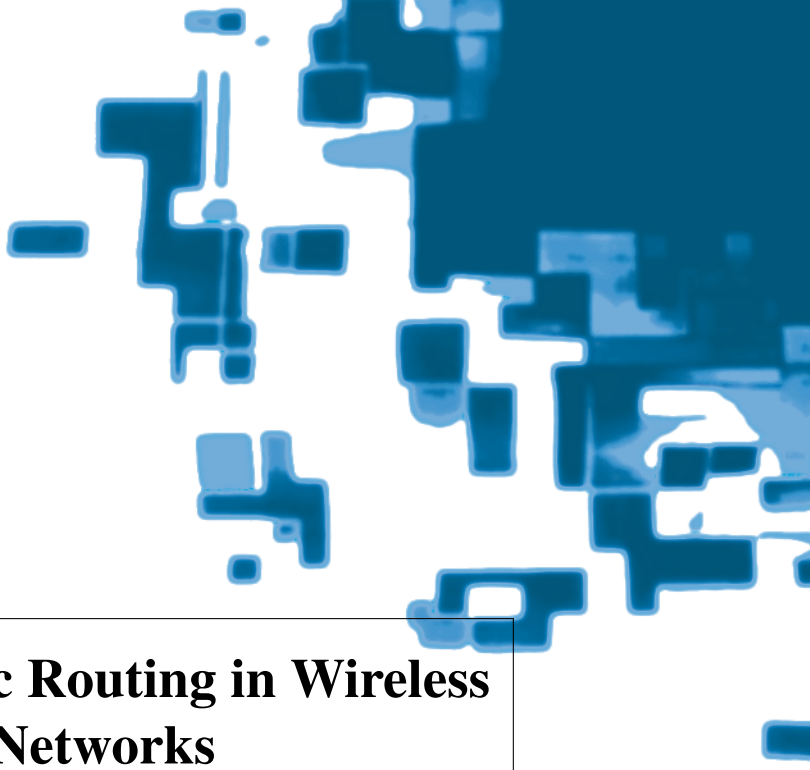

Secure Probabilistic Routing in Wireless Sensor Networks

Karine Altisen, Stéphane Devismes, Pascal Lafourcade, Clément Ponsonnet

Verimag Research Report n^o TR-2011-15

November 7, 2011

Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

Secure Probabilistic Routing in Wireless Sensor Networks

Karine Altisen, Stéphane Devismes, Pascal Lafourcade, Clément Ponsonnet

November 7, 2011

Abstract

In this report, we propose a provably secure routing protocol for wireless sensor networks (WSNs) based on random walks using tabu lists.
Many approaches have been developed in the literature to solve security problems of routing protocols for WSNs. But most of them are not resilient against attacks from compromised nodes because they only use cryptographic solutions.
In our work, we design a new secure routing protocol to improve this resiliency using realistic assumptions such as asynchronous communication. We propose a probabilistic approach that allows us to save energy and to avoid malicious nodes with a detection system of safe paths using tabu lists, acknowledgments and trust counters. Our main result is that, even in presence of compromised nodes, our protocol ensures data confidentiality, integrity, authenticity and a high delivery rate.

Keywords: Security, Secure routing, Routing protocol, Random Walk, Tabu list, Wireless sensor networks

Reviewers: Stéphane Devismes

Notes: This report is based on the Master thesis realised by Clément Ponsonnet

How to cite this report:

```
@techreport {ADPP-RR-2011,
  title = {Secure Probabilistic Routing in Wireless Sensor Networks},
  author = {Karine Altisen, Stéphane Devismes, Pascal Lafourcade, Clément Ponsonnet },
  institution = {{Verimag} Research Report},
  number = {TR-2011-15},
  year = {2011}
}
```

Contents

Introduction

A *Wireless Sensor Network* (WSN) consists of thousands of low cost sensors that communicate through wireless media (typically, radio transceivers). They could either have a pre-defined location or be randomly deployed to monitor the environment. We focus on a crucial problem for WSN: *routing*, and especially, how to secure routing. The goal of the routing is to transmit data from a sensor (so called the *source*) to another one (the *sink* or the *destination*). Two of the main issues in such a network is *energy-saving* to increase its lifespan because sensors are equipped of a small battery. And to ensure security properties because WSNs are sensitive to node capture due to the fact that sensors are cheap and deployed in an open environment.

Context

Sensor nodes are extremely basic in terms of interfaces and components. Mostly, they are just composed of a sensor, a processing unit with limited computational power and limited memory, a wireless system of communication and a limited capacity power source. The sensors are used to collect physical information such as temperature, sound, vibration, pressure, motion or pollutants. All these data may have an interest in many areas (see Appendix A). For example, we can deploy a network to monitor a forest fire outbreaks or even monitor all movements on a battlefield.

The neighborhoods of nodes define an implicit autonomous network: the WSN. We consider that two sensors are neighbors in the network if they are in range of each other (that is, the communication network is a UDG, Unit Disk Graph). We also note that the communication can be unreliable due to the message collisions. However, we assume here that communication primitives are reliable but asynchronous (the alternating bit protocol [BSW69] can be used to turn unreliable communication into reliable yet asynchronous communication). Throughout this report, the network will be always supposed connected.

A routing protocol encapsulates control data into messages to follow a path from the source to the sink. As in many WSNs, we are in the case of *All-to-One routing* where all source nodes must be able to transmit data to a single sink on request or according to a schedule. The sink can be arbitrarily far (in terms of hops) from other nodes. A WSN must match different needs according to applications. In the case of a fire, the most important is that the information arrives as soon as possible, or if we want to observe natural phenomena precisely, it is better to be sure to receive all messages.

Keeping in mind that one of the biggest problems of a WSN is its lifespan, routing protocols have to satisfy security properties. Since an attacker can easily eavesdrop communications in wireless network, most of applications requires the confidentiality of information routed through the network. Like other communication protocols, integrity and authenticity of messages are required, sensed data collected in the sink being unusable otherwise. Finally, the major security problem of WSNs is the resistance against denial of service attacks. In addition, its weaknesses increase when the WSN is deployed on a large scale in an unsecured area. The availability of the network is exposed to physical constraints, *i.e.*, an attacker can simply destructs or moves sensor nodes and communications can be disrupted with collisions or jamming signal. Moreover, adversaries are able to compromise nodes to launch internal

attacks. These attacks are more effective because they affect the entire network availability. Attack effects being amplified by energy constraints of a WSN, the network can be quickly destroyed.

Challenge

Routing protocols in WSN must consider security as an essential goal. It is crucial that the design of routing protocols finds a good compromise between the need of security and the energy cost of solutions. Consequently, secure routing is a considerable challenge, especially due to the severe constraints of WSNs. The limited capabilities of the nodes and the inherent characteristics of their deployment clearly distinguish these networks from other wireless networks like cellular networks or ad hoc networks. We thus study new approaches to solve this problem in the case of WSNs.

Many secure routing protocols [GGSE01] [WFSH06] [DHM06] and security mechanisms [ZLLF06] [PST+02] have been developed for WSNs. They propose cryptography techniques well suited to WSN because they minimize computational costs and communication costs. The number and size of messages is optimized and currently symmetric encryption is required. Some of them even manage to mitigate the effects of false routing information but they are sensitive to attacks from insider nodes. The inside attacker has the same access to the WSN as legitimate nodes. They can therefore change the behavior of compromised nodes by flooding the network or by selectively dropping packets. But they also can change the behavior of other sensor nodes by creating routing loops or by falsifying their neighbor tables.

We propose secure routing protocols which ensure a maximum of message arrivals. We take account of a strong adversary model, *i.e.*, adversaries are not limited by resources, are authorized to participate to network and can collaborate between us. We use standard cryptographic method to secure communications through the network. The protocol design is based on probabilistic routing using a detection method of safety paths, calling "safety path", a path avoiding compromised nodes. Our goal is to improve the number of messages that reach the sink keeping a reasonnable energy consumption. Moreover, we take realistic assumptions such as asynchronous communications or the impossibility to locate sensor nodes.

Outline

In first part, we introduce studied problematic. We present related work, *i.e.*, security properties, adversaries, attacks and proposal solutions in WSN literature. Then we explain our approach presenting probabilistic routing, assumptions chosen, specific attacks and solution ideas.

In second part, we present our secure routing protocols giving algorithms and security proofs of standard properties in the random oracle model. Finally, we study by experimentation, the protocol efficiency in term of number of hops to route messages and the network availability using delivery rate of messages.

Part I

Problematic

Chapter 1

Security for Routing in WSN

Security is a crucial problem in many WSNs and solutions dedicated to other networks (*e.g.* wired) are not well suited because a WSN has limited resources. Foremost, we need to properly define the posed problem in this context: What security services must ensure the WSN? What are the capabilities of the adversary? What are the attacks that the adversary is able to run to affect routing mechanisms? And what are the solutions discussed in the literature? We thus introduce security properties and adversary models in the WSN context. Then, we present attacks on routing protocols and security solutions.

1.1 Security Properties

A secure routing protocol in a Wireless Sensor Network must satisfy many security properties. Most of these properties are well-defined in recommendation X.800 [UIT91], so we explain properties in the WSN context based on this reference. Each property depends on the application for which the network is established. So, here we list required properties and some current avenues of research.

1.1.1 Confidentiality

Confidentiality, also called *secrecy*, is keeping information for the intended recipients. A secret data cannot be learned by an unauthorized person. In WSN, an adversary has access to messages due to wireless communication, so we may want to protect exchanged information such as sensed data or routing information between two nodes or between a node and the sink. In [PSW04] and in [CKM00], authors explain that cryptography is the standard defense. So, they propose to focus on encryption mechanisms suitable to WSN and key management mechanisms. For example in [EG02], Eschenauer and Gligor present a key management mechanism characterized by the choice of a random subset of keys from a large key pool pre-installed in each sensor before deploying the network.

1.1.2 Authenticity

Authenticity is the fact to ensure that someone or something is what it claims to be. So, a receiver needs an authentication mechanism that verifies that the received packet comes from the announced node. As it is proposed in the recommendation X.800 [UIT91], we distinguish two cases. First, the authentication of the origin of the data which can be achieved in the sink or in an aggregator node. We can note that various models of data aggregation verifying data authenticity have been proposed in the literature such as in [CMT05] or in [CÖN+06] where data are already encrypted before aggregation. In the second case, we are interested in the authentication of nodes that participate in the communication because it is a crucial problem to save energy. For example, a robust and energy efficient solution is given in [XyQLx09].

1.1.3 Integrity

This property ensures that data has not been modified during the packet exchange. In [Sen11], Sen specifies data *integrity* in WSN saying that "The mechanism should ensure that no message can be altered by an entity as it traverses from the sender to the recipient". So integrity is the detection of any tampering of data and in WSN the detection is achieved by recipients. Generally, the usage of hashing algorithms is used to ensure integrity but there are also watermarking schemes which reduce the size of packet headers (and so the energy consumption) as this has been proposed for WSN context in [JKK08].

1.1.4 Availability

Availability is the fact of being accessible and usable upon demand by an authorized entity. Here, availability ensures that functionalities or services provided by the WSN is always available even in presence of attacks or the crash of a node. This property being crucial and hard to obtain, many different notions have been proposed and many different approaches have been studied in the literature.

First, various Denial-of-Service type attacks are identified in [WS02]. The goal of these attacks is to "diminish or eliminate a network's capacity to perform its expected function". Here, protocols are very sensitive to DoS attacks due to the nature and characteristics of WSNs. So, it is crucial to take account this security goal during the design phase. The essential network service which must be ensured is the availability of sensed data in the sink. We want to ensure a maximum successful delivery of message to its recipient. This service is measured with the delivery rate of messages which is the number of messages emitted by sources divided by the number of messages received by the sink. A more precise approach is to study the quality of service. For example in [PLOP08], authors discuss security in WSN through an evaluation of QoS computing mathematically *reliability*, *availability* and *serviceability*, *i.e.*, they express these notions using the mean time between failure and the mean time to repair.

Different terms have been introduced to define the ability to ensure this service or rather the ability to make the network available as much time as possible. A clear distinction between *robustness*, *resiliency* and *survivability* is proposed in [EOMVK10]. So a *robust* network accommodates hardware and software failures or physical disruptions to continue to provide services. Typically, it survives different internal problems such as the loss of links (or nodes). For example in [PSW04], authors specify the term of robustness speaking of "robustness to communication denial of service", *i.e.*, the resistance against disruption of communications. The definition of *resiliency* is clearly given in [EOMVK10] as follow: "Resiliency is the ability of a network to continue to operate in presence of k compromised nodes, *i.e.*, the capacity of a network to endure and overcome internal attacks". Many works study this property for data aggregation in WSN such as in [Wag04] or in [YWZC08]. While a multipath routing is a solution idea published in [GGSE01] to solve the routing problem of resilience to node capture. Finally, the term of *survivability* is defined as "the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents" in [MEL+00] and it includes previous distinctions. This property is present in many networks system as it is explained in [WDXZ10]. But most recent works are produced in the WSN context with for example a survivable key management scheme proposed in [LY06].

There exist several factors influencing the availability and thus the means to attack. For example, the *load-balancing* avoids some nodes to drastically traffic more than other ones. The benefits of load balancing are studied in [PH09] and in [DDK09]. Clearly, if a node on the path between the source and the sink is abnormally preferred, this node consumes more, then the expected lifetime of the node decreases and can influence the lifetime of the network. Another aspect of the load-balancing from a security point of view and availability, is the fact that when a node concentrates traffic, then protocols

are more vulnerable at this level.

1.1.5 Other Security Properties

We can denote other security properties in the WSN literature or in the literature of system security. We now present a part of these properties which are even more application dependent.

Privacy

We distinguish several kinds of privacy problems in WSN that we can differentiate according what the adversary can learn. First, context *privacy* is generally specific to types of collected data (applications). For example, in [GL05] and [HG06], authors are interested in location privacy in pervasive environments, they want to avoid that an adversary knows the locations and times that the subject spent at this location. We also have privacy of medical data from body sensors [TWZL08] [SCK07] with for example a person who uses a pacemaker does not want that others know that he carries it.

In [PSW04], Perrig *et al.*, highlight the risks concerning our privacy on collected data, *i.e.*, WSN can be used for spying unaware people. There are two cases: either the network is deployed in the goal to observe individuals such as for example an employer monitoring its employees or a government agency spying citizens. In this case, we need new laws to respond to problems related to technological advance.

Either sensor networks is diverted from its initial legitimate use and generally, this new use is unanticipated and illegal. Users of WSN do not want that their personal information are publicly exposed. So the collection and dissemination of sensor data must be done confidentially and anonymously.

A survey of privacy-preserving techniques for WSNs was published in [LZDT09]. As previously, authors also present privacy through two types of private information: data-oriented and context-oriented privacy.

Freshness

In recommendation X.800 [UIT91], this property is expressed as integrity of a sequence of data units, *i.e.*, we want to be protected against sequencing errors, loss, repetition, insertion or modification of data. In WSN, staleness of the data can be a real drawback because some applications need recent data (or not) to decide to act. So we need a mechanism that checks the *freshness* of the data as it is noted in [Sen11] or in [AFN08]. Classic solutions are number of sequence, timestamp or cryptographic chaining. Generally in WSN literature, *freshness* includes data freshness and key freshness that ensure message freshness.

Access Control

As it is expressed in [SLSZ06], the *access control* avoids unauthorized nodes to use resources. Clearly, we can not prevent the sending of message (wireless communication) but we can prevent nodes to forward it. For example, a solution using node authentication and key establishment protocol is proposed in [ZZF07] to prevent malicious nodes from joining the sensor network.

Non-Repudiation

Non-Repudiation ensures that a node cannot refute the sending of a message to an other node or the reception of a message from an other node. This property is considered as basic in [SLSZ06] but is not that much present in the WSN literature. It is certainly due to the fact that this property is implicitly included in integrity or authentication properties. We can also remark that *Non-Repudiation* is not necessarily required for WSNs.

Reliable Data Readings

The WSN can report false data due to the fact that a participating node can be corrupted or sensed data can directly be falsified. For example, an adversary puts a lighter next to a temperature sensor. So we want to avoid that the sink accepts these false data. For example in [KKT09], authors present a scenario of opportunistic sensing where trustworthiness of data becomes crucial.

1.2 Definition of the adversary

In [EOMVK10], Authors present definitions of adversaries. There are classical security notions as passive or active attackers. But they also define other concepts that are more specific to wireless sensor network as capacities of attacker which are mote-class or laptop-class and the point of initiation of the attack (inside or outside security perimeter).

1.2.1 Intruder Capabilities

In [DY83], Dolev and Yao introduce a formal model to precisely study security problems for protocols. Their goal was to address the problem of "active saboteur" in the design of protocol. We define a *passive attacker* as an attacker which tries to learn information eavesdropping on the network. He can only listen to the communication. While an *active attacker* has many more capabilities. He is able to intercept, modify, block and replay messages or also generate and insert new messages. Its goal is to modify operations that must normally occur.

