons que cet aspect peut être développé par un expert de ces attaques. Il serait aussi possible d'ajouter des probabilités de détection dans le modèle, pour s'approcher de l'exemple des *attack trees*. Ceci augmenterait la précision au cout de complexité ajoutée. Enfin, notre modèle repose sur de nombreuses hypothèses fournies par l'utilisateur sur les propriétés des protocoles réseau utilisés. Cependant, beaucoup de protocoles ont erronément prétendu fournir ces propriétés, et idéalement, une plus grande confiance en les conclusions de notre modèle pourrait être obtenue en fournissant les moyens aux utilisateurs de prouver ces propriétés eux-mêmes. Ainsi, toute ambiguïté sur le sens de ces propriétés serait levée.