1.2.2 Attacker Resources

The distinction between *mote-class attacker* and *laptop-class attacker* was introduced in [KW03]. *Mote-class attackers* have access to sensor nodes with similar capabilities to our ordinary sensor nodes, *i.e.*, same battery power, same radio transmitter, *etc*. A *laptop-class attacker* may have access to more powerful devices such as laptops. So, attacker has great advantages over ordinary nodes. He has the possibility to do more than mote-class attackers because he has a better range with large antenna or a powerful radio transmitter. The computational power or memory storage are not restricted by a CPU with low power consumption or a small memory size. For example, a single laptop-class attacker is able to eavesdrop and to jam the entire network. We can specify the range of a laptop-class attacker in terms of the range of k-sensor nodes. For example a 2-laptop-class attacker has twice more range than a standard node. Consequently, he can directly communicate with a node at distance 2.

1.2.3 Access to WSN

Attacks can be characterized according to the authorization to participate in the WSN of the attacker who launches the attack [KW03]. The *outsider attacker* has no allowed access to the network, it is external to the network. An outsider attack is produced by an unauthorized or illegitimate user of the system. An *insider attack*[1] is produced by an authorized or legitimate user of the system. The attacker is authorized to participate in the network. Typically, these attacks come from compromised nodes running malicious code or from adversaries (malicious nodes) using secrete keys, code or data stolen from the legitimate nodes. We can even express the authorization level of an insider attacker specifying how many nodes he has compromised. For example, we assume that each node shares a key with its neighbors to secure the channel between two nodes (a unique key per link). Then if the attacker has compromised only one node, he is just authorized to communicate with the neighbors of the compromised node. This precision

[1] Also called byzantine nodes in distributed systems [LSP82]

is useless in the simple case where a unique key is shared between all nodes belonging to the network because if this key is compromised, then attacker is authorized to communicate with all nodes.

1.2.4 Attacker Movement

The attacker movement was introduced in [WCG+05] for physical attacks in WSNs. First, we have *static attacker* which is a stationary node, *i.e.*, he is at a fixed place in the network. This place is important because several attacks are more or less effective according to the distance between malicious node (launcher of the attack) and the sink. Otherwise, *mobile attacker* can change its position in the network. The malicious node has the mobility. It can follow the critical path according to the information learned by eavesdropping the traffic.

1.2.5 The number of attackers

In [WS04], authors consider attackers according to their capabilities such as the number of attakers and their coordination. An adversary can be clearly characterized by the number of attackers that produce an attack. So he may exist one single attacker or several attackers with two different behaviors. Either, there are several independant attackers in the network, *i.e.*, attackers do not cooperate. Or, they cooperate with a shared memory or with a channel of communication. Consequently, they can share their information and combine their actions.

1.2.6 Relations between definitions

We note that certain features of the adversary are included in others and some are independent. So, we define the partially ordered set $(A, \leqslant)$ with A the set of adversaries and $\leqslant$ a partial ordering. An element of A is the quadruplet (A_1, A_2, A_3, A_4) which defines each characteristic of the adversary. Clearly, we have:

- $A_1 = \langle Capabilities \rangle = \{passive, active\}$ with $passive \subseteq active$
- $A_2 = \langle Resources \rangle = \{mote\text{-}class, laptop\text{-}class\}$ with $mote\text{-}class \subseteq laptop\text{-}class$
- $A_3 = \langle Access \rangle = \{outsider, insider\}$ with $outsider \subseteq insider$
- $A_4 = \langle Movement \rangle = \{static, mobile\}$ with $static \subseteq mobile$

The relation $a \subseteq b$ means that capacities of a are included in capacities of b. The partial ordering is also defined for each characteristic, so $\forall a, b \in A, a \leqslant b \Leftrightarrow (a_1 \subseteq b_1) \wedge (a_2 \subseteq b_2) \wedge (a_3 \subseteq b_3) \wedge (a_4 \subseteq b_4)$. The partially ordered set $(A, \leqslant)$ is a lattice because any two elements, there exists a least upper bound and a greatest lower bound.

1.3 Attacks against Routing Protocols

We present specific attacks against routing protocols that only affect the availability because confidentiality, integrity, authenticity and privacy are ensured by cryptographic primitives such as symmetric encryption or message authentication code. These attacks have been published in [KW03]. We assume that the adversary is always static and active. Moreover, most of attacks are protocol-specific.

1.3.1 False routing information

The adversary injects false routing information into the network by spoofing, altering or replaying them. He creates disruptions of the traffic such as creation of routing loops, generation of false message, causes network partitioning or attracts (or repels) network traffic from selected nodes. Consequently, he increases delay (time delivery) and decreases lifetime of the network.

1.3.2 Selective forwarding

In WSN, it is assumed that nodes faithfully forward all received messages. So a malicious node (insider) selects messages to be dropped and transmits other messages. Consequently, data may be lost. For example, for the blackhole attack, all messages are dropped. This attack is more efficient when the malicious node is closed to the sink, *i.e.*, when much traffic goes through the malicious node.

1.3.3 Sinkhole attack

An adversary makes a malicious node (insider) which appears more attractive to its neighbors using routing information. It attracts the traffic from a larger area in the network. Typically, he is a laptop-class attacker (not necessarily) which creates metaphorical sinkhole providing a high quality route to the sink. Due to the large traffic going through sinkhole nodes, this attack becomes even more efficient when it is associated with selective forwarding attacks.

1.3.4 Sybil attack

This attack is developed in [NSSP04]. A sibyl node (insider) assumes the identity of several nodes. The malicious node fills the memory of a neighboring node with useless information from non existing neighbors. The sybil node pretends to be present in different parts of the network. So it can eliminate nodes from the routing table. This attack mostly affects geographical routing protocols.

1.3.5 Wormholes

There is a tunnel that sends packets received on one part of the network to another. The adversary (laptop-class attacker) use this low-latency link between two nodes to apply other attacks. It relays packets between two distant nodes to convince them that they are neighbors. So it can forward a packet to a part of the network away from the sink or it can still forward it to a sinkhole.

1.4 Routing Security

Security solutions are proposed to improve WSN routing protocols. These solutions provide different secure services that can be associated to obtain a good trade-off between resources constraints and the need to ensure security properties. We present security protocols optimized for sensor networks using cryptographic primitives and specific design. They focus on routing functionalities that provide authentication, message integrity, confidentiality, replay protection, key management, secure neighborhood discovery and message availability. We review some of these suitable solutions here.

1.4.1 Communications Security

Adrian Perrig *et al* [PST+02] proposed two secure building blocks: SNEP and μTESLA. SNEP offers several security properties and advantages for WSNs. It provides two-party data authentication, integrity, confidentiality and freshness using message authentication code and encryption. Due to resource

constraints, it encrypts data with a block cipher in counter mode (CTR) avoiding to transmit the counter value by keeping state at both end points. So, it is well-suited to WSN because it has low communication overhead adding only 8 bits per message. μTESLA is an authenticated broadcast based on delayed key disclosure which adapt TESLA [PCTS02] protocol to WSN context. It uses purely symmetric primitives and self-authenticating keys (one-way chains). It encompasses several advantages such as a low overhead and a robustness to packet loss. But it has a major drawback because it requires an additional service which is the time synchronization.

There are improvements of SNEP which have been proposed in [KSW04] and [LMPG07]. They provide similar security services but they are more energy efficient using different block cipher modes of operation.

1.4.2 Key management

To secure communications, we need to secure key management scheme despite the following important constraints: high probability of node capture, wireless communication, small memory size and computational power in nodes. Large surveys have already been conducted [SJBMC10] [ZV10] [RALS11]. Here we present major schemes of literature.

Several research groups [GPW+04], [WKfC+04], [GKS04] have successfully implemented public-key cryptography but for many WSN applications that want to deploy cheap sensors during a very long time (ideally 20 years), these solutions are not pratical. For the same reasons, we cannot use hybrid key establishment schemes for wireless sensor networks proposed in [HCK+03] and [ZV08]. Even if they exploit well the differences of resource constraints among the base station and different kinds of sensors.

We focus our approach on symmetric key management schemes and we quickly present different types of schemes through examples. First, a probabilistic key pre-deployed scheme is proposed in [EG02]. This scheme establishes communication keys in three steps. In the key pre-distribution step, a subset of a large pool of keys is stored in each sensor. After each sensor discovers its neighbors with their common keys. Then, the protocol establishes a path key used for secure links between pair of nodes. This scheme allows revocation of keys when a node is compromised. A trusted node broadcasts identifiers of compromised keys and these keys are removed from sensors. Instead, a deterministic scheme is introduced in [LS05] based on the star-like tree. Compared to probabilistic scheme, it reduces the number of keys by almost 50% using a hash function. There are also other key management algorithms based on binary tree [WHA99], matrix [Blo85] [YG05] or polynomial [LNL05] [BSH+92]. Most of these schemes have a *t-secure* property in the sense that in a network with N nodes, the collusion of less than $t+1$ nodes cannot reveal any key shared by other pairs of nodes.

In [ZSJ03], a Localized Encryption and Authentication Protocol (LEAP) is proposed. This scheme uses a pre-distributed master key to establish either a group key or a cluster key or a pairwise key or only an individual key shared with the sink. It uses one-way key chains to broadcast authentication, so it is efficient to limit storage in nodes and energy consumption. Moreover, it improves the resilience restricting the impact of a node compromise to its neighborhood. Another method to protect against node capture is proposed in [CP05], where authors describe Peer Intermediaries for Key Establishment (PIKE) that uses trusted nodes to facilitate key establishment.

1.4.3 Secure Neighborhood Discovery

Secure neighbor discovery is essential for almost every routing protocol in WSNs. This problem is complex and is usually treated as an independent secure building block. The difficulty is accentuated by the high dynamicity of the network. Moreover due to wireless communication, outsider attackers can significantly affect the efficiency of protocols. They just need to forward packets between two non-neighboring nodes (Wormhole attacks) or blur packets to avoid the link establishment (jamming attacks). Many defenses against the wormhole attack are proposed in literature such as in [HPJ03] where authors

present geographic and temporal leashes to restrict the maximum transmission distance of a packet or in [HE04] where they use directional antennas. This distance bounding approaches was introduced in [BC94]. Still using physical characteristics, authors try to overcome jamming in [XMTZ06] with either spectral evasion (channel surfing), spatial evasion (spatial retreats) or adjusting resources (power levels and communication coding). Finally, insider adversaries can easily modify neighborhood tables of its neighbors with false information. They create a false local view of the network for the legitimate nodes, and so they affect operation protocols.

1.4.4 Routing Defense

A clean-slate approach is presented in [PLGP06]. It operates in three phases: network planning, establishment and maintenance of path routes. This protocol has a preventive action because it detects compromised nodes using authenticated broadcast and Grouping Verification Trees (GVT). Furthermore, it uses multiple routes that naturally avoid compromised nodes. This technique of multi-path is used in other secure routing protocols such as in [GGSE01] because this allows a better delivery rate than deterministic routing [RPC+07]. An interesting intrusion-tolerant routing protocol (InSense) [DHM06] also employees this technique. It uses a unique symmetric key only shared between a node and the sink to save memory and isolate the attack in case of node capture. In this protocol, the sink built multipath routing tables after checking neighborhood topology information and then sent them to each node. A drawback of this protocol is the flooding of the network by route request messages. In the presence of many compromised nodes, this number of request messages drastically increases and the protocol becomes too power greedy. In [WFSH06], authors propose Secure Implicit Geographic Forwarding (SIGFE) which offers different security levels. They add security mechanisms in an existing protocol [BHSS03] to create three secure extensions. Obviously, more the extension is secure, more the protocol consumes energy.

Finally, there are many security solutions in the WSN literature but these solutions are very expensive in energy (synchronization, control messages) and do not protect routing against all attacks (in particular internal attacks). We thus propose an approach to design an energy efficient routing protocol that ensure security properties in presence of strong adversaries.

1.5 Conclusion

We presented security properties (confidentiality, integrity, authenticity and availability) that we consider indispensable, classes of attackers in a WSN, attacks applied on routing protocols and defenses developed in the literature. We have shown that attacks launched by insider attackers are very harmful for the network availability. As literature is rather scarce when we consider defenses against these attackers, our goal is to design a routing protocol for WSN that ensures security properties in presence of any class of attacker.

We are interested in the probabilistic routing to obtain a good tradeoff between the service quality and the cost of the protocol. This routing method does not give the maximum delivery rate but the energy cost is low compared with methods ensuring a delivery rate of 100% in WSN.

Chapter 2

Random Walk Routing Protocols

We decide to improve the security of probabilistic routing protocols. A random walk routing protocols have already been used in large scale WSN [SB02] to optimize energy consumption. This class of routing protocols computes the next hop to route a message according to a transition probability on the fly. This choice of probabilistic routing is motivated by the fact these protocols are naturally well suited to secure the routing in WSNs because insider attackers cannot learn information on the next hop choice. Moreover, this has been corroborated by the work published in [EOMVK10] where the resilience of some routing protocols without any security protections is studied. They show security features that are inherent in different routing protocols. And they advocate stateless routing protocols to avoid the proliferation of specific attacks and random behavior to prevent the adversary from determining which are the best nodes to compromise. Our approach authorizes the fact of losing messages because it is more expensive in energy to ensure a delivery rate of 100%. But it offers a good delivery rate compared with the energy consumption increasing due to the large number of hops and the time to stabilize.

We consider the All-to-One routing in a connected network with bi-directional and reliable links. We also assume that the neighborhood discovery is already performed and that each node has a unique ID. So here, we focus on the routing problem assuming that neighborhood tables are already built by lower layers (solutions have been given in Section 1.4.3).

First, we present a class of probabilistic routing protocols that have been designed last year in the team and then attacks that can be launched against these protocols. Finally, we propose an avenue of research to protect one of them.

2.1 Random Walk Routings

There are random walks as mathematical objects or random walks used as routing protocols. We present different variants of the random walk routing algorithm which is a probabilistic routing protocol.

2.1.1 Random Walk

In random walk routing, each message is sequentially routed: until the message reaches the sink, the next destination is uniformly randomly selected among the neighbors of the current node. This protocol is well-suited for WSNs, since those algorithms imply a low volume of computation and exchanged information. However, these low costs come at the expense of increasing the *hitting time*, *i.e.*, the expected number of hops to reach the sink. But several techniques are proposed to reduce the hitting time in the random walk maintaining their good energy-saving properties.

2.1.2 Random Walk Using Local Degrees

In the classical random walk (RW), the worst case of the hitting time is in $O(n^3)$ where n is the number of nodes. This complexity comes from the fact that using a uniform probability, the frequency of visits of a node depends on its *local degree*, *i.e.*, nodes with an high degree are visited more often than nodes with a low degree. Based on this remark, Yamashita *et al* [SIY09] improves the hitting time by using non-uniform probabilities to choose the next destination of the message. The probability $P(u)$ to choose a neighbor u of node v as next destination now depends on the local degree of u:

$$P(u) = \frac{\delta_u^{-\frac{1}{2}}}{\sum_{w \in \mathcal{N}_v} \delta_w^{-\frac{1}{2}}}$$

where $\mathcal{N}_v$ is the set of neighbors of the currently visited node v and δ_v is the degree of node v. Using this probability law, the worst case of the hitting time decreases from $O(n^3)$ to $O(n^2)$. In this work, we denote by $P^u_{\texttt{RWLD}}$ the probability to choose u according the probability of the random walk using local degree (RWLD) proposed by Yamashita.

2.1.3 Random Walk with Tabu List

In [ADLP10], we improve RWLD by augmenting it with small tabu lists. We chose this random walk because it offers the best hitting time. We propose two approaches mixing tabu lists and random walks. The first approach avoids cycles by using a tabu list of visited nodes inside each message. The second approach considers several tabu lists per node (one per neighbor) coupled with an accusation counter so as to detect cycles.

First, we recall some notations and management of tabu lists introduced in [ADLP10] and then we describe algorithms.

Lists Policies Nodes are assumed to have distinct identifiers. No distinction is made between a node and its identifier, *i.e.*, the identifier of node v is simply denoted by v. The test $e \in L$ is true if and only if the list L contains element e. We use the following functions to manage a list:

- `remove(`e, L`)` removes all occurrences of element e (if any) in the list L.
- `update(`e, L, sz`)` updates the list L of size sz, *i.e.*, adds element e according to the two following policies:
  - In FIFO[1] update lists (FIFOupdate), the first element to leave the list, is the older one (First In First Out). When the list is updated with some element *elt*, it is enqueued at the end of the list; before that, if *elt* was already in the list, it is removed in order to have only one occurrence of an element in the list. Finally if the list is full, the oldest element is removed.
  - In Random update lists (RANDupdate), the order of the list is meaningless: the next element to leave is uniformly chosen among possible elements. When the list is updated with a new element *elt*, if *elt* is not already in the list then it is simply enqueued if there is some place for it in the list. If the list L is full, L is replaced by the list L', which is obtained by removing a element selected uniformly at random from L enqueued by *elt*.

[1] Due to the fact that the list accepts only one occurrence of an element, the FIFO policy is also ususally called LRU policy

Tabu List in Messages In the first approach, we consider a protocol where a tabu list is attached to each message. The list contains node identifiers and keeps track of a part of the path the message has followed. As much as possible, the message avoids to visit a node in its list. Therefore, the random walk with tabu list in messages (TLM) uses this list to *avoid cycles*. It prevents the message to turn back to already visited nodes.

The probability to choose the next node in RWLD is modified in that sense: let v be a node and L a list of nodes to avoid; we defined the probability $P^v_{\texttt{TLM}}(L) : \mathcal{N}_v \to [0;1]$ to select a node u among the neighbors of v, apart from L by

$$\forall u \in \mathcal{N}_v \setminus L, \quad P^v_{\texttt{TLM}}(L)(u) = \frac{\delta_u^{-\frac{1}{2}}}{\sum_{w \in \mathcal{N}_v \setminus L} \delta_w^{-\frac{1}{2}}}$$
$$\forall u \in \mathcal{N}_v \cap L, \quad P^v_{\texttt{TLM}}(L)(u) = 0$$

Notice that if $L = \emptyset$, then $P^v_{\texttt{TLM}}(L)$ is the probability law used in RWLD. Note also that, if $\mathcal{N}_v \subseteq L$ (*i.e.*, at least all neighbors of u are in the tabu list), then $P^v_{\texttt{TLM}}(L)$ is not well-defined, so we use the probability law of RWLD instead, that is $P^v_{\texttt{TLM}}(\emptyset)$. This protocol comes in two variants according to the update policy of the list: TLM (sz,FIFOupdate) and TLM (sz,RANDupdate), where $sz \in \mathbb{N}$ is the bound on the size of the tabu lists.

Tabu List in Nodes In the second approach, tabu lists are stored in nodes (TLCN) and contain message identifiers. Actually, at each node v, there is a tabu list $TL[u]$ attached to each possible destination $u \in \mathcal{N}_v$. When a message leaves node v through some link (v, u), then the message identifier is added to the corresponding tabu list, $TL[u]$. This enables to detect that a message followed a cycle: when it comes back to some node, if its identifier is present in the list of the link through which it previously left the node, we are able to detect two links of the cycle: the link through which the message previously left the node and the link through which it comes back to the node.

At each node v, an *accusation counter*[2] $Counter[u]$ is also associated to each possible destination $u \in \mathcal{N}_v$ (henceforth, to each tabu list). Such a counter is used to remember how many times the node v detects that a message followed a cycle containing this link. Tabu lists and counters are therefore used to *detect cycles*: the higher the counter of a link, the more the link leads to some cycles. Thus, a node preferably sends the messages through the links with small counter values. The selection of the next link then uses the probability law $P^v_{\texttt{TLCN}}$ which is derived from the one of RWLD. $P^v_{\texttt{TLCN}}(Counter) : \mathcal{N}_v \to [0;1]$ selects a node u among the neighbors of a node v, is weighted by the array of counters $Counter$, and is defined as follows:

$$P^v_{\texttt{TLCN}}(Counter)(u) = \frac{Counter[u]^{-1} \times \delta_u^{-\frac{1}{2}}}{\sum_{w \in \mathcal{N}_v} Counter[w]^{-1} \times \delta_w^{-\frac{1}{2}}}$$

As for TLM, this protocol comes from two variants depending on the update policy and the bound sz on the size of the list: TLCN (sz,FIFOupdate) and TLCN (sz,RANDupdate).

2.2 Attacks against random walk routings

In [ADGFT08], we obtained several random walk algorithms that use more or less information to improve the number of hops. These algorithms have been studied in terms of performances, especially in

[2] This method was originally used in fault-tolerant algorithms, see [ADGFT08].

terms of hitting time and volume. Now, we try to break them. Specific attacks have been introduced in Section 1.3. We developed them in the case of random walk routings. We show that protocols using information to route messages are more vulnerable because the attacker can target this information. However, these protocols are inherently more tolerant to attacks than deterministic protocols, as their random behavior cannot be predicted by the adversary.

2.2.1 False routing information

Due to the fact that `RW` and `RWLD` does not use control information such as distance to the sink, attackers cannot create this type of attack for these protocols.

For `TLM`, malicious nodes can modify routing information introducing false routing information. They can change tabu lists when they receive messages. So they can create routing cycles such as in Figure 2.1. In this Figure, we assume a short part of the WSN composed of three legitimate nodes (A, B and C) and one intruder I. In `TLM`, the size of tabu list is 2 and we give the example of a message routing generated by A. First, it sends the message to B which forwards it to C. Then, C has to sent the message to I. The intruder has access to the tabu list and it puts the A's identifier to prevent the message to go out the cycle (I,B,C). This attack has two negative effects: messages carrying a cycle are lost, *i.e.*, they do

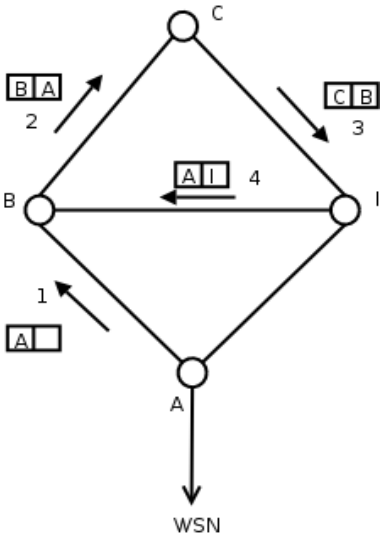

Figure 2.1: Message going around cycles through B, C and I

not reach the sink and nodes visited during the cycle exhaust their energy.

In the `TLCN` protocol, the adversary is able to choose counter values of legitimate nodes. Consequently, it influences the network availability with an increase of the hitting time because it may decide to attract or repel network traffic from selected nodes. The attack operates as follows:

1. The adversary observes the traffic of a targeted node using wireless communication.
2. It can thus obtain a message m and the link l by which m is sent.
3. It can choose to increment or not increment counter associated with l according to its wish. For this, it has several ways. For example, it sends m to the targeted node when it wants to increment the counter. Or on the contrary, It blocks m or sends other messages to remove m from the tabu list.

Finally, the adversary manages to increase the hitting time after a certain time.

2.2.2 Selective forwarding

In this attack, the message arrives in the malicious node and the adversary chooses to forward or not. Therefore, only the delivery rate of random walk routings is affected by this attack. For example, if we assume that malicious nodes lose 100% of messages received (blackhole), then we note that malicious nodes and the sink have the same behavior since they retain messages. Consequently, the average rate of delivery drastically decreases depending on the number of compromised nodes. This attack becomes more efficient when malicious nodes are placed next to the sink. Indeed, the adversary isolates the sink from the rest of the network. As the distance to the sink, the degree of malicious nodes has an impact because nodes with a high degree are visited more often in `RW`. Nevertheless, the probability to choose the next destination of the message proposed by Yamashita *et al* [SIY09] uses local degree to equalize the frequency of visits of a node. So, `RWLD`, `TLM` and `TLCN` are naturally protected against the impact of malicious node degrees.

2.2.3 Sinkhole attack

Malicious nodes want to attract the traffic from a larger area in the network. We can describe three methods to create sinkhole in random walk routings. As we have previously explained, in `RW`, messages pass through high degree nodes more frequently. If the adversary increases its degree distorting links, then it increases its traffic. This technique only affects the `RW` due to the use of local degree in the transition probability matrix of `RWLD`, `TLM` and `TLCN`.

Two other methods have already been presented. First, when adversary knows the location of the sink, it can compromise nodes closer to the sink that are more attractive. The second method uses the introduction of false routing information (section 2.2.1) to attract the traffic in `TLCN`. Finally, the last way is to produce Sybil attacks.

2.2.4 Sybil attack

In a Sybil attack, malicious nodes present multiple identities to legitimate nodes. This attack causes two effects on random walk routings. When compromised nodes fill the neighborhood tables with identities of non-neighboring nodes or identities built, then the adversary increases the average degree of the graph. The first effect is the loss of messages when the legitimate node chooses a false identity as next hop. But this problem should be solved by a secure neighborhood discovery.

The second effect is when the malicious node said to a legitimate node that it is its neighbor with many different IDs. This implies that the targeted node thinks it has a lot of different neighbors when in reality many links point to a single node (Sybil node). Consequently, the probability that the Sybil node receives the message increases.

2.2.5 Wormholes

Wormholes forward messages between two nodes which are at a large distance from each other. Random walk routings have the advantage to be designed to render meaningless these attacks. Indeed, the wormhole attack creates a false network topology but random walks do not use this information, unlike other routing protocols such as overlay-based protocols or geographic routing protocols. Nevertheless, if we mix wormhole and sinkhole, then the adversary is able to damage routing. For example, a wormhole situated close to the sink can transmit messages to a part of the network far from the sink.

In summary, the design of random walks already provides a good protection against attacks on the routing from strong adversaries because the number of possible attacks is enough limited. Attackers try to disrupt services provided by the WSN in two ways: they augment the number of nodes visited to

decrease the network lifespan, and they prevent the delivery of messages combining sybil attacks, wormholes and the introduction of false routing information to attract traffic in malicious nodes (sinkholes) that launch a selective forwarding attack.

2.3 Security of TLM

The major problem is the management of tabu lists because they are vulnerable: nodes routing the message must have access to this list. We introduce an avenue of research to secure TLM using Bloom filters [Blo70] to minimize the size of tabu lists and to scramble the contents of the list. This work is in progress, so proposals have not been proved and studied in depth. First, we introduce Bloom filter structure and how to use it in TLM. Then we describe solution ideas to secure TLM against outsider and insider attackers.

2.3.1 Bloom filters

A Bloom Filter (BF) is a simple space-efficient randomized data structure for representing a set in order to support membership queries. This membership test is probabilistic, *i.e.*, if the query returns "NO", we are sure that the element is not in the set whereas if the query returns "YES", we have a certain probability of its presence. This problem of false positives is widely compensated with the space advantage and moreover, adding and querying an element occurs in constant time. BFs are well suited to store tabu lists in messages because it is used to quickly verify the presence of an element belonging to a set and it implies low communication overhead.

Description A BF representing a set S of size n is an array of m bits which is initially set to 0. We fix the size of the BF and choose k hash functions such that their output interval corresponds to the size of the array. For example, if $m = 32$ then hash functions return a number between 1 and 32 which is an index of the table.

To add an element to the BF, we compute the k hash values. Each value represents an index of the array. The bits corresponding to these indexes are then set to 1.

When we want to know if an element e belongs to S in the BF, we compute hash values of e and we have two cases: if an array element is 0, then e does not belongs to E with certitude, else all array elements are 1 and so, e belongs to E with a probability p that e is a false positive. As hash functions guarantee that the addresses generated are an uniform distribution on all possible values, we can estimate p by $(1 - e^{\frac{-kn}{m}})^k$.

Application to TLM In WSNs context, we limit the energy consumption minimizing the size of packets. Thus, the space-efficient data structure (BF) is used to store the tabu list in messages. Moreover this structure introduces a degree of randomness in the contents of the tabu list.

When a node receives the message containing the BF, it verifies membership of neighboring nodes with their ID. According to test responses, it chooses the next hop and update the list. However, the algorithm of tabu list update is modified because the deletion of an element can cause the deletion of other elements.

The use of a BF implies two changes in TLM. In first, we remark that the BF can only be used with a random update policy due to the fact that there is no way to distinguish the recent elements from the past ones, since no time information is kept. In second, we need a solution to remove elements. A method is introduced in [FCAB00]. They maintain a small counter instead of a single bit. When an item is inserted, the corresponding counters are incremented whereas when an item is deleted, the corresponding counters are decremented. If a counter does overflow, then we leave it at its maximum

value (implying false negatives). But as deletions are random, this event is relatively scarce. In [DR06], authors add a random deletion operation into the BF so that it does not exceed its capacity. The proposed stable BF takes in input a maximum number of elements in the BF, a number of hash function, the size of the BF and it is able to provide acceptable rates of false positives and false negatives.

2.3.2 Suggesting ways to secure TLM

We use classical cryptographic protections to ensure integrity, authenticity, confidentiality and to improve the availability. We present informal solutions (*i.e.*, unproven) to secure TLM according to the satisfied properties and the class of attacker. Moreover, we can also note that the energy consumption increases with the degree of security.

Secure TLM *Vs* Outsider attacker Weaknesses of this protocol are tabu lists in messages. We want to prevent the adversary to choose elements of the tabu list. We assume that pairwise keys are established with a t-secure key management scheme. A basic solution idea is to add message authentication code (MAC). This system guarantees the authenticity and the integrity of the message. In particular, we know that when the tabu list has been modified by an attacker because he cannot generate a valid MAC.

Data are encrypted by symmetric keys to ensure confidentiality. So now, the outsider attacker can only affect the availability. As it possesses any keys, it cannot create a valid message. Nevertheless, due to wireless communication, the adversary can easily replay messages and so reduce the network lifespan. A random number used only once ("nonce") allows to resist to replay attacks. It is generated by a pseudo-random number generator in order that the recipient detects an adversary using old communication.

Finally, to protect TLM against outsider attacker, we construct a message m sent by a node A to a node B as in the following:

$$m = \{Data \,||\, nonce\}_{k_{AB}} \,||\, BF \,||\, MAC(k_{AB}, \{Data \,||\, nonce\}_{k_{AB}} \,||\, BF)$$

with $Data$ the packet to be routed, BF the tabu list saved in a Bloom filter and k_{AB} the shared key between A and B.

To avoid the use of a nonce and to keep security requirements, we can adapt SNEP[3] [PST+02] protocol adding the BF to the message. In this protocol, Perrig *et al* use a CTR mode of operation to avoid replay attacks. They also analyse precisely the security properties and their running. This kind of approach is widely present in literature [KSW04] [LMPG07] where the energy consumption is optimized and operation modes of the block cipher are different.

Secure TLM *Vs* Insider attacker If we assume that for each sensor node, an individual key is shared with the sink then, confidentiality, integrity and authenticity can be ensured between the source node and the sink with a similar scheme to the previous. But the problem of the tabu list management is impossible to solve in presence of compromised nodes. Indeed, the dilemma is the following: How to allow access to the BFs only for the legitimate nodes whereas we cannot distinguish them from malicious nodes. The only protection to ensure avaibility is the inherent property of random walk routings that have a non-zero probability to avoid the crossing of malicious nodes. This limits the degradation of network services.

Nevertheless, we might envisage to use a solution based on reputation [MGLB00] to improve TLM.

In conclusion, this first part of the report introduced the problematic of our work: how secure a routing protocol in WSN? We studied the state of the art around this problematic, and we observed that solutions are usually not effective in terms of delivery rate or energy consumption when many nodes are

[3] This secure building block has been presented in Section 1.4.1, it provides two-party data authentication, integrity, confidentiality and freshness using message authentication code and encryption

compomised by the adversary. Then, we proposed and justified a probabilistic approach to improve the efficiency of the routing in WSNs. We showed the existing attacks on random walk routing protocols. Therefore, in the next part, we propose a new random walk routing protocol that will be protected against these attacks, in particular, we focus on the improvement of the delivery rate without increasing the energy cost.

Part II

Contribution: A Secure Protocol

Chapter 3

A New Routing Protocol

Ideally, a secure routing protocol should guarantee the integrity, authentication, confidentiality and availability of messages in the presence of adversaries of arbitrary power. However, in the presence of insider adversaries, especially those with laptop class capabilities, it seems that some, if not all, of these goals are not fully attainable. An important goal of secure routing protocols is thus to ensure a maximum of message arrivals taking account of a strong adversary model. To achieve the above goal requires that a routing protocol degrades no faster than a rate approximately proportional to the ratio of compromised nodes to total nodes in the network.

We propose to adapt the routing protocol `TLCN` proposed in [ADLP10] and explained in Section 2.1.3 into a secure random walk routing with tabu lists in nodes (`TLCN-S`). We uses the same idea, *i.e.*, it is a random walk for which tabu lists are maintained in nodes and the message is preferably sent to some neighbor nodes according to counters. The main difference is that counters now provide a measure of confidence that a node has on some neighbors. We describe the basic algorithm of `TLCN-S` which is analyzed and studied in terms of security performance in the next chapters. Then we propose ideas of optimization which can make the algorithm more efficient.

3.1 The `TLCN-S` Protocol

In the introduction of Chapter 2, we explained the advantages of random walks routings for the resiliency of protocols. The goal of our protocol is to find a trade-off between the delivery rate and the cost of the algorithm. Indeed, the delivery rate can collapse when malicious nodes participate to WSN. So we need to improve it by affecting the minimum possible energy consumption. Our mechanism to improve the delivery rate is now presented. Then we add cryptographic primitives to ensure other properties. Finally, we precisely describe the final algorithm of `TLCN-S` and we show that nodes can only trust on their own behavior implying that multisource routing is necessary.

3.1.1 Algorithmic Avoiding of Malicious Nodes

The main idea is to avoid malicious nodes during the routing of messages. The detection of these nodes is usually complex and takes time and energy. We thus decide not to detect malicious nodes but to identify safe paths using a learning mechanism. For this, we rely on the fact that if a message has reached the sink, then it has a low probability of having encountered malicious nodes. Next, the sink informs legitimate nodes that it has received a message. Using an acknowledgement containing the identifier of the received message. Consequently, when a node u receives the acknowledgement of a message that it has previously routed, then it can assume that the neighbor node that was chosen to forward the message is legitimate. u has detected a safe link. Since this method works locally for each nodes, we can construct a distributed algorithm detecting the entire safe path. This detection mechanism

has to face to several issues such as what are the information that nodes need to store, or how nodes will choose the next hop. We manage these issues using a similar mechanism to TLCN, *i.e.*, we use tabu lists to store message IDs in nodes, we increment counters to save detected safe links and we use the probability law associated with the counter to choose the next hop. We also note that security properties are currently not satisfied. Cryptographic tools are added to ensure the operation of the detection and required properties in the next section.

The protocol is thus built as follows: At each node v, there is a tabu list $TL[u]$ attached to each possible destination $u \in \mathcal{N}_v$ ($\mathcal{N}_v$ denotes the set of neighbors of node v). When a message is created by the node v, it leaves v through some link (v, u), then the message identifier is added to the corresponding tabu list, $TL[u]$. The message is randomly routed to the sink which returns an acknowledgement containing the message identifier. This enables to detect that a message reached the sink: when the node receives an acknowledgment of its message, its identifier is present in the list of the link through which the message delivery was successful. We are able to detect the link through which routing until the sink works. Therefore at each node v, a *trust counter* $Counter[u]$ is also associated to each possible destination $u \in \mathcal{N}_v$ (henceforth, to each tabu list). Such a counter is used to remember how many times the node v detects that a message reached the sink through this link. Tabu lists, acknowledgments and counters are therefore used to *detect successful delivery*: the higher the counter of a link, the more the link leads to the successful delivery of the message. Thus, a node preferably sends the messages through the links with high counter values. The selection of the next link then uses the probability law $P^v_{\texttt{TLCN-S}}$ which is derived from the one of RWLD. $P^v_{\texttt{TLCN-S}}(Counter) : \mathcal{N}_v \to [0; 1]$ selects a node u among the neighbors of a node v, is weighted by the array of counters $Counter$, and is defined as follows:

$$P^v_{\texttt{TLCN-S}}(Counter)(u) = \frac{Counter[u] \times \delta_u^{-\frac{1}{2}}}{\sum_{w \in \mathcal{N}_v} Counter[w] \times \delta_w^{-\frac{1}{2}}}$$

Notice that when all counter values are equal, then the probability law is the one of RWLD.

We just created a new routing protocol more resilient to internal attacks. When the WSN is compromised by the capture of several nodes, we improves the delivery rate using a mechanism to detect safe paths until the sink. The protocol must be multi-source and it requires the confidentiality of the message ID. We thus use cryptographic primitives to obtain the confidentiality and the other security properties.

3.1.2 Cryptographic Mechanisms

The availability of TLCN-S protocol is improved by its probabilistic routing and by the detection of secure paths. We ensure data confidentiality using a block cipher. The source node encrypts the data and the message ID with the symmetric key only shared between itself and the sink. The adversary (outsider or insider) should not be able to obtain a single bit of information about this ciphertext. Finally, a message authentication code is used to verify the integrity and authenticity of the message. This MAC is the hash of the message ID that we concatenate to the message. The sink checks the integrity computing the hash of the message ID. Since the message ID is always the same during communication, an attacker can use the hash of an intercepted message to authenticate messages that it has forged. We use a random number used only once (nonce) to identify a message. This nonce prevents the adversary to use previous hashes. Furthermore, as the nonce is generated by a cryptographically secure pseudo-random number generator, the adversary cannot guess the next number used in messages. The sink node decrypts the ciphertext and it is sure that the message has been forged by the declared source node.

We recap the TLCN-S protocol improved by cryptographic mechanisms. Tabu lists are stored in the nodes and contain message identifiers, *i.e.*, random nonces. Messages are built such that data and the random nonce generated for this message are concatenated, then encrypted and finally sent with the hash of

the nonce and source ID. In each node, the next hop is chosen using the probability $P_{\texttt{TLCN-S}}(Counter)$. When the sink node receives a message, if the verification was successful, *i.e.*, if the sink has been able to decipher the nonce and to verify its hash, then an acknowledgment composed of the nonce is sent. Cryptographic mechanisms ensure confidentiality, integrity and authenticity of our protocol.

3.1.3 The TLCN-S Algorithm

First, we present preliminary assumptions that TLCN-S needs to work. Then we introduce notations used to formally write the algorithm.

Assumptions and Notations In this protocol, each source node is equipped with a tabu list, a counter for each outgoing link and a secret key used in a block cipher. The tabu list of a link contains identifiers of a part of the messages that were sent through the link. The counter of a link represents the number of time (plus one) the protocol detected that the link was involved in a successful delivery to the sink, namely some message passed through this link to reach the sink. The secrete keys are symmetric. These keys are pre-installed (during the coding phase) such that a node has only one key that is shared with the sink. We remark that the sink node has all the keys. Nodes also share a public hash function and pseudo-random number generator. We know that the routing must be multi-source to allow the counter incrementation.

In the descrition of the TLCN-S algorithm, we use the same update policies (fifo ou random) of tabu list that in Section 2.1.3. We denote by $\perp$ an empty list and by $||$ the concatenation of two elements. Other notations have an explicit name.

Overview of Algorithm TLCN-S (Algorithm 1) At the beginning (Lines 6-11), all the tabu lists are empty and the counters are set to one.[1]

When a source node v produces a new datum $Data$ (Lines 12-21), it first generates a random nonce N and computes the hash H of this nonce. Then it concatenates $Data$ with N and encrypts them to obtain the ciphertext C. Finally, this ciphertext is encapsulated in a message labelled by (H, S) where H uniquely identifies the message among all others whose source ID S is v. Hence, the label (H, S) uniquely determines the message in the network. Now, the message $\langle C, H, S\rangle$ is sent to the next node using the probability law $P^v_{\texttt{TLCN-S}}(Counter)$ and the tabu list of the link is updated with N.

When a source node v receives a message $\langle C, H, S\rangle$ from a node u (Lines 22-25), it just randomly forwards the message according to the probability law $P^v_{\texttt{TLCN-S}}$ which gives preferences to links with high counters. v does not check the message and does not change the message identifier from its list.

When the sink node s receives a message $\langle C, H, S\rangle$ from a node u, it decrypts the cipher using the key associated with the node source S of the message. Then it verifies the message validity computing the hash of the nonce. If the message is valid, then $Data$ is delivered to the upper layer and the sink sends an acknowledgment to u. This acknowledgment is composed of destination node ID S and the nonce N because this proves that the sink has received the message.

When a source node v receives an acknowledgement $\langle N, S\rangle$ from a node u (Lines 26-34), we have two cases. Either v is not the node targeted by the acknowledgement (*i.e.*, $v \neq D$), then it just forwards $\langle N, D\rangle$ using the probability law $P^v_{\texttt{RWLD}}$. Or v is the destination of the acknowledgement (*i.e.*, $v == D$), then it verifies the presence of the message identifier (*i.e.*, the nonce) in one of its tabu lists. If the

[1] We do not initialize the counter to zero because in $P^v_{\texttt{TLCN-S}}(Counter)$ we divide by the sum of counters.

Algorithm 1 TLCN−S (sz,update)

Code for every node v

Variables:
  $TL[i]$: array of lists of integer (nonces) defined for all $i \in \mathcal{N}_v$
  $Counter[i]$: array of integer defined for all $i \in \mathcal{N}_v$
  k_v: the symmetric key shared with the sink
End Variables

Initialization
  for all $i \in \mathcal{N}_v$ **do**
    $TL[i] \leftarrow \perp$
    $Counter[i] \leftarrow 1$
  end for
End Initialization

On request to route $\langle Data \rangle$
  generate a random nonce N
  $H \leftarrow$ Hash(N)
  $P \leftarrow (Data||N)$
  $C \leftarrow$ encrypt(P, k_v)
  $S \leftarrow v$
  randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{TLCN-S}}(Counter)$
  send $\langle C, H, S \rangle$ to $next$
  update$(N, TL[next], sz)$
End Request

Upon receiving $\langle C, H, S \rangle$ **from** u
  randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{TLCN-S}}(Counter)$
  send $\langle C, H, S \rangle$ to $next$
End Receive

Upon receiving $\langle N, S \rangle$ **from** u
  if $S == v \wedge \exists z \in \mathcal{N}_v : N \in TL[z]$ **then**
    $Counter[z] \leftarrow Counter[z] + 1$
    remove$(N, TL[z])$
  else if $S \neq v$ **then**
    randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{RWLD}}$
    send $\langle$N,S$\rangle$ to $next$
  end if
End Receive

Code for the sink s

Variables:
  k_i: symmetric keys for each node i in the network
End Variables

Upon receiving $\langle C, H, S \rangle$ **from** u
  $Data||N \leftarrow$ decrypt(C, k_S)
  if Hash(N) $== H$ **then**
    deliver$(Data)$
    send $\langle N, S \rangle$ to u
  end if
End Receive

Upon receiving $\langle N, S \rangle$ **from** u
  randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{RWLD}}$
  send $\langle$N,S$\rangle$ to $next$
End Receive

message identifier is present, then this means that the sink has received this message and so that the link associated with the tabu list leads to a successful delivery. The counter of the link is thus incremented. If the message identifier is not present, then this means that the acknowledgement has been altered or that the message identifier has been removed from the tabu list due to the tabu list size. So, we just stop the forwarding of the acknowledgement.

3.1.4 Multisource Issue

During the design phase, we wanted that all nodes routing a message save the message ID and increment their counters during the passage of the acknowledgement. If we assume that each node increment its counter when a message passes, then it can create the route during this passage. But we noted that this technique created several problems. The message ID must be confidential because otherwise adversaries are easily able to create an acknowledgement and so, persuade legitimate nodes that a lost message is arrived. Moreover, as we are in presence of compromised nodes, the message ID must be only shared between the source node and the sink node. We discuss the way to keep secret the message ID in the next section. Then, we thought to save a hash of the message ID in intermediary nodes instead of the message ID because of the preimage resistance[2] property of the hash ensure the confidentiality of the message ID. But now, our protocol became sensitive to new attacks having as result that the attacker could change the counters as he wants. Consequently, it is able to direct all traffic messages to a chosen node (*e.g.* blackhole).

The only solution to avoid that the adversary manages to manipulate counters of legitimate nodes is to increment counters when nodes receive acknowledgements of their own messages. From a safety point of view, this is justified by the fact that a sensor node has trust in himself only, in presence of insider adversaries. We thus have only the source node that can increment its counters. Since the routing is achieved according to counters, the secure path is detected when nodes have a significant difference between counters. Therefore, a *multi-source* routing is mandatory, *i.e.*, all nodes must send messages to the sink so that the counters stabilize towards routes to the sink.

3.2 "Optimizations"

We propose different kinds of "optimization". We call the next presented methods: optimization, because we hope that they improve the protocol in terms of energy consumption. But since these methods have not been verified, we are not sure of their effectiveness.

In the first optimization, the passage of the message is saved in intermediary nodes to improve the speed of delivery of the acknowledgement to the source node. This optimization is called *reverse chaining* due to the fact that the same chain of nodes is traversed by the message and its acknowledgement. The second optimization is based on a simulated annealing method. Clearly, the idea is that more the counters have been incremented, more the counters are significant.

3.2.1 Reverse chaining

We present the reverse chaining optimization in the algorithm. Here, information is kept in intermediary nodes during the routing of the message. This information should help to reduce the number of hops accomplished by the acknowledgement to reach the source node. The idea is to save the route taken by the message avoiding to pass twice through the same node, *i.e.*, the route saved is an elementary path between the sink and the source. To identify the message in intermediary nodes, we need to save relevant information. We thus save the hash of the nonce and node source ID which are associated with the message. We ensure uniqueness of the message. Moreover, we just add the ID of the sender of this

[2] Given a hash h, it should be difficult to find any message m, such that h=hash(m).

message, *i.e.*, the ID of the neighbor node which has forwarded the message. Then for the acknowledgement routing, the node checks that the message is passed through itself and so apply the reverse chaining.

We keep the same notation. Algorithm 2 gives this version. We describe modifications compared with Algorithm 1. Here, we define two kinds of list. The first list $TL1[i]$ corresponds to the tabu lists of previous algorithm and moreover they have the same behaviors. The second list $TL2$ is associated with each node and contains the hint of the message (*i.e.*, the hash of the nonce and the source node ID) and the oldest sender ID of this message (the sender is the neighbor node which has forwarded the message).

The first modification (Lines 25-28) occurs when a node receives a message to forward. It verifies if the hint is already in the tabu list $TL2$. If it is not the case, then we add the hint associated with the sender in the list and forward the message. If it is the case, then it just forward the message because this means that the message has looped and so it is the oldest sender which makes sense.

After saving the path of the message m in nodes, the second modification (Lines 36-37) wants the acknowledgement of m to follow the same reverse path as m. So, when a node v receives an acknowledgment of m, it forwards the acknowledgement according to the verification of the hint of m in the list $TL2$ of v. Clearly, if the hint is in the list, then it sends the acknowledgment to the sender, else it sends it randomly.

3.2.2 Simulated annealing

Simulated annealing is a generic probabilistic metaheuristic for the global optimization problem published in [KGV83]. This method is an adaptation of the Metropolis-Hastings algorithm [MRR+53] [Has70]. We use simulated annealing to give a chance to all counters to be incremented.

Application to `TLCN-S` Each source node u sends a message to the sink. To choose the next hop, it uses the probability law $P^u_{\texttt{TLCN-S}}(Counter)$. But at the beginning, counters have no meaning because the node has not received enough acknowledgments of messages. So we assume a probability p that depends on the number of acknowledgments received ($nbAck$) and the number of counters in the node (equivalent to the node degree δ_u). Consequently, the source node chooses the next hop using the probability law $P^u_{\texttt{RWLD}}$ with the probability p and it chooses the next hop using the probability law $P^u_{\texttt{TLCN-S}}(Counter)$ with the probability $1-p$. The exponential is used to express p due to the Metropolis rule. So we have the basic formula:

$$p = \exp^{-\frac{nbAck}{\delta_u}}$$

This formula is totally arbitrary, so we need many experiments to find the good settings. For example, the probability p can depend on: constant factors, the size and degree of the graph, the distance to the sink of the node (implying that each node does not have the same probability law), *etc.*

3.2.3 Conclusion

We presented a new secure routing protocol designed to be resilient against the stronger adversaries. It offers an interesting technique to avoid malicious nodes. We verify this in Chapter 5 by simulation, after proving the classic security properties. Then, we proposed two avenues of research to save energy. The reverse chaining is supposed to improve the hitting time of acknowledgements, but we remark that it increases the code size at nodes. The simulated annealing is supposed to improve the hitting time of messages, but it must be precisely empirically adjusted.

Algorithm 2 TLCN−S ($sz1$,$sz2$,update)

Code for every node v

Variables:
  $TL1[i]$: array of lists of nonces defined for all $i \in \mathcal{N}_v$
  $TL2$: list of triplet defined for the node v
  $Counter[i]$: array of integer defined for all $i \in \mathcal{N}_v$
  k_v: the symmetric key shared with the sink
End Variables

Initialization
  $TL2 \leftarrow \perp$
  for all $i \in \mathcal{N}_v$ **do**
    $TL1[i] \leftarrow \perp$
    $Counter[i] \leftarrow 1$
  end for
End Initialization

On request to route $\langle Data \rangle$
  generate a random nonce N
  $H \leftarrow$ hash(N)
  $P \leftarrow (Data||N)$
  $C \leftarrow$ encrypt(P, k_v)
  $S \leftarrow v$
  randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{TLCN-S}}(Counter)$
  send $\langle C, H, S \rangle$ to $next$
  update$(N, TL1[next], sz1)$
End Request

Upon receiving $\langle C, H, S \rangle$ **from** u
  if $(H, S) \notin TL2$ **then**
    $sender \leftarrow u$
    update$((H, S, sender), TL2, sz2)$
  end if
  randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{TLCN-S}}(Counter)$
  send $\langle C, H, S \rangle$ to $next$
End Receive

Upon receiving $\langle N, S \rangle$ **from** u
  if $S == v \wedge \exists z \in \mathcal{N}_v : N \in TL1[z]$ **then**
    $Counter[z] \leftarrow Counter[z] + 1$
    remove$(N, TL1[z])$
  else if $S \neq v \wedge \exists (H, S, sender) \in TL2 : H ==$Hash($N$)$\wedge S == D$ **then**
    send ⟨N,S⟩ to $sender$
  else if $S \neq v$ **then**
    randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{RWLD}}$
    send ⟨N,S⟩ to $next$
  end if
End Receive

Code for the sink s

Variables:
  k_i: symmetric keys for each node i in the network
End Variables

Upon receiving $\langle C, H, S \rangle$ **from** u
  $Data||N \leftarrow$ decrypt(C, k_S)
  if Hash(N) $== H$ **then**
    deliver$(Data)$
    send $\langle N, S \rangle$ to u
  end if
End Receive

Upon receiving $\langle N, S \rangle$ **from** u
  randomly select $next \in \mathcal{N}_v$ using probability law $P^v_{\texttt{RWLD}}$
  send ⟨N,S⟩ to $next$
End Receive

Chapter 4

A Provably Secure Routing Protocol

We prove security properties of TLCN−S. In first, we describe the protocol to clearly prove the data confidentiality and message unforgeability that imply data integrity and authenticity. For this proof, we use the random oracle model. First, we recall some standard notation and definition based on [BDJR97] and [Sho04]. We use the classic approach of security in cryptography, *i.e.*, a scheme is declared provably secure if there is some polynomial-time reduction from a hard problem to it. The security depends on the resources of the adversary (Definition 1).

Definition 1 (Adversary)**.** *An adversary $\mathcal{A}$ is a probabilistic polynomial time Turing machine.*

Definition 2 (Negligible)**.** *We call a function $\mu : \mathbb{N} \to \mathbb{R}^+$ negligible if for every positive polynomial p there exists an integer N such that:*

$$\forall n > N \, , \, \mu(n) < \frac{1}{p(n)}$$

For a set X, we write $x \xleftarrow{R} X$ meaning that an element sampled from the uniform distribution on X is assigned to the variable x. If A is a probabilistic algorithm, then the notation $y \xleftarrow{R} A(x)$ denotes the action of running A on input x and assigning the output to y.

In our scheme, the security is based on the security of a block cipher which is a classic symmetric encryption primitive. The estimated cryptanalytic strength of specific block ciphers gives us estimates for values of input sizes and key sizes. These ciphers are still widely studied. For example, nowadays we consider DES [How87] as no secure cipher and we usually use AES [DR02]. The security of a block cipher is based on the notion of a PseudoRandom Function (PRF) family F defined such that a function F_k associated with the key k maps l-bits to L-bits with k from the key space of F, noted $Keys(F)$. In particular, if $l = L$ and F_k is a permutation for all k, then we speak of pseudorandom permutation.

Let F_k be a family of keyed functions that map l-bits to L-bits. Let $Rand$ be the set of all functions from l-bits to L-bits. Consider an oracle algorithm, known as a distinguisher D, that attempts to distinguish between the case where its oracle $\mathcal{O}$ is chosen randomly from F_k and the case where $\mathcal{O}$ is chosen randomly from $Rand$. Pseudorandom function family has the property that the any poly-time distinguisher can differentiate a random function in F_k, from a random function in $Rand$.

Definition 3 (Security of PRF families [BKR94])**.** *A keyed function family F_k is said to be a secure PRF family if for any distinguisher D, $ADV^{prf}(D)$ is negligible, where $ADV^{prf}(D)$ is define as:*

$$ADV^{prf}(D) = |Pr[\mathcal{O} \xleftarrow{R} F_k : D^{\mathcal{O}(.)} = 1] - Pr[\mathcal{O} \xleftarrow{R} Rand : D^{\mathcal{O}(.)} = 1]|$$

Let $\mathcal{K} = \{0,1\}^m$ be the space keys. We use a block cipher as a building block for our protocol. The signature of the block cipher is the following: encrypt/decrypt : $\{0,1\}^n \times \mathcal{K} \to \{0,1\}^n$ with n a fixed

integer. It must satisfy decrypt(encrypt$(x, k), k) = x$. We use a nonce which is a number used only once, it is unpredictable and generated randomly from a set $\{0,1\}^l$ according to an uniform distribution. We use a cryptographically secure pseudo-random number generator to achieve these goals. We also use a ϵ-Universal family of hash functions (defined below) that takes an arbitrary input (arbitrary size) and returns an output of fixed size.

Definition 4 (ϵ-Universal family of hash functions [CW79] [WC81])**.** *Let l_1 and l_2 be positive integers such that $l_1 > l_2$. Let K be a space finite of bits. Let $\mathcal{H} = \{H_k\}_{k \in K}$ be a family of functions indexed by k, so that for every $k \in K$, H_k is a function from $\{0,1\}^{l_1}$ to $\{0,1\}^{l_2}$. Then $\mathcal{H}$ is an ϵ-Universal family of hash functions, if and only if for all x, $x' \in \{0,1\}^{l_1}$ with $x \neq x'$, we have*

$$Pr[k \xleftarrow{R} K : H_k(x) = H_k(x')] \leq \epsilon$$

where ϵ is negligible.

4.1 Protocol Description

We present our protocol that ensures security properties. It consists of 3 algorithms (Init, Gen, Verif) that, respectively, initialize keys for users, generate an encrypted message, and verify the validity of a message. The "Gen" algorithm uses an encryption oracle $\mathcal{O}$, knowing that in our protocol, it is the encrypt function of the block cipher. Formally:

- Init($\mathcal{K}$): Outputs symmetric keys $k \in \mathcal{K}$. Before the deployment of the WSN, these keys are pre-distributed in sensor nodes such that each node store one single key and the sink stores all these keys. We note k_{QP}, the unique key shared between a node Q and the sink P. This should also pick a key for the hash function.
- Gen$^{\mathcal{O}}(Data, Q)$: Outputs a message $m = (C, H, Q)$ where Q is the node ID of the sender (source) which runs the "Gen" algorithm:
  - Generate a random nonce N
  - $C = \mathcal{O}(Data||N, k_{QP})$
  - $H =$ hash(N)

  Once the message m was generated by Q, then m is sent to the recipient, *i.e.*, the sink node P which shares the symmetric key.
- Verif(m): Outputs a single bit indicating validity (1) or invalidity (0) of the message $m = (C, H, Q)$, *i.e.*, the sink node P runs the "Verif" algorithm on the message received:
  - P chooses the key k_{QP} associated with the node Q
  - P computes $Data||N =$ decrypt(C, k_{QP})
  - P outputs 1 if $H =$ Hash(N) and outputs 0 otherwise

4.2 Confidentiality

We define data confidentiality through the following game in which an adversary $\mathcal{A}$ tries to "break" the protocol. The game is based on the Find-then-Guess game of [BDJR97]. An adversary $\mathcal{A}$ is a pair of poly-time oracle algorithms $(\mathcal{A}_1, \mathcal{A}_2)$, each having access to an oracle $\mathcal{O}$ which comes from F_k, a keyed function family representing the block cipher. We remark that the nonce N^* is given to the

adversary $\mathcal{A}_2$ because in our protocol, this nonce is revealed by the sink when the message is received. Let $b \in \{0,1\}$ chosen randomly, for all source node Q and for a unique sink node P, we consider the following experiment
FG$^b(\mathcal{A})$:

- $k_{QP} \xleftarrow{R}$ Init($\mathcal{K}$)
- $(Data_0^*, Data_1^*, s) \xleftarrow{R} \mathcal{A}_1^{Gen^{\mathcal{O}}(.,Q)}$
- $m^* \leftarrow$ Gen$^{\mathcal{O}}(Data_b, Q)$
- $b' \xleftarrow{R} \mathcal{A}_2^{Gen^{\mathcal{O}}(.,Q)}(m^*, N^*, s)$

We add the variable s to underline that $\mathcal{A}$ keeps an internal state throughout the whole experiment. This game tests the ability of the adversary to guess which data has been encrypted when the ciphertext is given. We speak about of security by indistinguishability, *i.e.*, the adversary cannot make the difference between encryption of two plaintexts. This definition formalizes the idea that an adversary should not be able to obtain a single bit of information about an encrypted data. We define the advantage of an adversary $\mathcal{A}$ in the experiment FG as follows:

$$ADV_{FG}(\mathcal{A}) = |2Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} FG^b(\mathcal{A}) : b' = b] - 1|.$$

Definition 5 (Data confidentiality)**.** *Data confidentiality is ensured if and only if, for every adversary $\mathcal{A}$, the function $ADV_{FG}(\mathcal{A})$ is negligible.*

We express Theorem 1 that provides data confidentiality using a block cipher as a secure PRF family (cryptanalysis gives us bounds) as follows:

Theorem 1. *If the block cipher is a secure PRF family (Definition 3) then for any adversary $\mathcal{A}$ that makes q queries and a nonce $N \in \{0,1\}^n$, $ADV_{FG}(\mathcal{A}) \leq 2ADV^{prf}(D) + \frac{q}{2^n}$.*

The proof is achieved by construction of successive experiments allowing us to bound the initial advantage of FG experiment. We first introduce an experiment (Exp_1) that uses a truly random function in the encryption oracle. The two attack games (FG and Exp_1) cannot be distinguished by an adversary due to Definition 3 on PRF. Then, we find an upper bound on Exp_1 using an experiment Exp_2 which is not correlated with the encryption function due to the fact that the searched ciphertext is randomly generated. We conclude showing that the only way to obtain some data information in Exp_2 is to look for a collision between two ciphertexts.

We introduce the series of experiments recalling the FG experiment that we study. In the following, experiments are defined for all source nodes Q, for a unique sink node P, for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and with $b \in \{0,1\}$ chosen randomly.

FG$^b(\mathcal{A})$:

- $k_{QP} \xleftarrow{R}$ Init($\mathcal{K}$)
- $(Data_0^*, Data_1^*, s) \xleftarrow{R} \mathcal{A}_1^{Gen^{\mathcal{O}}(.,Q)}$
- $m^* \leftarrow$ Gen$^{\mathcal{O}}(Data_b, Q)$
- $b' \xleftarrow{R} \mathcal{A}_2^{Gen^{\mathcal{O}}(.,Q)}(m^*, N^*, s)$

We modify the experiment FG in the experiment Exp_1 using a random function instead of a block cipher. In the "Gen" operation, we now have $\mathcal{O}$ coming from $Rand$ where $Rand$ is the set of all functions from $\{0,1\}^l$ to $\{0,1\}^L$.

$Exp_1^b(\mathcal{A})$:

- $\mathcal{O} \xleftarrow{R} Rand$
- $(Data_0^*, Data_1^*, s) \xleftarrow{R} \mathcal{A}_1^{Gen^{\mathcal{O}}(.,Q)}$
- $m^* \leftarrow \text{Gen}^{\mathcal{O}}(Data_b^*, Q)$
- $b' \xleftarrow{R} \mathcal{A}_2^{Gen^{\mathcal{O}}(.,Q)}(m^*, N^*, s)$

Next, we want to compare the Exp_1 experiment with the experiment Exp_2 which ignores the challenge step to construct a message.

$Exp_2^b(\mathcal{A})$:

- $\mathcal{O} \xleftarrow{R} Rand$
- $(Data_0^*, Data_1^*, s) \xleftarrow{R} \mathcal{A}_1^{Gen^{\mathcal{O}}(.,Q)}$
- $m = (R, H, Q)$ where $R \xleftarrow{R} \{0,1\}^n$
- $b' \xleftarrow{R} \mathcal{A}_2^{Gen^{\mathcal{O}}(.,Q)}(m, s)$

We now introduce the lemmas that we will use in the proof of Theorem 1. The $\mathcal{A}$'s advantage in Exp_1 is defined as $ADV_{Exp_1}(\mathcal{A}) = |2Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} Exp_1^b(\mathcal{A}) : b' = b] - 1|$. We show that if F_k is a family of PRF, then no adversary $\mathcal{A}$ can distinguish the real attack game (FG) from Exp_1.

Lemma 1. *For any adversary $\mathcal{A}$, there exists a distinguisher D_1 such that $|ADV_{FG}(\mathcal{A}) - ADV_{Exp_1}(\mathcal{A})| \leq 2ADV^{prf}(D_1)$*

Proof. We construct a distinguisher $D_1^{\mathcal{O}}$ that tries to determine if its oracle $\mathcal{O}$ comes from the function family F_k representing the block cipher or comes from $Rand$. $D_1^{\mathcal{O}}$ will use $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and it will simulate $Gen_{k_{QP}}^{\mathcal{O}}(., Q)$ using oracle $\mathcal{O}$.
Algorithm $D_1^{\mathcal{O}}$:

- When $\mathcal{A}_1$ asks oracle $Gen^{\mathcal{O}}(., Q)$ queries $q_i = (Data_i, Q)$, D_1 answers to $\mathcal{A}_1$, $m_i = (C_i, H_i, Q)$ such that $C_i = \mathcal{O}(Data_i || N_i)$
- $\mathcal{A}_1$ outputs $(Data_0^*, Data_1^*, s)$
- D_1 samples $b \in \{0,1\}$
- D_1 computes $m^* = (C^*, H^*, Q)$ such that $C^* = \mathcal{O}(Data_b^* || N^*)$
- It gives (m^*, N^*, s) to $\mathcal{A}_2$
- When $\mathcal{A}_2$ asks oracle $Gen^{\mathcal{O}}(., Q)$ queries $q_i = (Data_i, Q)$, D_1 answers to $\mathcal{A}_2$, $m_i = (C_i, H_i, Q)$ such that $C_i = \mathcal{O}(Data_i || N_i)$
- $\mathcal{A}_2$ outputs b'
- If $b = b'$ then D_1 returns 1 else D_1 returns 0

So we have constructed the distinguisher $D^{\mathcal{O}}$ such that if $\mathcal{O} \xleftarrow{R} F_k$ then $\mathcal{A}$'s environment is as in game FG, and if $\mathcal{O} \xleftarrow{R} Rand$ then $\mathcal{A}$'s environment is as in game Exp_1. Thus by construction:

$$\begin{aligned} ADV^{prf}(D_1) &= | \, Pr[\mathcal{O} \xleftarrow{R} F_k : D_1^{\mathcal{O}} = 1] - Pr[\mathcal{O} \xleftarrow{R} Rand : D_1^{\mathcal{O}} = 1] \, | \\ &= | \, Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} FG^b(\mathcal{A}) : b' = b] - Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} Exp_1^b(\mathcal{A}) : b' = b] \, | \end{aligned}$$

Consequently, we can bound probabilities of experiments and obtain the inequality of Lemma 1:

$$
\begin{aligned}
&|ADV_{FG}(\mathcal{A}) - ADV_{Exp_1}(\mathcal{A})| \\
&= | \, |2Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} FG^b(\mathcal{A}) : b' = b] - 1| - |2Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} Exp_1^b(\mathcal{A}) : b' = b] - 1| \, | \\
&\leq | \, 2Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} FG^b(\mathcal{A}) : b' = b] - 1 - (2Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} Exp_1^b(\mathcal{A}) : b' = b] - 1) \, | \\
&\leq | \, 2(Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} FG^b(\mathcal{A}) : b' = b] - Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} Exp_1^b(\mathcal{A}) : b' = b]) \, | \\
&\leq 2ADV^{prf}(D_1)
\end{aligned}
$$

□

As the experiment Exp_1 uses a random function, we are able to explictly bound its advantage.

Lemma 2. *For any adversary $\mathcal{A}$ that makes q oracle queries, we have $ADV_{Exp_1^b(\mathcal{A})}(\mathcal{A}) \leq \frac{q}{2^n}$ with n the bit size of the nonce.*

Proof. We define $\mathcal{A}$'s advantage in Exp_2 as $ADV_{Exp_2^b}(\mathcal{A}) = |2Pr[b' \xleftarrow{R} Exp_2^b(\mathcal{A}) : b' = b] - 1|$. We note that the adversary has no information because R is a random number independent from b. Hence:

$$Pr[b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} Exp_2^b(\mathcal{A}) : b' = b] = \frac{1}{2}$$

Thus:

$$
\begin{aligned}
ADV_{Exp_2^b}(\mathcal{A}) &= |2Pr[b' \xleftarrow{R} Exp_2^b(\mathcal{A}) : b' = b] - 1| \\
&= 2 \times \frac{1}{2} - 1 \\
&= 0
\end{aligned}
$$

We also note that $\mathcal{A}$ can only distinguish Exp_1 from Exp_2 if the nonce N^* used in the challenge is reused in another "Gen" query. Otherwise, $f(Data^*||N^*)$ has been queried only once and thus "looks random" since f is a random function.

Let q be the total number of queries done by $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the Exp_1 experiment. Formally, we have:

$$
\begin{aligned}
| \, ADV_{Exp_1}(\mathcal{A}) - ADV_{Exp_2}(\mathcal{A}) \, | &= Pr[\text{collision of two ciphertexts on } q \text{ queries}] \\
&\leq Pr[\text{collision of } N^* \text{ on } q \text{ queries}]
\end{aligned}
$$

Consequently, we look for the probability that the nonce of a query collides with the challenge nonce. We note A_i the following event: the i-th nonce generated by the oracle is the same that the challenge nonce. We assume that the nonce comes from the space $\{0,1\}^n$ and it is chosen randomly with an uniform law. Therefore, the probability that two nonces are equal, is $\frac{1}{2^n}$ and so $Pr[A_i] = \frac{1}{2^n}$. Finally, we can bound the previous probability on collisions in the experiment:

$$
\begin{aligned}
Pr[\text{collision of } N^* \text{ on } q \text{ queries}] &= Pr[\bigcup_{i=1}^{q} A_i] \\
&\leq \sum_{i=1}^{q} Pr[A_i] \\
&\leq \sum_{i=1}^{q} \frac{1}{2^n} \\
&\leq \frac{q}{2^n}
\end{aligned}
$$

Since $ADV_{Exp_2}(\mathcal{A}) = 0$, we have $ADV_{Exp_1}(\mathcal{A}) \leq \frac{q}{2^n}$. □

Finally, we use previous lemmas to bound the advantage of FG experiment and obtain the proof of data confidentiality.

Proof of Theorem 1. Due to the Lemma 1, we have:

$$
\begin{aligned}
&|ADV_{FG}(\mathcal{A}) - ADV_{Exp_1}(\mathcal{A})| \leq 2ADV^{prf}(D_1) \\
&\Rightarrow ADV_{FG}(\mathcal{A}) \leq 2ADV^{prf}(D_1) + ADV_{Exp_1}(\mathcal{A}) \\
&\Rightarrow ADV_{FG}(\mathcal{A}) \leq 2ADV^{prf}(D_1) + \frac{q}{2^n} \text{ according to the Lemma 2}
\end{aligned}
$$

□

4.3 Unforgeability

The second security property is unforgeability, *i.e.*, it is computationally unfeasible for an adversary (not having the secret key k) to be able to create a valid message which is "new". Here, "new" means that the message has not already been created by a party Q. Let $\mathcal{A}$ an adversary (forger) who has access to an oracle for $Gen^{\mathcal{O}}(.,Q)$ with $\mathcal{O}$ representing the block cipher encryption in our scheme, for all source node Q and for a unique sink node P, we consider the following experiment noted
Forge($\mathcal{A}$):

- $k_{QP} \xleftarrow{R}$ Init($\mathcal{K}$)
- $m^* \xleftarrow{R} \mathcal{A}^{Gen^{\mathcal{O}}}(.,Q)$
- If Verif$(m^*) = 1$ and m^* was not a response from an oracle query of $\mathcal{A}$ then return 1 else return 0

We use this experiment to evaluate the probability that $\mathcal{A}$ succeeds in creating a new valid message.

Definition 6 (Message Unforgeability [BKR94])**.** *The messages in our protocol are existentially unforgeable if and only if for any poly-time adversary $\mathcal{A}$ Pr[Forge($\mathcal{A}$) $= 1$] is negligible.*

We use the notion of ϵ-Universal hash function (see Definition 4) which we will need to prove message unforgeability of our protocol. Definition 4 states that for any two distinct inputs the probability for a collision is at most ϵ. In the following, we denote the ϵ of Definition 4 by ϵ_{uh}. We can also remark that there exists many solutions to construct an ϵ_{uh}-Universal family of hash functions in the literature.

The probability of the Forge experiment depends on the security of the block cipher and the ϵ-universal property of the hash function. For the block cipher, if the adversary obtains information on the key, then it improves its chances to create a valid ciphertext and so a valid message because it will know the nonce and it will be easily able to compute the valid hash. For the hash function, if the adversary produces a lot of random message, then it improves its chances to create a valid hash and so a valid message. We recall that N is a random nonce which comes from $\{0,1\}^n$.

Theorem 2. *The block cipher is a secure PRF family (Definition 3) and the hash function belongs to the ϵ_{uh}-universal family (Definition 4), then for any adversary $\mathcal{A}$, there exists a distinguisher D_2 such that $Pr[Forge(\mathcal{A}) = 1] \leq ADV^{prf}(D_2) + \frac{1}{2^n} + \epsilon_{uh}$.*

We construct a distinguisher that generates messages for the forger using its encryption oracle. The forger's environnement is similar to the forge experiment when the oracle of D comes from F_k. So now to bound the probability that the forge experiment returns a valid message, we need to know the probability that the forger returns 1 when the oracle of D comes from $Rand$. As the message validity is verified using the hash of the nonce, the previous probability depends on the collision probability of the hash function. We formally explain this reasoning in the following proof.

Proof of Theorem 2. We design a distinguisher D_2 which uses either a function comes from F_k (block cipher family) or a function comes from $Rand$ as encryption oracle $\mathcal{O}$. D_2 will run the adversary of the forge experiment providing answers of oracle queries of $\mathcal{A}$.
Algorithm of $D_2^{\mathcal{O}}$:

- $\mathcal{A}$ asks oracle $Gen^{\mathcal{O}}(.,Q)$ queries $q_i = (Data_i, Q)$, D_2 answers to $\mathcal{A}$, $m_i = (C_i, H_i, Q)$ such that $C_i = \mathcal{O}(Data_i||N_i)$
- $\mathcal{A}$ outputs $m^* = (C^*, H^*, Q)$
- If Verif(m^*)=1 then return 1 else return 0

We have expressed the distinguisher $D_2^{\mathcal{O}}$ such as the distinguisher of the Definition 3:

$$\begin{aligned} ADV^{prf}(D_2) &=\mid Pr[\mathcal{O} \xleftarrow{R} F_k : D_2^{\mathcal{O}} = 1] - Pr[\mathcal{O} \xleftarrow{R} Rand : D_2^{\mathcal{O}} = 1] \mid \\ &=\mid Pr[Forge(\mathcal{A} = 1)] - Pr[\mathcal{O} \xleftarrow{R} Rand : D_2^{\mathcal{O}} = 1] \mid \end{aligned}$$

Because if $\mathcal{O} \xleftarrow{R} F_k$ then D_2 provides $\mathcal{A}$ with environment as in Forge. The distinguisher $D_2^{\mathcal{O}}$ outputs 1 when $\mathcal{O}$ comes from $Rand$ only when hashes are equals, *i.e.*, when $H^* = Hash(N^*)$. Since m^* is "new" and f is random, $\mathcal{A}$ cannot know $f^{-1}(C^*)$ that is equal to $(Data^*||N^*)$. So it cannot also know N^*. Thus the H^* is produced such that either $\exists\ N\ /\ H^* = Hash(N)$ or $\nexists\ N\ /\ H^* = Hash(N)$. We denote the event "the hash of the nonce decrypted (N^*) is equal to the hash of the message (H^*) computed from a random nonce (N_r)" by $Hash(N^*) = Hash(N_r)$. This event only occurs when we take the same input or when we have a collision. As the nonce N is drawn uniformly in the set $\{0,1\}^n$, the probability that hash function takes the same input is $\frac{1}{2^n}$. Moreover the probability of a collision is bounded by definition. Hence the following inequality;

$$\begin{aligned} & Pr[\mathcal{O} \xleftarrow{R} Rand : F^{\mathcal{O}} = 1] \\ =& Pr[H^* = Hash(N^*)] \\ =& Pr[H^* = Hash(N^*) \wedge \exists\ N\ /\ Hash(N) = H^*] + Pr[H^* = Hash(N^*) \wedge \nexists\ N\ /\ Hash(N) = H^*] \\ =& Pr[H^* = Hash(N^*) \wedge \exists\ N\ /\ Hash(N) = H^*] \\ =& Pr[\exists\ N\ /\ Hash(N) = Hash(N^*)] \\ =& Pr[Hash(N_r) = Hash(N^*)] \\ =& Pr[N^* = N_r\ \cup\ (Hash(N^*) = Hash(N_r)\ \cap\ N^* \neq N_r)] \\ =& Pr[N^* = N_r] + Pr[Hash(N^*) = Hash(N_r)\ \cap\ N^* \neq N_r] \\ =& \frac{1}{2^n} + \epsilon_{uh} \text{ according to Definition 4} \end{aligned}$$

Finally, we thus have:

$$\begin{aligned} & ADV^{prf}(D_2) =\mid Pr[Forge(\mathcal{A} = 1)] - Pr[\mathcal{O} \xleftarrow{R} Rand : D_2^{\mathcal{O}} = 1] \mid \\ & \Rightarrow Pr[Forge(\mathcal{A} = 1)] \leq ADV^{prf}(D_2) + Pr[\mathcal{O} \xleftarrow{R} Rand : D_2^{\mathcal{O}} = 1] \\ & \Rightarrow Pr[Forge(\mathcal{A} = 1)] \leq ADV^{prf}(D_2) + \frac{1}{2^n} + \epsilon_{uh} \end{aligned}$$

□

Features of unforgeability

Integrity: Message integrity is ensured when the receiver is sure that the received message has not been altered by an adversary. In WSN, we are interested in data integrity which is included in message integrity. We propose to verify that the message sent by the source is the same as the message received by the sink. For any adversary $\mathcal{A}$, for any $Data$, for all source nodes Q and for the unique sink P we define the following experiment noted
INT$(\mathcal{A}, Data, Q)$:

- $k_{QP} \leftarrow$ Init($\mathcal{K}$)
- $m = (C, H, Q) \leftarrow$ Gen$(Data, Q)$
- $\tilde{m} = (\tilde{C}, \tilde{H}, \tilde{Q}) \xleftarrow{R} \mathcal{A}(m)$
- If $m \neq \tilde{m} \wedge$ Verif$(\tilde{m}) = 1$ then return 1 else return 0

This experiment shows that when $\mathcal{A}$ wins (*i.e.*INT$(\mathcal{A}, Data, Q)$ returns 1) then the received message may have been altered.

Definition 7 (Message integrity)**.** *Message integrity is ensured if and only if, for any adversary $\mathcal{A}$, any $Data$ and for all Q, the function Pr[INT$(\mathcal{A}, Data, Q)$=1] is negligible.*

We show that message unforgeability implies data integrity because we remark that if an adversary can not create a valid message, then it cannot alter a message without the sink noticing.

Lemma 3. *If for any adversary $\mathcal{A}$, Pr[Forge($\mathcal{A}$) $= 1$] is negligible, then for any $Data$ and for all Q, Pr[INT$(\mathcal{A}, Data, Q)$=1] is negligible.*

Proof. We assume that for an adversary $\mathcal{A}_1$, $\exists\, Data, Q$ such that the function Pr[INT$(\mathcal{A}_1, Data, Q)$=1] is not negligible. So, we build an adversary $\mathcal{A}_2$ which will use $\mathcal{A}_1$ to forge a message.
Algo $\mathcal{A}_2$:

- Call oracle on input $(Data, Q)$ and it returns m^*
- Run $\mathcal{A}_1$ on m^* to obtain $\tilde{m}$
- Return $\tilde{m}$

Inputs of oracle are exactly the same as inputs of the INT experiment. Moreover, the output of $\mathcal{A}_2$ is also the same as the output of $\mathcal{A}_1$. So we have Pr[Forge($\mathcal{A}_2$) $= 1$]=Pr[INT$(\mathcal{A}_1, Data, Q)$=1] and according to the assumption, we obtain that Pr[Forge($\mathcal{A}_2$) $= 1$] is not negligible. Thus by contradiction, the lemma is proved. □

Authenticity: Message authenticity is ensured when the receiver has verified that the received message was really created by the claimed sender. Here, we want to verify that the received message has been created by the source node ID present in the message. For any adversary $\mathcal{A}$, for any $Data$, for all source nodes Q and for the unique sink P, we describe the following experiment:
AUT$(\mathcal{A}, Data, Q)$:

- $k_{QP} \leftarrow$ Init($\mathcal{K}$)
- $m = (C, H, Q) \leftarrow$ Gen$(Data, Q)$
- $\tilde{m} = (C, H, \tilde{Q}) \xleftarrow{R} \mathcal{A}(m)$
- If $m \neq \tilde{m} \wedge$ Verif$(\tilde{m}) = 1$ then return 1 else return 0

This experiment returns 1 when a message with a false sender is validated.

Definition 8 (Message Authenticity)**.** *Message Authenticity is ensured if and only if, for any adversary $\mathcal{A}$, any $Data$ and for all Q, the function Pr[AUT$(\mathcal{A}, Data, Q)$=1] is negligible.*

Informally, the decryption of the cipher in the message is computed with the wrong key, so the probability that the hash of the message and the hash of the nonce found are equal, is negligible. This probability can be associated with the probability that an adversary forge a new message because the source node ID is inside the message.

Lemma 4. *If for any adversary, Pr[Forge$(\mathcal{A}) = 1$] is negligible, then for any $Data$ and for all Q, Pr[AUT$(\mathcal{A}, Data, Q)$=1] is negligible.*

Proof. We assume that in the INT experiment, the adversary only changes Q and do not modify C and Q. Then the two experiments (INT and AUT) are similar and so we can use the same proof that Lemma 3. Consequently, we obtain $Pr[AUT(\mathcal{A}, Data, Q) = 1] \leq Pr[Forge(\mathcal{A}) = 1]$. □

Chapter 5

Performances

We evaluated the relative performance of the protocol TLCN−S in terms of hitting time and delivery rate in presence of several attacks. Finally, we introduce a new attack and we propose countermeasures.

5.1 Settings

We ran our simulations using Sinalgo[1], an event-driven simulator for wireless networks written in Java [Gro08]. In this simulator, nodes are deployed on a square plane using a uniform random distribution. Nodes are equipped with radio. Two nodes u and v can communicate together if and only if their euclidean distance is at most r, where r is the transmission range. In other words, Sinalgo models networks as *Unit Disk Graphs* (UDG). Note that, UDGs are commonly used in the literature to model WSNs.

We consider physical and MAC layers to be ideal: there are neither interference nor collision.[2] Moreover, nodes are motionless. However, node executions and message transmissions are concurrent and asynchronous. More precisely, the transmission time of every message is chosen uniformly at random in $[1..10]$. Every node checks if it received messages every 30 time units. Hence, every message is treated at most 40 time units after its sending.

We only considered connected networks where all nodes are sources except one which is the sink. Every source node generates messages to be routed. The time between each data generation is chosen uniformly at random in $[400..600]$.[3]

In the following, we study the performance of our protocols according to the two possible update policies (FIFO and Random), different probability laws for data messages and acknowledgements (RW and RWLD) and several sizes of tabu list. Of course, for TLCN−S, the greater the bound on the size of tabu lists, the better the performance is. The two update policies give similar results. In contrast, probability laws have an important impact in favor of RWLD. This is why, we choose to present results obtained only with a FIFO policy and the RWLD probability law, moreover this law represents a defense against the influence of malicious node degrees. We also arbitrarly choose a tabu list size of 15 (as in [ADLP10]).

5.2 Hitting Time

We evaluated the hitting time as the average number of hops to route any message from its source to the sink. We considered graphs of size 200 having the same average degree, here 8. As TLCN in [ADLP10],

[1] http://disco.ethz.ch/projects/sinalgo/

[2] Such ideal layers can be implemented using the alternating bit protocol [BSW69].

[3] We choose the average generation time to be highly greater than the average transmission time to avoid congestion.

the probability law to choose the next hop of TLCN−S depends on counters, so these protocols use a learning mechanism. The simulation is achieved without an adversary because we want to compare the hitting time of our secure protocol with that of other random walk protocols. We thus present in Figure 5.1 the evolution of the hitting time in function of the number of sent messages in long-term execution. This experiment confirm that in RW, RWLD and TLM, each message is routed independently and in TLCN and TLCN−S, a probabilistic route is built from counters. We can conclude that TLCN−S is better than RW and RWLD and seems to stabilize. The fact that TLCN−S stabilizes around the TLM (of size 15) hitting time in this experiment is just an experimental coincidence. The hitting time of TLCN is drastically smaller than TLCN−S. This is due to the fact that TLCN has been designed to minimize this hitting time whereas TLCN−S has been designed to increase the delivery rate.

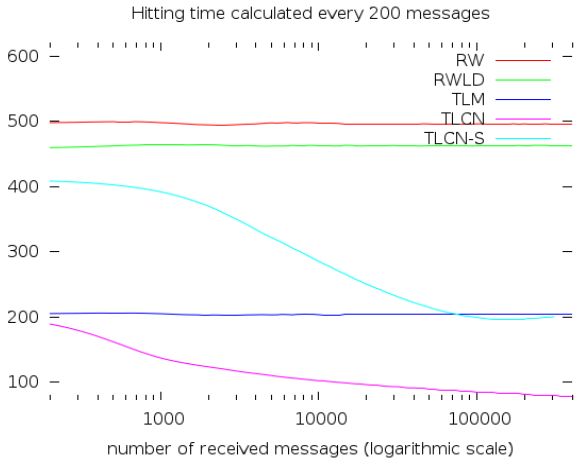

Figure 5.1: Hitting Time Evolution

In Figure 5.2, we observe that wormhole has no influence on routing protocols. On the contrary, as the average degree increases then hitting time is improved. The hitting time of TLCN−S is higher than other protocols, this is due to the fact that we have computed the average (hitting time) on only 100 messages per sources, and so TLCN−S is not yet stabilized.

5.3 Delivery Rate

We analysed random walk routings according to the delivery rate, *i.e.*, the number of messages received by the sink divided by the number of messages emitted by sources. This metric is essential to analyse security of routing protocols, it shows the network availability. When there is no attacker as for experiments on hitting time, the delivery rate is 1. In Figure 5.3, we evaluate the delivery rate in presence of x attackers randomly distributed on graphs of size 200 and average degree 8. Each node sends 400 messages to the sink but compromised nodes are blackhole, *i.e.*, they do not forward messages that they receive.

We observe that TLCN−S is widely better than other random walk routings. We can explain these results for no secure random walk protocols as follows: if the sink and malicious nodes are randomly chosen, *i.e.*, their degree and their distance to the sink are random, then a message has the same probability of hitting them in first. This is due to the fact that experiments are achievied on many random graphs, so the average of hitting time smooth results, degree effects and other curious effects. Consequently, we

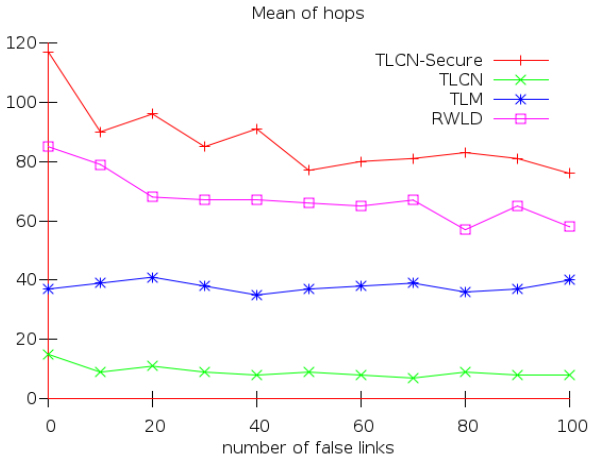

Figure 5.2: Hitting Time with wormhole attack

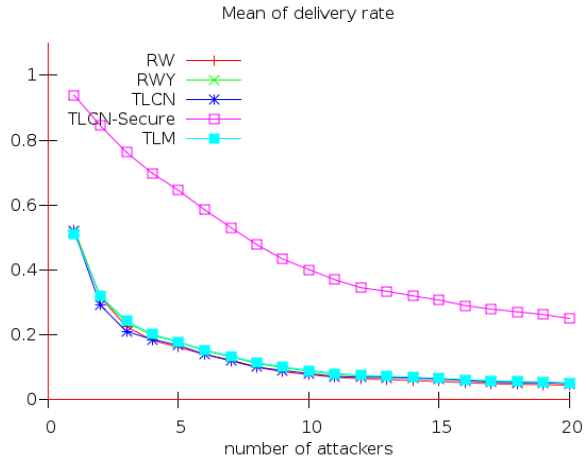

Figure 5.3: Delivery rate with blackhole nodes

conjecture that the average rate of delivery random walk routings is $\frac{1}{\mathcal{N}_i+1}$ with $\mathcal{N}_i$, the number of malicious nodes. Moreover, when the network is composed of $\mathcal{N}_s$ sinks, then we can generalize by $\frac{1}{\mathcal{N}_i+\mathcal{N}_s}$. In `TLCN-S`, the mechanism put in place is efficient. Nodes detect the safe paths and choose to send message to this direction. We clearly understand the protocol behavior in Figure 5.4 where the mean of delivery rate is computed on the last 200 messages sent. In this experiment, the graph is composed of 100 nodes (average degree of 8) but on these 100 nodes, 10% are compromised and create a blackhole.

At the beginning of the execution of the `TLCN-S` protocol, values of counters are not significant and the `TLCN-S` protocol route messages following the same random policy as other random walks, so the delivery rate is similar. However, each time a message arrival is detected, the counter values of some links increase when the acknowlegment is received. The more time elapses, the more the counter values attached to the links leading to secure paths increase. As the probability of choosing one link varies in proportion to the value of its counter, then eventually messages avoid malicious nodes with high probability and the average delivery rate increases. Hence, in long-term execution, we can then observe a convergence phenomenon, *e.g.*, after 10^6 message sendings the delivery rate is above 95% and

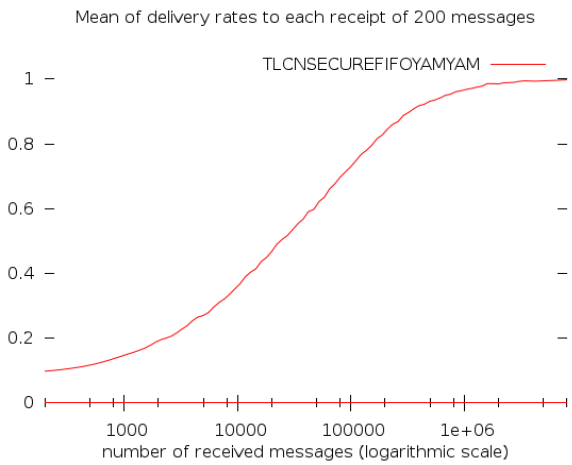

Figure 5.4: Convergence of delivery rate with the presence of 10% of malicious nodes

it continues to be improved.

We try to observe the influence of the distance between the sink and attackers in Figure 5.5. Attackers are positioned in two ways: either they are at distance 1 or 2 of the sink such that they enclose it or they are randomly positioned. Results have been achieved on random graphs of size 100 and average degree 8.

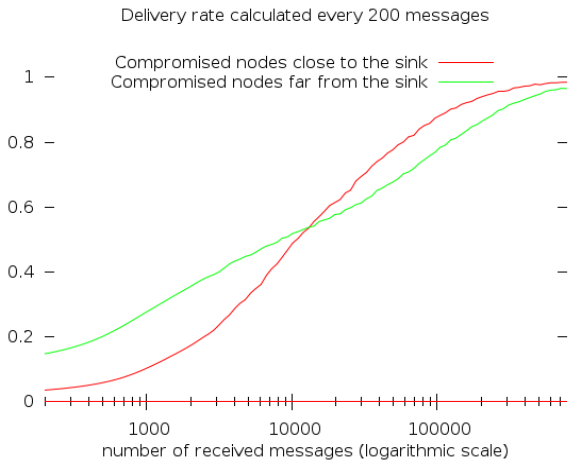

Figure 5.5: Influence of the distance between the sink and attackers

When malicious nodes are closed to the sink, they obtain an augmentation of their traffic because messages must pass through or beside them to attain the sink. As they receives more messages than nodes away from the sink, the delivery rate is more affected. But we also remark that `TLCN-S` solves this difficulty because one time that safe paths are detected, messages are conveyed on these routes. Furthermore, since safe paths are limited in the case of malicious nodes around the sink, `TLCN-S` quickly detects them and the delivery rate quickly converges to 100%.

In this section, we observed that our secure protocol TLCN−S provides a good solution against selective forwarding attacks. The delivery rate is drastically improved compared with other random walk routings. This probabilistic routing solution has the great advantage that if there exists a safe path leading to the sink, then the protocol detects it and improves the probability that the message follows it. In the future, it would be interesting to compare ourselves to other secure routing protocols against this type of attack.

5.4 A Specific Attack

TLCN−S uses additional information to route messages, so this implies a new weakness that can be used by adversaries. Indeed, nodes take into account counters in the probability law and these counters are linked with received acknowledgements. A strong adversary creates sinkhole disrupting the acknowledgement routing and then it launches a selective forwarding attack with a disastrous impact on the delivery rate. First, we precisely describe this attack and then we propose solutions to mitigate its effects. In the following, we denote by SSF[4] this attack.

5.4.1 Description

We first present the adversary model that can launch this attack. We consider that it is an active, insider and laptop class attacker. This attack is very efficient with many compromised node. The crucial point is that malicious nodes can share their memory with for example a common server or a channel of communication. Data messages are normally forwarded by the adversary and acknowledgements are selected to forward, until the outbreak of a second step where he stops forwarding both.

TLCN−S save paths considered as secure using acknowledgements. In the attack, the adversary deceives source nodes blocking acknowledgements that detect a path passing only through legitimate nodes and on the contrary, it forwards those that have not detected the passage of the message in a malicious node. The attack works as follows:

1. In first, we have the sinkhole step. Malicious nodes store identifiers of data messages (the hashed nonce) that reach them in their shared memory but they normally forward these data messages.

2. The sink receives data messages and returns acknowledgments in the network.

3. When a malicious node receives an acknowledgment, it verifies the presence of the data message ID in the shared memory. If it is present, then it forwards the message, else it does not forward it.

4. After a certain time, the sinkhole step finishes, *i.e.*, when the traffic has been attracted in malicious nodes. Consequently, the selective forwarding step starts: malicious nodes launched a selective forwarding attack (where data messages and acknowledgements are lost) which became much more efficient.

This attack is specific to the TLCN−S protocol and very dangerous. However, during the full lifespan of the WSN, we can assume that the delivery rate of TLCN−S is still better than other random walk routings. This is due to the fact that this attack occurs in two steps. In the first step, data messages reach the sink because the adversary only loses acknowledgements. In the second step, many messages are lost but on the total duration of these two steps, the delivery rate remains acceptable. The adversary must allow to pass enough data messages to deceive legitimate nodes. For example, in the experiment scenario described latter, the total delivery rate is of 66% in presence of 10% of malicious nodes.

[4] Sinkhole followed by Selective Forwarding

5.4.2 Countermeasures

We propose two ideas to mitigate this attack based on the fact that counters need to take into account behavior change of nodes during the entire network life. We cannot avoid that the adversary creates sinkholes but we can improve the detection of selective forwarding that they launch.

Reset the simulated annealing We recall that this method is described in Section 3.2.2 and that performances have not been verified by simulation. Since counters and especially the difference between counters always increase, then the TLCN−S protocol becomes more and more deterministic. We want to use the method of simulated annealing for that all to counters to have a chance to be incremented.

In the simulated annealing, the node chooses the next hop according to either the probability law of a random walk or the probability law depending on counters. It makes its choice based on a probability $p = \exp^{-\frac{nbAck}{\delta_u}}$. The idea is to reset this probability in order that the protocol regularly uses the random walk probability law. More explicitly, nodes regularly set to zero the number of acknowledgments received ($nbAck$) which is the evolution parameter of the simulated annealing probability. Nevertheless, the adversary is also able to switch between sinkhole and selective forwarding to suit its attack because it knows the timer reset. Consequently, nodes randomly reset timers to avoid this problem.

A second "temporary" counter A second idea is to introduce a second counter on links to check the last delivery rate. In other words, we count the number of received acknowledgements and compute the delivery rate every k messages sent by a link (k predefined), instead of to count the total number of received acknowledgements as in trust counters. When this temporary delivery rate is not sufficient, we penalize the trust counter. This method can be used with many different parameters and kinds of penalty. We list them like this:

1. The number k of messages sent by a link to check the temporary delivery rate.
2. The acceptable temporary delivery rate $tempRate$ which has to take into account the number of messages or acknowledgements still in the network.
3. The penalty to be imposed that can be:
   (a) Decrement the associated trust counter
   (b) Reset the simulated annealing
   (c) Set the associated trust counter to the mean of all counters.
   (d) Set the associated trust counter to the value of the minimum counter.

In the experiment scenario, source nodes send 189000 messages on random graphs of size 50 and average degree 8. There are 10% of malicious nodes. They are in sinkhole step until the sending of about 4500 messages and then they starts the selective forwarding step. We choose the following parameters arbitrarily and empirically to test our countermeasure: $k = 50$, $tempRate =$ and the penalty is the decrementation of the trust counter. Figure 5.6 gives us results of the experiment. We observe that the curve gradient of the TLCN−S using the countermeasure is steeper than the curve gradient of the basic TLCN−S. So, the delivery rate is improved with this defense. This is due to the fact that the countermeasure allows TLCN−S to detect the emergence of new malicious nodes.

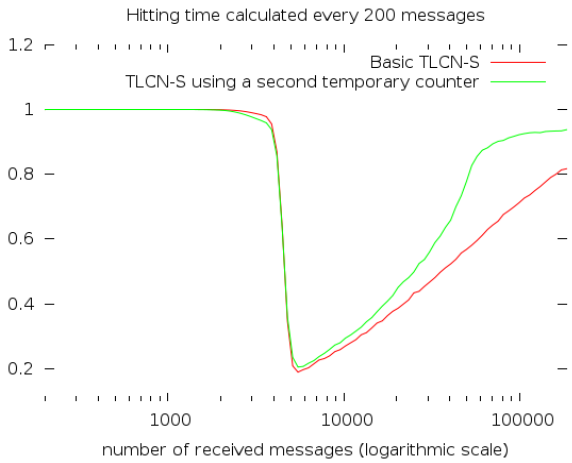

Figure 5.6: An efficient defense against SSF attack

Conclusion

In WSN, the service quality is crucial. The routing protocol must ensure a large delivery rate at the same time that the confidentiality, authenticity and integrity. Most of current protocols take into account these security properties but their cryptographic defenses are useless against insider attackers.

In this report, we constructed a new routing protocol that ensures security properties. Since TLCN−S is a probabilistic routing, the attacker is not able to predict the behavior of legitimate nodes. By cons, the probability that a message reaches a compromised node is not zero. TLCN−S thus uses tabu lists and trust counters to detect safe paths. It also uses the symmetric encryption, the hash and the nonce which are crucial for classic security properties despite their energy costs. Our results of experiments show an improvement of the delivery rate, even when many nodes are compromised in any location.

First, we investigated the security for the routing problem in WSNs. Then we focused on a proposed secure routing protocol using probabilistic methods. We summarize different work produced in this report as follows:

1. We present a state of the art on security requirements, the adversary model, attacks against routing protocols and the secure building blocks of the WSN literature.
2. We analyse the impacts of attacks on different random walk routing protocols.
3. We suggest an approach to reduce the message overhead due to tabu lists in TLM based on Bloom filters.
4. We proposed a new routing protocol TLCN−S using tabu lists in nodes to improve the number of message arrivals to the sink. This protocol does not need synchronization, geo-localization or key exchange. Moreover, we offer several possible optimizations to improve the energy conservation.
5. We prove data confidentiality, integrity and authenticity of TLCN−S in the random oracle model.
6. We study its hitting time and its delivery rate in presence of attackers by simulation.
7. We introduce a specific attack on TLCN−S and propose several countermeasures.

To conclude, the routing solution proposed in this report is well suited for WSNs because it seems to provide a good trade-off between security requirements and energy constraints.

In the future, we shall investigate the energy costs of TLCN−S according to its security levels. The protocol can be improved. We can explore avenues of research already presented. We also need to precisely adjust settings, for example, taking into account the type of applications and the network topology. Moreover, we can look for other attacks against TLCN−S and protect it. Its resiliency and its cost must be compared with other secure protocols of the literature. We can also eventually secure the TLM protocol with introduced proposals.

Bibliography

[ADGFT08] Marcos Kawazoe Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. On implementing omega in systems with weak reliability and synchrony assumptions. *Distributed Computing*, 21(4):285–314, 2008. 2.2, 2

[ADLP10] K. Altisen, S. Devismes, P. Lafourcade, and C. Ponsonnet. Routing algorithms using random walks with tabu lists. preprint, 2010. 2.1.3, 3, 5.1, 5.2

[AFN08] H. Alzaid, E. Foo, and J.G. Nieto. Secure data aggregation in wireless sensor network: a survey. In *Proceedings of the sixth Australasian conference on Information security-Volume 81*, pages 93–105. Australian Computer Society, Inc., 2008. 1.1.5

[BC94] S. Brands and D. Chaum. Distance-bounding protocols. In *Advances in Cryptology-EUROCRYPT'93*, pages 344–359. Springer, 1994. 1.4.3

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *focs*, page 394. Published by the IEEE Computer Society, 1997. 4, 4.2

[BHSS03] B. Blum, T. He, S. Son, and J. Stankovic. Igf: A state-free robust communication protocol for wireless sensor networks. 2003. 1.4.4

[BKR94] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In *Advances in Cryptology-CRYPTO 94*, pages 341–358. Springer, 1994. 3, 6

[Blo70] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970. 2.3

[Blo85] R. Blom. An optimal class of symmetric key generation systems. In *Advances in Cryptology: Proceedings of EUROCRYPT 84-A Workshop on the Theory and Application of Cryptographic Techniques, Paris, France, April 1984*, page 335. Springer, 1985. 1.4.2

[BSH+92] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. In *CRYPTO*, pages 471–486, 1992. 1.4.2

[BSW69] K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Commun. ACM*, 12(5):260–261, 1969. (document), 2

[CKM00] D.W. Carman, P.S. Kruus, and B.J. Matt. Constraints and approaches for distributed sensor network security (final). *DARPA Project report,(Cryptographic Technologies Group, Trusted Information System, NAI Labs)*, 2000. 1.1.1

[CMT05] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 109–117, Washington, DC, USA, 2005. IEEE Computer Society. 1.1.2

[CÖN+06] H. Cam, S. Özdemir, P. Nair, D. Muthuavinashiappan, and H. Ozgur Sanli. Energy-efficient secure pattern based data aggregation for wireless sensor networks. *Computer Communications*, 29(4):446–455, 2006. 1.1.2

[CP05] Haowen Chan and Adrian Perrig. Pike: peer intermediaries for key establishment in sensor networks. In *INFOCOM*, pages 524–535, 2005. 1.4.2

[CW79] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. 4

[DDK09] K. Daabaj, M. Dixon, and T. Koziniec. Experimental study of load balancing routing for improving lifetime in sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*, pages 1–4. IEEE, 2009. 1.1.4

[DHM06] J. Deng, R. Han, and S. Mishra. Insens: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2):216–230, 2006. (document), 1.4.4

[DR02] J. Daemen and V. Rijmen. *The design of Rijndael: AES–the advanced encryption standard*. Springer Verlag, 2002. 4

[DR06] F. Deng and D. Rafiei. Approximately detecting duplicates for streaming data using stable bloom filters. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 25–36. ACM, 2006. 2.3.1

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983. 1.2.1

[EG02] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *ACM Conference on Computer and Communications Security*, pages 41–47, 2002. 1.1.1, 1.4.2

[EOMVK10] Ochirkhand Erdene Ochir, Marine Minier, Fabrice Valois, and Apostolos Kountouris. Resilient networking in wireless sensor networks. Research Report RR-7230, INRIA, 03 2010. 1.1.4, 1.2, 2

[FCAB00] L. Fan, P. Cao, J. Almeida, and A.Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking (TON)*, 8(3):281–293, 2000. 2.3.1

[GGSE01] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '01, pages 251–254, New York, NY, USA, 2001. ACM. (document), 1.1.4, 1.4.4

[GKS04] Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar. Public key cryptography in sensor networks - revisited. In *ESAS*, pages 2–18, 2004. 1.4.2

[GL05] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 620–629. IEEE, 2005. 1.1.5

[GPW+04] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *CHES*, pages 119–132, 2004. 1.4.2

[Gro08] Distributed Computing Group. Sinalgo - simulator for network algorithms, 2008. http://disco.ethz.ch/projects/sinalgo/. 5.1

[Has70] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97, 1970. 3.2.2

[HCK+03] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 141–150. ACM, 2003. 1.4.2

[HE04] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Network and Distributed System Security Symposium (NDSS)*, pages 131–141. Citeseer, 2004. 1.4.3

[HG06] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 194–205. IEEE, 2006. 1.1.5

[How87] R. Howard. Data encryption standard. *Information age*, 9(4):204–210, 1987. 4

[HPJ03] Y.C. Hu, A. Perrig, and D.B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1976–1986. Ieee, 2003. 1.4.3

[JKK08] H. Juma, I. Kamel, and L. Kaya. On protecting the integrity of sensor data. In *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, pages 902–905. IEEE, 2008. 1.1.3

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671, 1983. 3.2.2

[KKT09] A. Kapadia, D. Kotz, and N. Triandopoulos. Opportunistic sensing: Security challenges for the new paradigm. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–10. IEEE, 2009. 1.1.5

[KSW04] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys*, pages 162–175, 2004. 1.4.1, 2.3.2

[KW03] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, September 2003. 1.2.2, 1.2.3, 1.3

[LDH06] Y.W. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):65–93, 2006. C, C.6

[LMPG07] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil D. Gligor. Minisec: a secure sensor network communication architecture. In *IPSN*, pages 479–488, 2007. 1.4.1, 2.3.2

[LNL05] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(1):41–77, 2005. 1.4.2

[LS05] Jooyoung Lee and Douglas R. Stinson. Tree-based key distribution patterns. In *Selected Areas in Cryptography*, pages 189–204, 2005. 1.4.2

[LSP82] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982. 1

[LY06] X. Li and D. Yang. A quantitative survivability evaluation model for wireless sensor networks. In *Networking, Sensing and Control, 2006. ICNSC'06. Proceedings of the 2006 IEEE International Conference on*, pages 727–732. IEEE, 2006. 1.1.4

[LZDT09] N. Li, N. Zhang, S.K. Das, and B. Thuraisingham. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Networks*, 7(8):1501–1514, 2009. 1.1.5

[MEL+00] N.R. Mead, R.J. Ellison, R.C. Linger, T. Longstaff, J. McHugh, and CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST. *Survivable network analysis method*. Citeseer, 2000. 1.1.4

[MGLB00] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265. ACM, 2000. 2.3.2

[MRR+53] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, et al. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087, 1953. 3.2.2

[NSSP04] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 259–268. ACM, 2004. 1.3.4

[PCTS02] Adrian Perrig, Ran Canetti, J.D̃. Tygar, and Dawn Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5(Summer), 2002. 1.4.1

[PH09] D. Puccinelli and M. Haenggi. Lifetime benefits through load balancing in homogeneous sensor networks. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1–6. IEEE, 2009. 1.1.4

[PLGP06] B. Parno, M. Luk, E. Gaustad, and A. Perrig. Secure sensor network routing: A clean-slate approach. In *Proceedings of the 2006 ACM CoNEXT conference*, page 11. ACM, 2006. 1.4.4

[PLOP08] T. Pazynyuk, J.Z. Li, G.S. Oreku, and L.Q. Pan. Qos as means of providing wsns security. In *Seventh International Conference on Networking*, pages 66–71. IEEE, 2008. 1.1.4

[PRRJ06] N.R. Potlapally, S. Ravi, A. Raghunathan, and N.K. Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing*, pages 128–143, 2006. C, C.1, C.2, C.3

[PST+02] A. Perrig, R. Szewczyk, JD Tygar, V. Wen, and D.E. Culler. Spins: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002. (document), 1.4.1, 2.3.2

[PSW04] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6):53–57, 2004. 1.1.1, 1.1.4, 1.1.5

[RALS11] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering*, 37(2):147–159, 2011. 1.4.2

[RPC+07] T. Roosta, S. Pai, P. Chen, S. Sastry, and S. Wicker. Inherent security of routing protocols in ad-hoc and sensor networks. In *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pages 1273–1278. IEEE, 2007. 1.4.4

[SB02] S.D. Servetto and G. Barrenechea. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 12–21. ACM, 2002. 2

[SCK07] G. Shobha, R.R. Chittal, and K. Kumar. Medical Applications of Wireless Networks. In *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on*, page 82. IEEE, 2007. 1.1.5

[Sen11] J. Sen. Routing Security Issues in Wireless Sensor Networks: Attacks and Defenses. *Arxiv preprint arXiv:1101.2759*, 2011. 1.1.3, 1.1.5

[Sho04] V. Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004. 4

[SIY09] Izumi Kubo Satoshi Ikeda and Masafumi Yamashita. The hitting and cover times of random walks on finite graphs using local degree information. *Theoretical Computer Science*, 410:94–100, 2009. Contents lists available at ScienceDirect. 2.1.2, 2.2.2

[SJBMC10] M.A. Simplício Jr, P.S.L.M. Barreto, C.B. Margi, and T.C.M.B. Carvalho. A survey on key management mechanisms for distributed wireless sensor networks. *Computer Networks*, 54(15):2591–2612, 2010. 1.4.2

[SLSZ06] R.A. Shaikh, S. Lee, Y.J. Song, and Y. Zhung. Securing distributed wireless sensor networks: issues and guidelines. 2006. 1.1.5, 1.1.5

[TWZL08] C.C. Tan, H. Wang, S. Zhong, and Q. Li. Body sensor network security: an identity-based cryptography approach. In *Proceedings of the first ACM conference on Wireless network security*, pages 148–153. ACM, 2008. 1.1.5

[UIT91] UIT. Security architecture for open systems interconnection for ccitt applications, 1991. 1.1, 1.1.2, 1.1.5

[Wag04] D. Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87. ACM, 2004. 1.1.4

[WC81] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981. 4

[WCG+05] X. Wang, S. Chellappan, W. Gu, W. Yu, and D. Xuan. Search-based physical attacks in sensor networks. In *Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on*, pages 489–496. IEEE, 2005. 1.2.4

[WDXZ10] T. Wang, H. Dai, K. Xiang, and B.Y. Zou. Network system survivability survey: An evolution approach. In *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, volume 1, page V1. IEEE, 2010. 1.1.4

[WFSH06] Anthony D. Wood, Lei Fang, John A. Stankovic, and Tian He. Sigf: a family of configurable, secure routing protocols for wireless sensor networks. In *SASN*, pages 35–48, 2006. (document), 1.4.4

[WGE+05] A.S. Wander, N. Gura, H. Eberle, V. Gupta, and S.C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. 2005. C, C.4, C.5

[WHA99] D. Wallner, E. Harder, and R. Agee. Rfc2627: Key management for multicast: issues and architectures. *RFC Editor United States*, 1999. 1.4.2

[WKfC+04] Ronald J. Watro, Derrick Kong, Sue fen Cuti, Charles Gardiner, Charles Lynn, and Peter Kruus. Tinypk: securing sensor networks with public key technology. In *SASN*, pages 59–64, 2004. 1.4.2

[WS02] A.D. Wood and J.A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002. 1.1.4

[WS04] A.D. Wood and J.A. Stankovic. A taxonomy for denial-of-service attacks in wireless sensor networks. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, pages 739–763, 2004. 1.2.5

[XMTZ06] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *Network, IEEE*, 20(3):41–47, 2006. 1.4.3

[XyQLx09] Y. Xiao-yuan, Z. Qian, and W. Li-xian. A Robust Entity Authentication in Wireless Sensor Networks. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, pages 1–4. IEEE, 2009. 1.1.2

[YG05] Z. Yu and Y. Guan. A robust group-based key management scheme for wireless sensor networks. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 4, pages 1915–1920. IEEE, 2005. 1.4.2

[YWZC08] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 11(4):1–43, 2008. 1.1.4

[ZLLF06] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Location-based compromise-tolerant security mechanisms for wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):247–260, 2006. (document)

[ZSJ03] S. Zhu, S. Setia, and S. Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 62–72. ACM, 2003. 1.4.2

[ZV08] Junqi Zhang and Vijay Varadharajan. A new security scheme for wireless sensor networks. In *GLOBECOM*, pages 128–132, 2008. 1.4.2

[ZV10] J. Zhang and V. Varadharajan. Wireless sensor network key management survey and taxonomy. *Journal of Network and Computer Applications*, 33(2):63–75, 2010. 1.4.2

[ZZF07] Y. Zhou, Y. Zhang, and Y. Fang. Access control in wireless sensor networks. *Ad Hoc Networks*, 5(1):3–13, 2007. 1.1.5

Appendix A

Applications of WSNs

We give a short overview of WSNs applications. At the beginning, this technology was developed for a military application to monitor a battlefield (see Figure A.1).

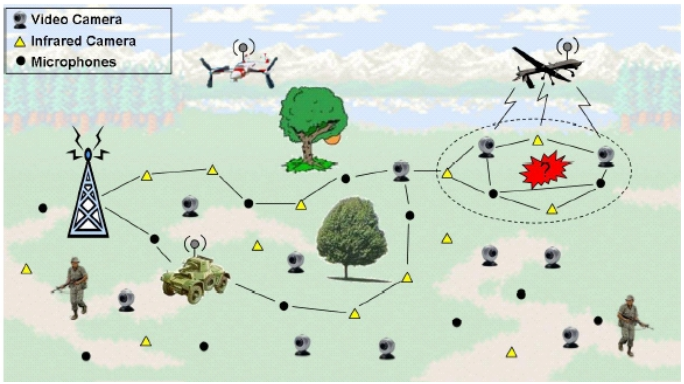

Figure A.1: Battlefield Monitoring

Now, WSNs can be used for many applications such as environmental monitoring, structure monitoring or medical care. For example, a research group from Harvard deployed a WSN as in Figure A.2 to monitor a volcano in Ecuador. This was motivated by the fact that the hostile environment discourages the human presence. More information on this project are available at http://fiji.eecs.harvard.edu/Volcano.

32 sensors are also deployed on great duck island to monitor the Seabird Environmental (see Figure A.3). Researchers use a satellite link connected to Internet to collect data such as temperature, pressure, humidity,...

This technology is expanding in the medical world. For example, Intel created a network of 130 sensors to monitor the activity of seniors in nursing homes.

There is even WSNs in Grenoble where sensors provide statistic information on water and electricity consumption. The website is available at http://www.senscity-grenoble.com

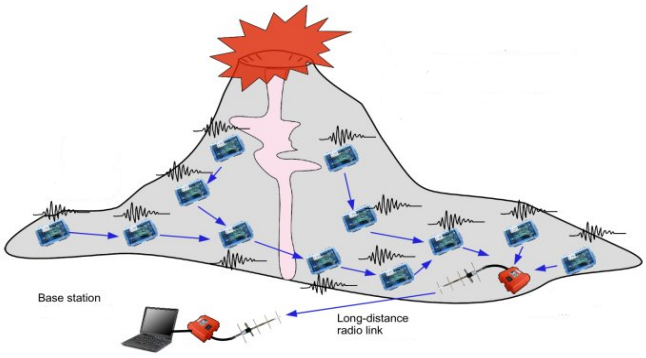

Figure A.2: Volcano and Earthquake Monitoring

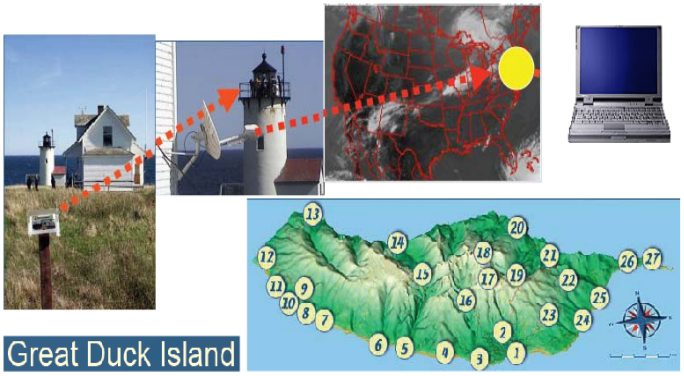

Figure A.3: Environmental Monitoring

Appendix B

Features of Sensors

First, we describe all measures that are possible with wireless sensors in Figure B.1. Then we give a comparison of different wireless sensors in Figure B.2.

Measurements for Wireless Sensor Networks		
	Measurand	**Transduction Principle**
Physical Properties	Pressure	Piezoresistive, capacitive
	Temperature	Thermistor, thermo-mechanical, thermocouple
	Humidity	Resistive, capacitive
	Flow	Pressure change, thermistor
Motion Properties	Position	E-mag, GPS, contact sensor
	Velocity	Doppler, Hall effect, optoelectronic
	Angular velocity	Optical encoder
	Acceleration	Piezoresistive, piezoelectric, optical fiber
Contact Properties	Strain	Piezoresistive
	Force	Piezoelectric, piezoresistive
	Torque	Piezoresistive, optoelectronic
	Slip	Dual torque
	Vibration	Piezoresistive, piezoelectric, optical fiber, Sound, ultrasound
Presence	Tactile/contact	Contact switch, capacitive
	Proximity	Hall effect, capacitive, magnetic, seismic, acoustic, RF
	Distance/range	E-mag (sonar, radar, lidar), magnetic, tunneling
	Motion	E-mag, IR, acoustic, seismic (vibration)
Biochemical	Biochemical agents	Biochemical transduction
Identification	Personal features	Vision
	Personal ID	Fingerprints, retinal scan, voice, heat plume, vision motion analysis

Figure B.1: Data available

	Btnode 3	mica2	mica2dot	micaz	telos A	tmote sky	EYES
Manufacturer	Art of Technology	Crossbow			Imote iv		Univ. of Twente
Microcontroller	Atmel Atmega 128L				Texas Instruments MSP430		
Clock frequency	7.37 Mhz		4 MHz	7.37 MHz	8 MHz		5 MHz
RAM (KB)	64 + 180	4	4	4	2	10	2
ROM (KB)	128	128	128	128	60	48	60
Storage (KB)	4	512	512	512	256	1024	4
Radio	Chipcon CC1000 315/433/868/916 MHz 38.4 Kbauds			Chipcon CC2420 2.4 GHz 250Kbps IEEE 802.15.4			RFM TR1001868 MHz 57.6 Kbps
Max Range (m)	150-300			75-100			
Power	2 AA batteries		Coin cell	2 AA Batteries			
PC connector	Through PC-connected programming board				USB		Serial Port
OS	Nut/OS	TinyOS					PEEROS
Transducers	On acquisition board				On board		On acquisition board
Extras	+ Bluetooth radio						

Figure B.2: Current Wireless Sensors

Appendix C

Cryptographic Primitives

We present energy consumption of cryptographic primitives published in [PRRJ06] and public key cryptography optimized for WSNs in [WGE+05].

Algorithm	Key size ($bits$)	Key generation (mJ)	Sign (mJ)	Verify (mJ)
RSA	1,024	270.13	546.50	15.97
DSA	1,024	293.20	313.60	338.02
ECDSA	163	226.65	134.20	196.23
ECDSA	193	281.65	166.75	243.84
ECDSA	233	323.30	191.37	279.82
ECDSA	283	504.96	298.86	437.00
ECDSA	409	1034.92	611.40	895.98

Figure C.1: Public key cryptography from [PRRJ06]

Algorithm	MD2	MD4	MD5	SHA	SHA1	HMAC
Energy (μJ/B)	4.12	0.52	0.59	0.75	0.76	1.16

Figure C.2: Hash functions studied from [PRRJ06]

	DES	3DES	IDEA	CAST	AES	RC2	RC4	RC5	BLOW FISH	
Key Setup	27.53	87.04	7.96	37.63	7.87	32.94	95.97	66.54	3166.3	(µJ)
Enc/Dec	2.08	6.04	1.47	1.47	1.21	1.73	3.93	0.79	0.81	(µJ/byte)

Figure C.3: Symmetric key cryptography from [PRRJ06]

We remark that energy consumption of public key cryptography is about 1000 times larger than energy consumption of symmetric key cryptography. The use of block ciphers is thus well-justified. In [LDH06], a team from the university of twente has classified block ciphers for WSNs according to their physical constraints in Figure C.6.

Algorithm	Energy
SHA-1	5.9 µJ/byte
AES-128 Enc/Dec	1.62/2.49 µJ/byte

Figure C.4: Example of symmetric and hash from [WGE+05]

Algorithm	Signature	
	Sign	Verify
RSA-1024	304	11.9
ECDSA-160	22.82	45.09[3]
RSA-2048	2302.7	53.7
ECDSA-224	61.54	121.98[3]

Figure C.5: Public key cryptography optimized in mJ from [WGE+05]

By key setup:

Rank	Size optimised			Speed optimised		
	Code mem.	Data mem.	Speed	Code mem.	Data mem.	Speed
1	RC5-32	MISTY1	MISTY1	RC6-32	MISTY1	MISTY1
2	KASUMI	Rijndael	Rijndael	KASUMI	Rijndael	Rijndael
3	RC6-32	KASUMI	KASUMI	RC5-32	KASUMI	KASUMI
4	MISTY1	RC6-32	Camellia	MISTY1	RC6-32	Camellia
5	Rijndael	RC5-32	RC5-32	Rijndael	Camellia	RC5-32
6	Camellia	Camellia	RC6-32	Camellia	RC5-32	RC6-32

By encryption (CBC/CFB/OFB/CTR):

Rank	Size optimised			Speed optimised		
	Code mem.	Data mem.	Speed	Code mem.	Data mem.	Speed
1	RC5-32	RC5-32	Rijndael	RC6-32	RC5-32	Rijndael
2	RC6-32	MISTY1	MISTY1	RC5-32	MISTY1	Camellia
3	MISTY1	KASUMI	KASUMI	MISTY1	KASUMI	MISTY1
4	KASUMI	RC6-32	Camellia	KASUMI	RC6-32	RC5-32
5	Rijndael	Rijndael	RC6-32	Rijndael	Rijndael	KASUMI
6	Camellia	Camellia	RC5-32	Camellia	Camellia	RC6-32

Figure C.6: Ranking of block ciphers from [LDH06]